Multi-Modal and Deep Learning for Robust Speech Recognition

by

Xue Feng

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author . . . .
Department of Electrical Engineering and Computer Science
August 31, 2017

Certified by . . . .
James R. Glass
Senior Research Scientist
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Multi-Modal and Deep Learning for Robust Speech Recognition

by

Xue Feng

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Automatic speech recognition (ASR) decodes speech signals into text. While ASR can produce accurate word recognition in clean environments, system performance can degrade dramatically when noise and reverberation are present. In this thesis, speech denoising and model adaptation for robust speech recognition were studied, and four novel methods were introduced to improve ASR robustness.

First, we developed an ASR system using multi-channel information from microphone arrays via accurate speaker tracking with Kalman filtering and subsequent beamforming. The system was evaluated on the publicly available Reverb Challenge corpus, and placed second (out of 49 submitted systems) in the recognition task on real data.

Second, we explored a speech feature denoising and dereverberation method via deep denoising autoencoders (DDA). The method was evaluated on the CHiME2-WSJ0 corpus and achieved a 16% to 25% absolute improvement in word error rate (WER) compared to the baseline.

Third, we developed a method to incorporate heterogeneous multi-modal data with a deep neural network (DNN) based acoustic model. Our experiments on a noisy vehicle-based speech corpus demonstrated that WERs can be reduced by 6.3% relative to the baseline system.

Finally, we explored the use of a low-dimensional environmentally-aware feature derived from the total acoustic variability space. Two extraction methods are presented: one via linear discriminant analysis (LDA) projection, and the other via a bottleneck deep neural network (BN-DNN). Our evaluations showed that by adapting ASR systems with the proposed feature, ASR performance was significantly improved. We also demonstrated that the proposed feature yielded promising results on environment identification tasks.

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist

Acknowledgments

I would never have been able to finish my dissertation without the support from my advisor, committee members, colleagues, friends, and family.

First, I would like to express my sincere gratitude to my advisor Jim Glass for his continuous support of my Ph.D study, for his knowledge, patience and motivation. I still remember the moment I met with Jim for the first time five years ago, as an attempt to change research directions after receiving my Master's degree at MIT. This is one of the best decisions I've ever made, and I could not have imagined having a better advisor and mentor for my Ph.D study. Research-wise, Jim has provided many high level insights, not to mention the detailed help he provided on experiments and thesis writing. For example, the e-vector work started off mainly from Jim's intuition. I also appreciate Jim's encouragement and patience when I tend to procrastinate. In terms of personality, I have learned so much from Jim, on how to always remain calm under stress and pressure, and how to speak softly but still with an opinion.

I also would like to thank my committee, Regina Barzilay, Vivienne Sze, and Victor Zue, for their feedback in our committee meetings and the comments that help broaden and improve this thesis.

I would like to thank my fellow doctoral students for their feedback, cooperation and friendship. I am extremely fortunate to be a part of the Spoken Language Systems (SLS) group. Each member has helped me in many ways, from answering all my questions, whether in research or in life, to creating an enjoyable working environment. Several works in this thesis relied on GPU computing, and I am very grateful to the people in SLS who spent their time maintaining the GPU machines and updating software. Many thanks to Dave, Ekapol, Stephen, Tuka, Yaodong and Yu for sharing various scripts that contributed to the work in this thesis. Thanks to Mandy and Tuka for proofreading this thesis, which was very helpful. Thanks to Xiao, whom I also met in SLS, for being my flatmate for three years and supporting me in almost all conceivable ways.

My sincere thanks also goes to external collaborators. Thanks to John Mcdonough and Kenichi Kumatani for working on the multi-channel speech recognition system together, and to Brigitte Richardson and Scott Amman for providing the Ford speech dataset.

Last but not least, I would like to thank my parents for their constant love and support, their encouraging me to strive for the best, and their giving me the freedom to pursue whatever path I like.

This work is sponsored by Ford Motor Company and Quanta Research.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

With the advancement of technology, various machines have been invented and adopted to ease human lives. Yet, the natural communication with these machines has been a collective dream since their first existence. From prehistory to the digital age, speech communication has been the dominant mode of human bonding and information exchange, so it follows that speech would be extended to human-machine interaction.

The first step towards speech communication with machines is Automatic Speech Recognition (ASR), which is the transcription of speech signals into word sequences. The objective of ASR is to recognize human speech, such as words and sentences, using algorithms executed by a computer. ASR is essentially a multi-class sequential pattern recognition task. The features are usually a sequence of representative vectors that are extracted from speech signals, and the classes are either words or sub-word units such as phonemes.

In contrast to the development of the first speech synthesizer in 1936 by AT&T, the first automatic speech recognizer, a simple digit recognizer, did not appear until 1952 [3]. By 1969, John Pierce of Bell Labs claimed that ASR would not be a reality for several decades, however, the 1970s witnessed a significant theoretical breakthrough in speech recognition - Hidden Markov Models (HMMs) [4]. Since then, the multidisciplinary field of ASR has proceeded from its infancy to its coming of age and into a quickly growing number of practical applications and commercial markets. HMMs were extensively investigated and became the most successful technique for acoustic modeling in speech recognition for many decades. The maximum likelihood based Expectation Maximization (EM) algorithm and the forward-backward (Baum-Welch) algorithm have been the principal means by which HMMs are trained. Over the past few years the success of deep neural networks (DNNs) has further boosted the recognition performance, and has already reached human parity on the benchmark TIMIT corpus [5–7].

■ 1.1 Motivation

State-of-the-art ASR systems are known to perform well when the speech signals are captured in a noise-free environment, and using a close-talking microphone worn near the mouth of the speaker. While speech technology has progressed to the point where commercial systems have been deployed (office-place dictation software, smart-phone assistants), a majority of our voice-based interactions occur in less than ideal conditions, thus making ASR far from being a solved problem.

■ 1.1.1 Rising Demand

There is a continuously growing demand for hands-free speech input for various applications [8,9]. One driving force behind this development is the increasing use of portable devices such as smart-phones and tablets, as dictated by market demands for more sophisticated devices. The adoption of such devices has lead to a surge in data available to industries and government, motivating investments in data analytics to improve services and to meet market needs, which in turn fuels the demand for more sophisticated devices. This trend is particularly visible in the domain of speech technology, where consumers are expecting better speech-based interfaces not only on their portable devices, but also in their cars and homes.

■ 1.1.2 Challenges

Cars, homes, and other such environments often involve distant microphones in which either safety or convenience preclude the use of a close-talking microphone. For example, while operating a vehicle, the very act of putting on a microphone is distracting and dangerous for the driver. The car interior is practically reflection-less[1], but both the running engine and the airflow over the car's exterior introduce strong additive noise, which significantly degrades system performance. Another example is in a living room, where microphones may restrict the movement of users and overall detracts from the organic nature that is expected of speech communication. However, distant microphones

[1]due to cabin padding material absorbing sound, and small-space minimizes reverberation time.

are usually embedded in devices themselves, in the walls, or ceiling, and are prone to capturing additive and convolution distortions such echoes and reverberations.

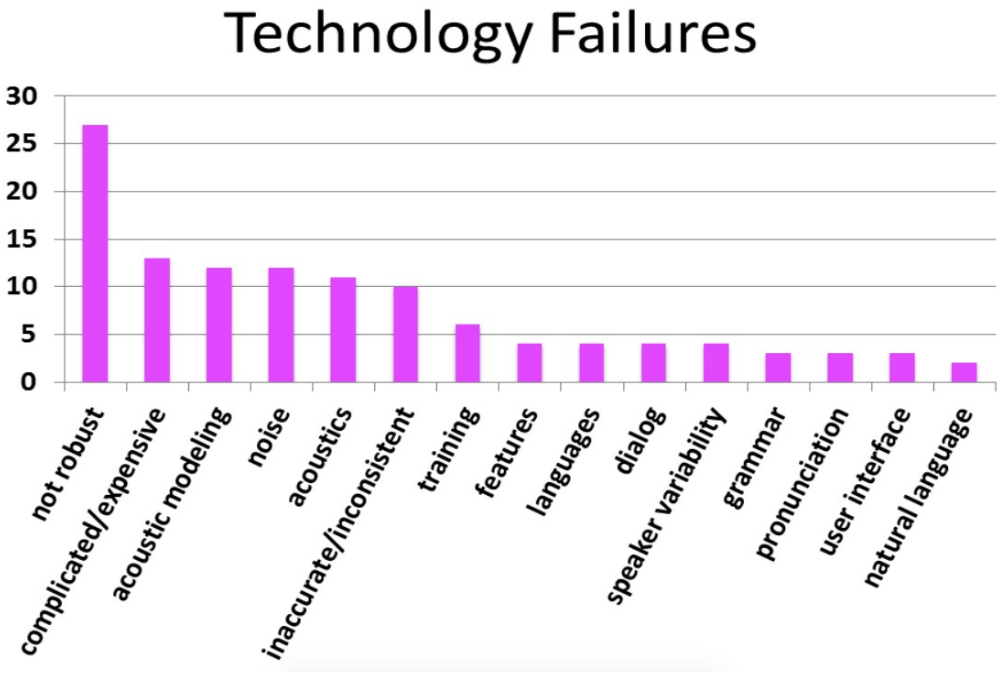

Figure 1-1: Technology failures in current automatic speech recognition according to expert survey reported in [1]

In a recent survey, Fig. 1-1, users of speech and language systems were asked to identify where the current technology had failed. While several components in the framework were pointed out, many of the informants identified the lack of **robustness** as a primary failure of the technology [1].

In the context of this thesis, we consider robustness as an ASR system's ability to overcame challenges introduced by noisy environments. A lack of robustness in ASR may be due to many factors, however, a major source comes from noise distortions [10]. It has been observed that ASR systems trained from clean speech usually degrade significantly when tested on noisy data [11]. When speech signals are corrupted by noise, the corruption is propagated to the extracted speech features ultimately deteriorating ASR performance. Given users' sensitivities to poor ASR performance, it becomes necessary to address this problem in order to enable the deployment of ASR systems for everyday use.

■ 1.2 Scope of the Thesis

This thesis is focused on tackling the lack of robustness in distant-talking ASR systems operating in various noisy environments. The performance degradation in these scenarios is mainly due to the statistical mismatch between the noisy test speech features and the clean-trained acoustic model of the ASR system. We investigated both feature-based compensation methods as well as model-based compensation methods to combat this mismatch. In particular, we were interested in those environments in which multi-modal data existed, such as spatial data coming from a microphone array, and sensory data from a noise source (e.g. engine, airflow). Effective solutions to several common distant speech applications are provided in this thesis.

■ 1.2.1 Contributions

The goal of this thesis is to improve the overall performance of ASR in noisy reverberant environment. The main contributions in this thesis are

1. Built a multi-channel ASR system that is robust to noise by fusing multi-channel information with speaker tracking and beamforming.

2. Developed a Deep Denoising Autoencoder based method for learning feature denoising transformations from parallel noisy and clean speech.

3. Provided an effective method to incorporate heterogeneous multi-modal data with a DNN-based acoustic model and improved ASR performance.

4. Proposed a novel low-dimensional environmentally-aware feature derived from the total acoustic variability space.

■ 1.3 Outline of this Thesis

In this thesis, four novel techniques are proposed to improve ASR performance in noisy conditions, and will be discussed in the upcoming chapters as follows.

Chapter 2 reviews the existing literature related to the research work presented in this thesis. In **Chapter 3** , we propose an ASR system that improves front-end feature quality by fusing multi-channel data using several techniques. **Chapter 4** presents a front-end denoising method called deep denoising autoencoder (DDA), which effectively learns a stochastic mapping from noisy features to clean features. **Chapter 5** introduces the idea of a heterogeneous DNN, that harvests heterogeneous data, in parallel to speech, to improve robustness. **Chapter 6** proposes a novel environment-aware feature representation, called an E-vector, derived from the total variability space, which can be used for noisy environment adaptation.

Chapter 2

ASR System Review

This chapter provides background on techniques used in ASR systems. Figure 2-1 depicts a typical block diagram of a state-of-the-art ASR system. We first introduce signal processing and conventional feature extractions in Section 2.1. Next acoustic modeling using HMMs are discussed in Section 2.2. We cover both the most commonly used GMM-HMM model, as well as the recently popular DNN-HMM model. Then we briefly present language modeling, decoding, and the evaluation metric used in ASR systems in Section 2.3. Finally, the current techniques and challenges for robust ASR are discussed in Section 2.4.

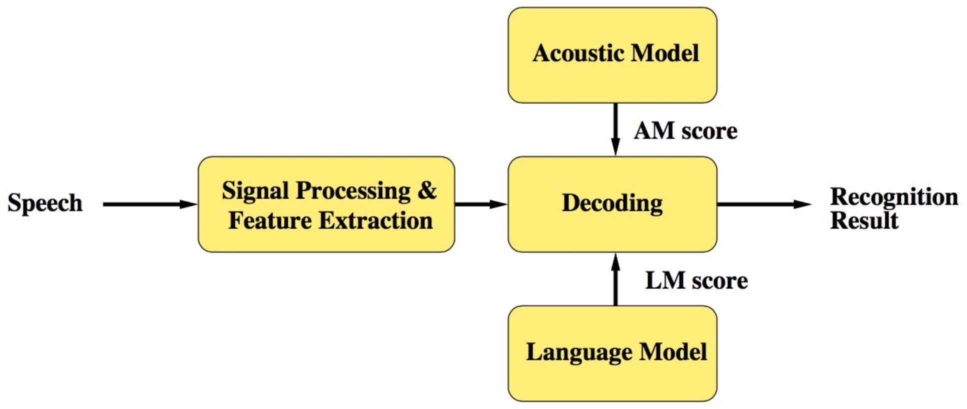

Figure 2-1: Block diagram of a general ASR system.

■ 2.1 Pre-processing and Feature Extraction

Speech processing is the first stage in a speech recognition system. The aim of front-end processing is to extract features that are optimal for the recognition task and (ideally) invariant to irrelevant factors, such as speaker differences and environment distortions.

When speech is recorded by a microphone, the signal is first digitized and represented by discrete amplitudes as a function of time given a fixed sampling rate. Most modern speech recording devices have a default sampling rate of at least 16kHz for human speech, while the standard telephone speech coding method only supports 8kHz in order to save transmission bandwidth. From the statistical learning point of view, with a high sampling rate of 16kHz or 8kHz, it is difficult to process speech directly from the waveform. Therefore, there have been a number of signal processing methods focusing on converting the speech waveform to a short-time spectral representation. A spectral representation has inherent advantages such as having lower dimensionality, yet preserving relevant phonetic information [12].

■ 2.1.1 Common Speech Features

Mel-frequency cepstral coefficients (MFCCs) and perceptual linear prediction (PLP) coefficients are two commonly used speech feature representations in speech recognition systems. They will be discussed in more detail in the following paragraphs.

Mel-frequency Cepstral Coefficients

Mel-frequency cepstral coefficients (MFCCs) are one of the widely used spectral representations for ASR and have become a standard front-end module for feature extraction in most modern ASR systems. In order to compute MFCCs for a recorded speech signal $x[t]$, the following standard steps are applied. Figure 2-2 depicts the steps in the procedure.

First, the speech waveform is normalized and then pre-emphasised. A common preprocessing approach is to apply mean and magnitude normalization:

$$x_n[t] = \frac{x[t] - mean(x[t])}{max\{x[t]\}} \tag{2.1}$$

This is followed by a pre-emphasis filter:

$$x_p[t] = x_n[t] - 0.97x_n[t] \tag{2.2}$$

Then a short-time Fourier transform (STFT) is applied:

$$X_{STFT}[t,k] = \sum_{m=-\infty}^{\infty} x_p[t]w[t-m]e^{-2\pi mk/N} \tag{2.3}$$

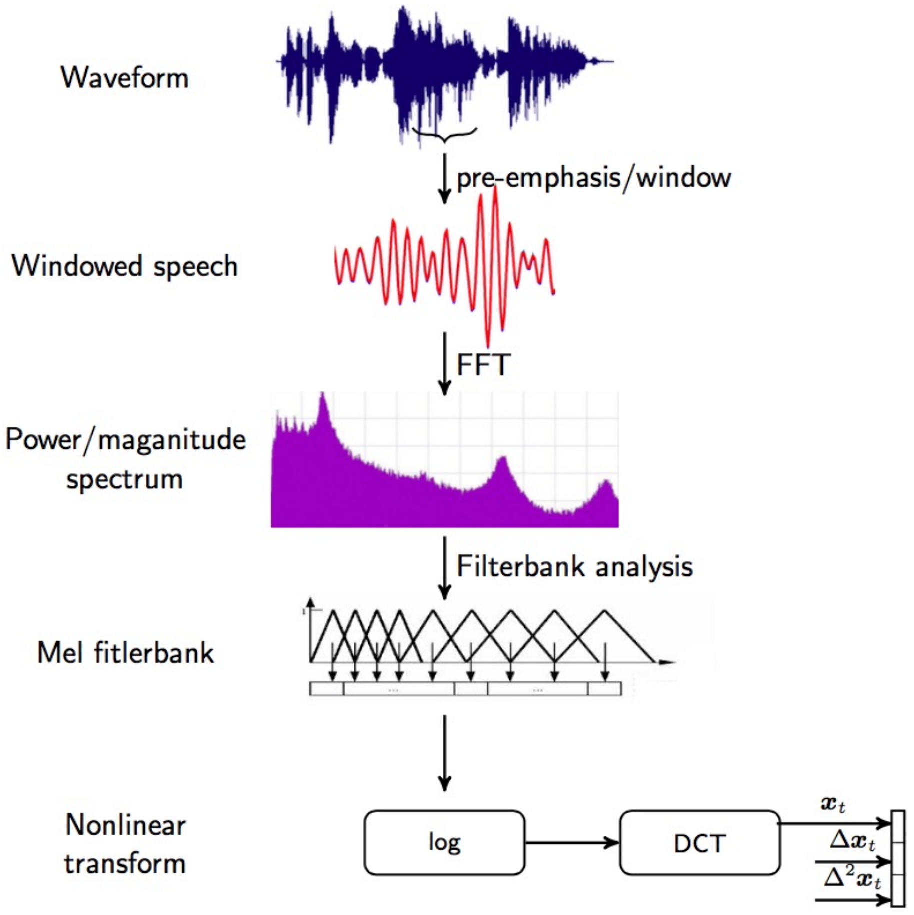

Figure 2-2: The MFCC feature extraction procedure

where w is the Hamming window, N is the number of points for the discrete Fourier transform (DFT) and $X_{STFT}[t,k]$ is the k-th spectral component at time t. For ASR, the STFT is usually performed with a window size of 25ms, and an analysis shift of 10ms. The most commonly used window is the Hamming window. In the following chapters, each analysis is often referred as a speech frame.

Third, Mel-frequency spectral coefficients (MFSCs) are calculated. The Mel-frequency filter is designed based on an approximation of the frequency response of the inner ear [85]. The Mel-filter frequency response is shown in Figure 2-2. On each speech frame after the STFT, a Mel-frequency filter is used to reduce the spectral resolution, and convert all frequency components according to the Mel-scale.

$$X_{MFSC}[t,i] = \frac{\sum_{k=-\infty}^{\infty} |X_{STFT}[t,k]|^2}{|M_i|} \tag{2.4}$$

Last, the Discrete Cosine Transform (DCT) is applied to the logarithm of the MFCSs to further reduce the dimensionality of the spectral vector. Typically only the first 12 DCT coefficients are kept.

$$X_{MFCC}[t,i] = \frac{\sum_{k=0}^{M-1} 10\log_{10}(X_{MFSC}[t,i])\cos(\frac{2\pi}{M}ki)}{M} \tag{2.5}$$

Perceptual Linear Prediction

PLP coefficients are another popular feature representation based on the short-time spectrum analysis. In PLP, the linear frequency of the power or magnitude spectrum is wrapped onto another perceptually motivated scale, the Bark frequency scale, via:

$$f_{bark} = 6\log\left(\left[\frac{f_{Hz}}{600}+1\right]^{0.5} + \frac{f_{Hz}}{600}\right). \tag{2.6}$$

Critical band filters spaced equally in the Bark frequency scale are used to filter the power or magnitude spectrum. The output of these filters are non-linearly transformed, based on an equal-loudness and intensity-loudness power law. Linear prediction (LP) analysis is applied and the resulting LP coefficients are converted to cepstral coefficients [13].

■ 2.1.2 Feature Post-processing

The MFCC or PLP features discussed in the previous section are often referred to as static features. Due to the conditional independence assumption in HMMs, if only static features are used, dynamic information in speech will not be incorporated into recognition systems. A simple way to address this limitation is to append dynamic features, such as delta and delta-delta features, to the base, static, features. Delta features δx_t can be computed using simple differences, e.g., $\Delta x_t = x_t - x_{t-1}$, or using a linear regression:

$$\Delta x_t = \frac{\sum_{i=1}^{w} i(x_{t+i} - x_{t-i})}{2i^2}$$

where w is the window length. It is clear that this delta feature is a linear combination of $2w + 1$ static features. Higher order coefficients, such as delta-delta features, can be calculated using the delta features in a similar way. In many state-of-the-art ASR systems, a 13-dimensional MFCC static feature vector is concatenated with the first and the second order derivatives, forming a feature vector of 39 elements.

■ 2.2 Acoustic Modeling

The previous section discussed feature representations for speech signals. This section will discuss acoustic modelling for speech recognition, in which statistical models are used to calculate the probability of a sequence of observed feature vectors. HMMs are the most widely used statistical model in the speech processing area. This section will discuss the basic concept of HMMs and their application to speech recognition.

■ 2.2.1 HMM

In HMM-based speech recognition, the observation vectors of a particular acoustic unit (e.g., a word or a phone) are assumed to be generated by a finite state machine. At each time instance, there is a hidden state. The hidden state can jump from the current state to other states according to certain probabilities. An observation vector is also generated at each time instance, according to a state-dependent output distribution.

As speech signals are sequences in time, left-to-right HMMs are often used to model speech signals. Figure 2-3 shows an example of such an HMM with 3 emitting states. Let $\mathbf{O} = \{o_1, ..., o_T\}$ be a sequence of observation vectors that is generated by this 3-state left-to-right HMM, in which o_t is the observation vector at time t and T is the length of the speech sequence. The generation process starts from the first, non-emitting state, i.e., state s_1. At each time, the state can jump to the next state or stay at the current state according to transition probabilities, a_{ij}. Here, a_{ij} denotes the probability of switching from state i

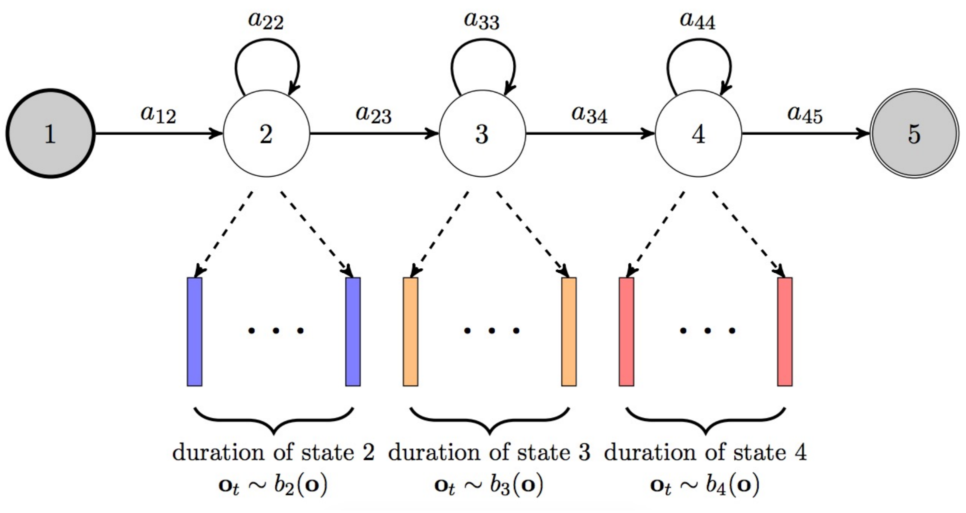

Figure 2-3: Block diagram of the a three-state HMM system.

to j. Once an emitting state j (e.g., states s_2 to s_4 in Figure 2-3) is reached, an observation vector is generated at the current time instant with a probability density $b_j(o)$. Note that the entry and exit states in Figure 2-3 are non-emitting. It is clear that for the observation vector sequence O with the length T, there is a state sequence $s = [s_1, \ ..., \ s_T]$ with the same length, where s_t is the state at time t. However, only O can be observed, while s is hidden and needs to be inferred from the observations. The sequence of observation vectors and the sequence of hidden states can be written together as O, s and will be referred to as the complete data set. The parameters of an N-state HMM include the state transition probability matrix A, and the state output probability distribution B. The transition probability a_{ij} can be arranged into a state transition probability matrix A, with its element at the j^{th} row and the i^{th} column defined as:

$$A_{ji} = a_{ij} = P(s_{t+1} = j | s_t = i).$$

Note that a_{ij} does not dependent on the time index t. To be a valid probability distribution, each column of this matrix must satisfy:

$$\sum_{j=1}^{N} P(s_{t+1} = j | s_t = i) = \sum_{j=1}^{N} a_{ij} = 1; \text{for } i = 1, \ ..., \ N.$$

Note that because of the use of the entry state, state s_1, a_{1j} specifies the initial state distributions of emitting states j, $j = 1, 2 \ldots N-1$. At each emitting state j, a state-dependent output probability, $b_j(o) = p(o|s = j)$ is used to govern the observation generation process, in which s is the current state. $b_j(o)$ can be a discrete distribution, which yields the so called discrete HMM (DHMM). Alternatively, $b_j(o)$ can be a probability density function. This yields the so called continuous density HMM (CDHMM). To use HMMs as acoustic models for speech recognition, there are a few practical considerations, e.g., the choice of acoustic units and state output probability distributions. These acoustic modelling techniques are briefly reviewed in the following paragraphs.

The choice of acoustic units includes word and sub-word units. For speech recognition tasks with a small recognition vocabulary (less than 1K words), e.g., in a digit recognition task, HMMs are often used to model individual words. However, for speech recognition tasks with a medium (1K-10K words) vocabulary to a large vocabulary ($>$ 10K words), it is not possible to collect sufficient training data for each word in the vocabulary. To solve this problem, HMMs are normally used to model sub-word units. Sub-word units are then composed to form word HMMs according to rules specified by a dictionary. As an example, the composition of two sub-word unit (a and b) HMMs to form a word (ab) HMM is illustrated in Figure 2-3. Note that in the figure, the non-emitting exit state of model a and the entry state of model b have been removed while the last emitting state of model a is connected to the first emitting state of model b. The entry state of model a and the exit state of model b become the new entry and exit state of the newly formed word model ab. The phone, which is a small speech segment that has distinct perceptual properties, is often chosen as the sub-word unit. The number of phones is normally significantly smaller than the number of words in a vocabulary. For example, the number of phones in English is between 40 to 60, while typical state-of-the-art speech recognition systems for English use a vocabulary which ranges from 20K to 64K words. Given a phone set, it is possible to build one HMM for each phone, regardless of its contexts. This is referred to as a monophone system. However, the factorization from a word HMM to context independent phone HMMs discards contextual information. Due to coarticulation, the pronunciation of a phone is influenced by the preceding and following phones, and thus varies with respect to its context [14]. To model the variations caused by context, context-dependent phone sets are normally used in large vocabulary speech recognition systems. A triphone is a context-dependent phone which uses the immediate left and right phone as the context for the phone in the centre position. For example, a phone *'b'* in the context *'ey b l'* is usually denoted as a triphone *ey-b+l*, where "-" denotes the left context and "+" denotes the right context. In this way, an isolated word "able" with silence "sil" at the start and the end of the word, can be mapped to a sequence of triphones as: *sil-ey+b* *ey-b+l* *b-l+sil*.

Using context-dependent triphone models significantly increases the number of acoustic units and thus requires a large amount of training data. Moreover, some triphones may not exist in the training set. To solve this problem, parameter tying techniques are usually used. The most widely used parameter tying technique is state clustering [15]. The basic idea of state clustering is to share the state output distributions between similar acoustic units. Initially, in the untied system, each state of each HMM has a unique output distribution. This gives 9 output distributions to be estimated. Clustering algorithms (e.g., [15,16]) can be used to cluster these 9 distributions into several groups. In this way, a lower number of state distributions would need to be estimated. Observation vectors belonging to the same group can be pooled together to estimate the parameters of one distribution. This ensures there are sufficient training data for each of the clustered state distributions.

There are generally two approaches that can be used for state clustering. One is a bottom-up approach, in which clusters are built in a bottom-up fashion. Initially, every un-tied distribution is a class. The most similar distributions are then merged and a new distribution is generated. This process can be repeated until a pre-defined number of classes is reached. The main problem with this approach is that it can not appropriately handle unseen contexts which do not appear in the training data. This problem can be addressed using the second, top-down, approach. Initially, all the states are grouped into a single root node. At each node, a phonetic question about the context of states is asked to split the states within the current node into left and right children nodes. For example, a question may ask whether the left context of the states in the current node is a nasal, i.e., in the set ng-*, n-*, m-*; the states with a positive answer will be grouped into the left childs node while the states with a negative answer will be grouped into the right childs node. Many phonetic questions can be asked at each node. The best question which maximizes the likelihood after splitting is selected. This process is repeated until the amount of training data associated with the current node falls below a threshold. Besides efficiency in clustering acoustic units, another advantage of using this top-down clustering is that it can easily handle unseen contexts. For example, when a new triphone is observed in test data, a series of phonetic questions can be asked to classify each state of this triphone into a leaf node and synthesis a new HMM for this triphone. The phonetic decision tree clustering method has been widely adopted in the most state-of-the-art ASR systems and is also used in this thesis.

■ 2.2.2 GMM-HMM System

A Gaussian Mixture Model (GMM) is a statistical generative model that can very effectively model the static cepstral features, while an HMM is a statistical model that is able to model the temporal dynamics of speech. Thus, the fusion of these two components creates a model capable of describing both spectral and temporal characteristics of the speech. The use of GMM-HMMs for speech modelling involves selecting an appropriate structure. This decision is usually done expertly and depends on the type of modeled speech units.

The typical GMM-HMM structure for sub-word units consists of 3 emitting states $\{s1,\ s2,\ s3\}$, the *entry* and *exit* states. The model is fully described by the transition matrix A = a_{ij}, which defines the probability of moving from state i to state j, and the emitting functions $b_i(o_t)$. The model usually lacks backward transitions as only forward and state-repeating transitions are allowed. Each state is assigned its emitting function $b_i(o_t)$ which estimates the probability of the observation vector o_t being generated by the state i and can be expressed as:

$$b_i(o_t) = \sum_{m=1}^{M} c_m \mathcal{N}(o_t; \mu_{im}, \sum_{im}) \tag{2.7}$$

The set of acoustic model parameters $\Theta_{AM} = \{c_m \ \mu_{im} \ \sum_{im}\}$ are the weight, the mean and the covariance matrix of a multivariate normal distribution $\mathcal{N}(o_t; \ \mu_{im}, \ \sum_{im})$. The probability of generating the state sequence $S = \{s_1, \ s_2, \ ..., \ s_k\}$ is dependent only on the transition probabilities and the observation probability for the frame at time t is dependent only on the emission probability $b_i(o_t)$ of the corresponding state i. The total likelihood of generating the observed sequence of acoustic features $O = [o_1, \ o_2, \ ..., \ o_T]$ is then expressed as

$$P(O|\Theta) = \sum_{s_1, \ ..., \ s_k} \prod_{t=1}^{T} a_{s_t|s_{t-1}} b_{s_t}(o_t) \tag{2.8}$$

where $a_{s_t|s_{t-1}}$ represents the state transition probability $p(s_t|s_{t-1})$. The described acoustic model is fully defined by the set of acoustic model parameters Θ that need to be inferred from the training data.

Several learning schemes already known in the machine learning field have been adopted for this purpose, while many others have been proposed specifically for ASR. Historically, the most common method for AM parameter estimation was based on Maximum Likelihood Estimation (MLE). However, this conventional approach has several drawbacks, some of which arise from the conditional independence assumptions of HMMs when used for modelling the human speech, and others from the assumptions of the MLE itself. This fact can cause the MLE to yield sub-optimal results in terms of classification accuracy. As a result, discriminative training algorithms have taken over as the principal training algorithms. Their main advantage is the fact that they dont make any assumptions about the distribution of training data.

■ 2.2.3 DNN-HMM hybrid system

A Deep Neural Network (DNN) is an M-layer network of coupled, stochastic binary units where M is usually greater than two. It contains a set of visible units, and layers of hidden units. For basic feed-forward DNNs, there are forward connections only between hidden units in adjacent layers, as well as between the visible and hidden units in the first hidden layer. That is, there are no within layer connections.

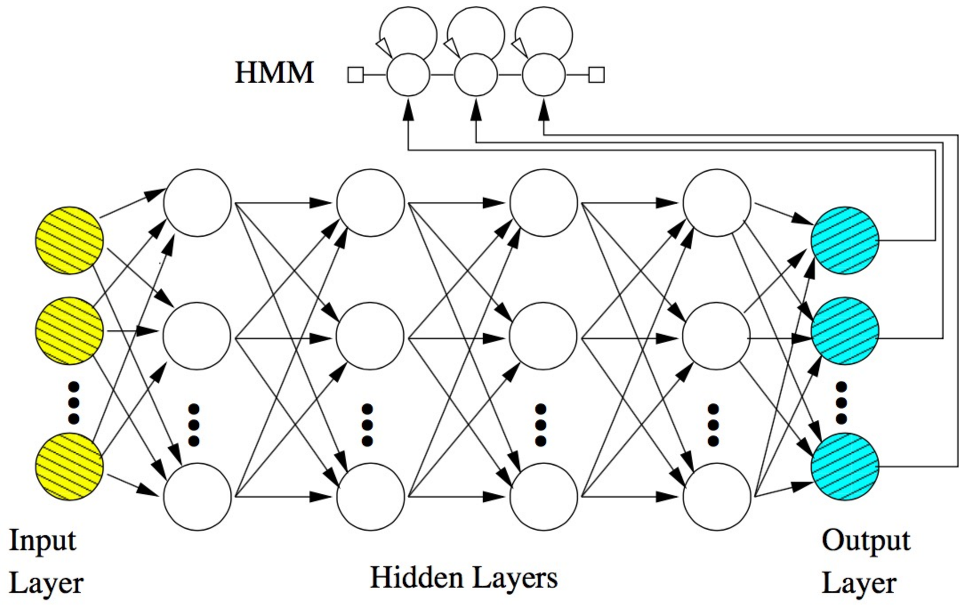

Figure 2-4: An example of a DNN-HMM hybrid model with 4 hidden layers

Figure 2-4 illustrates an example of a hybrid DNN-HMM system with a feed-forward architecture, where the temporal dynamics of speech are modelled by the HMM, and the DNN is used to model the observation probabilities within a static frame. The actual structure of the DNN in the figure is composed of an input, an output, and 4 hidden layers with a different number of units in each layer.

A unit j in each hidden layer employs a non-linear activation function to map the total sum of inputs from the preceding layer to the output that is sent to the next layer. The unit input x_j at the current layer is computed as a weighted linear combination as per Eq. 2.9.

$$x_j = b_j + \sum_i y_i w_{ij}, \quad y_j = \frac{1}{1+e^{-x_j}} \tag{2.9}$$

where b_j is the bias, y_i is the output from the unit i in the preceding layer and w_{ij} is the weight of a connection from unit i at the previous layer to unit j at the current layer. The most common activation functions include the logistic (sigmoid) function, hyperbolic tangent, and a rectified linear (ReLu) function. The recognition experiments in this thesis are conducted with a DNN-HMM system with a sigmoid activation function Eq. 2.9.

The output value of j^{th} unit represents the probability $P(j, o)$ that the observation vector o belongs to class j which can be done by using the softmax function according to Eq. 2.10.

$$P(i|o) = \frac{e^{-x_j}}{\sum_{j=1}^{C} e^{-x_j}} \tag{2.10}$$

Currently, the most popular DNN recognition frameworks are built to model HMM states of context-dependent phones (called senones). This approach, also called continuous density DNN-HMM, is very similar to previous state-of-the-art GMM-HMM systems which contributed a lot to their rapid development since many previously developed processes and methods were easily transferable to this newer framework. In the DNN-HMM, the output layer of the DNN is trained to estimate the conditional state posterior probabilities $p(s_t = i|o_t)$ given the observation o_t. The most popular DNN training method is to employ the error back-propagation algorithm in conjunction with a gradient descent optimization method. However, the original algorithm suffers from the problem of vanishing or exploding gradients, but that problem has been effectively solved by introducing improved deep learning algorithms. This section only touches on the problems of DNN training and does not elaborate on the details.

■ 2.3 Language Modeling, Decoding, and Measurement

■ 2.3.1 Language Modeling

The purpose of the Language Model (LM) is to estimate the probability of generating the hypothesized word sequence $P(W)$, which can be further decomposed using the chain rule as:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1}) \tag{2.11}$$

where $P(w_i|w_1, ..., w_{i1})$ is the probability that word w_i is spoken given the previously uttered word sequence $w_1, ..., w_{i1}$. The past word sequence is also called the history. The purpose of the language model is to provide the recognizer with an adequate estimate of $P(w_i|w_1,...,w_{i-1})$. However, it is practically unfeasible to create a model with a large history given all possible word sequences. As a consequence, the current state-of-the-art approach is to employ n-gram models which limit the history length down to $n-1$ number of words. The recognition experiments presented in this thesis were done using a tri-gram LM, which simplifies the formulated probability to the form of:

$$P(W) = P(w_1)\prod_{i=1}^{n} P(w_i|w_{i-1}) \tag{2.12}$$

One metric to evaluate language models is to compute the perplexity on heldout data, defined as:

$$\text{Perplexity} = 2^{-\frac{1}{n}log_2(P(W))}$$

where n is the number of words in the heldout data, and $P(W)$ is the probability of the data estimated by the model.

■ 2.3.2 Decoding

The final component of any speech recognizer is the decoding block, which combines the probability scores of the AM and LM, and outputs the most likely word sequence $\hat{W}$. Although it is computationally unfeasible to search the whole recognition space for the optimal solution, it can still be can be very effectively solved by utilizing the dynamic programming and Viterbi algorithm. Their application in speech recognition greatly simplifies the decoding process by utilizing the optimality principle which postulates that the optimal path through a directed graph is equivalent to taking optimal partial paths between the nodes. The optimality principle ensures that the likelihood for each state at each stage t can be computed by means of a simple recursion. Besides the described recursion, the Viterbi algorithm requires additional steps of recursion initialization, termination and path-backtracking. Another advantage of the algorithm is that it does not need to keep track of all partial paths leading to stage $t+1$. It is important to realize that the described procedure can be applied to both GMM-HMM as well as DNN-HMM architectures

■ 2.3.3 Error Measurement

Word Error Rate (WER) is the standard metric used to measure ASR performance. It can be considered as the word-level edit distance from the ground truth transcription. It is composed of three kinds of errors, substitution errors, insertion errors, and deletion errors.

$$\text{Substitution Error} = \frac{\text{\#of substitution errors}}{\text{\#of ground truth words}}$$

$$\text{Insertion Error} = \frac{\text{\#of insertion errors}}{\text{\#of ground truth words}}$$

$$\text{Deletion Error} = \frac{\text{\#of deletion errors}}{\text{\#of ground truth words}}$$

$$\text{WER = Substitution Error + Insertion Error + Deletion Error}$$

WER is often reported as a percentage. A perfect transcription will have 0% WER. Note that since we can have insertions, the WER can be higher than 100%. One caveat of the WER metric is that it weights all errors equally. A substitution of ”Sheet” with ”Cheat”, which sound very similar, counts the same as substituting ”Sheet” with ”Kittens.” In this sense, under certain conditions, such as high WER, an increase in WER does not necessarily mean a poorer system.

■ 2.4 ASR robustness

The current ASR systems have been able to achieve satisfactory results under clean acoustic conditions with high quality signals. However, when the recordings come from a noisy environment or the speech signal is distorted on its way from the talker to the microphone, the system performance can drop significantly.

■ 2.4.1 Noise and Reverberation

From the signal processing point of view, the distorted clean speech signal has two aspects: additive distortion and convolutional distortion.

Additive distortion is represented by background noise and competing speech from other speakers, if there is any. The electronic noise in recording equipment is usually negligible in additive distortion.

Convolutional distortion comes from the acoustic and electroacoustic components in the recording system. It is a joint outcome of the electroacoustic property of the microphone used for recording, and the acoustic properties of the physical environment where the recording takes place. The overall effect of convolutional distortion is usually approximated with a finite impulse response (FIR) filter $h(n)$, which is frequently referred to as the acoustic impulse response (AIR). When the electroacoustic distortion caused by recording equipment is negligible, the convolutional distortion is mainly determined by the room acoustics. Therefore the convolutional distortion is frequently referred to as the reverberation which could be approximated by another FIR filter. Such an FIR filter is usually referred to as the room impulse response (RIR), though it is not only dependent on the room acoustic properties but also on the installation of microphone, the speaker location in the room, and the direction in which the speaker is talking.

■ 2.4.2 Robustness Techniques

Many methods have been proposed to combat performance degradation in distant ASR systems. These can be classified into two approaches: a feature compensation approach and a model adaptation approach.

A feature compensation approach modifies the noisy features to make them more similar to the unobserved clean features during or after the feature extraction process. Such methods include speech enhancement techniques that were originally designed to enhance speech signal for human listening [17,18], feature compensation techniques that try to estimate clean features from noisy features [19, 20], feature normalization techniques that normalize both clean and noisy features to a new space where the noise distortion is reduced [21, 22], and temporal filters that reduce the dissimilarity of features in the modulation spectrum domain [23,24], etc.

In contrast, the model adaptation approach adapts the clean-trained acoustic model to better represent the noisy features. Examples of this approach include parallel model composition (PMC) [25], maximum a posteriori adaptation (MAP) [26], maximum likelihood linear regression adaptation (MLLR) [27], statistical re-estimation (STAR) technique [28] and joint compensation of additive and convolutive noises (JAC) [29].

In recent years, research on robust speech recognition has achieved progress in both the fields of front-end speech processing and acoustic models, mainly driven by multidisciplinary approaches combining ideas from signal processing and machine learning. Promising techniques can be merged, combining ideas from methods in noise robustness and novel ways for modeling noisy reverberant data.

Chapter 3
Multi-channel Speech Processing for Robust ASR

As discussed in the previous chapter, speech signals captured by a microphone located away from the user can be significantly corrupted by additive noise and reverberation. One method of reducing the signal distortion and improving the quality of the signal is to use multiple spatially-separated microphones rather than a single microphone. The technique of processing signals captured by these multiple microphones is referred to as array processing.

Array processing is a relatively mature field, developed initially to process narrow-band signals for radar and sonar applications, and then later applied to broadband signals such as speech. Microphone array processing methods have historically been designed according to principles pertinent to signal enhancement applications instead of speech recognition applications. These algorithms are usually concerned with generating the optimal output waveform and as such, they process the array signals according to various signal-level criteria, e.g. minimizing signal error or maximizing SNR. However, such criteria do not necessarily result in an improvement for speech recognition applications. As a result, traditional array processing schemes which are capable of producing high quality output signals may not result in significant improvements in speech recognition accuracy. Moreover, beamforming in microphone array processing systems need to know an angle to form beam pattern towards, and the beamforming result is very sensitive to this input angle. Therefore, accurate localization of the speaker is a critical step to the function of a multi-channel ASR system. The angle of arrival calculated using the phase difference between a pair of microphones at a particular time could deviate from the actual speaker angle by a lot, so advanced speaker localization techniques are in demand.

In this chapter[1] , we propose a robust speech recognition system that improves ASR performance using multi-channel information from microphone arrays. Figure 3-1 presents

[1]Portions of this work have been published in [30].

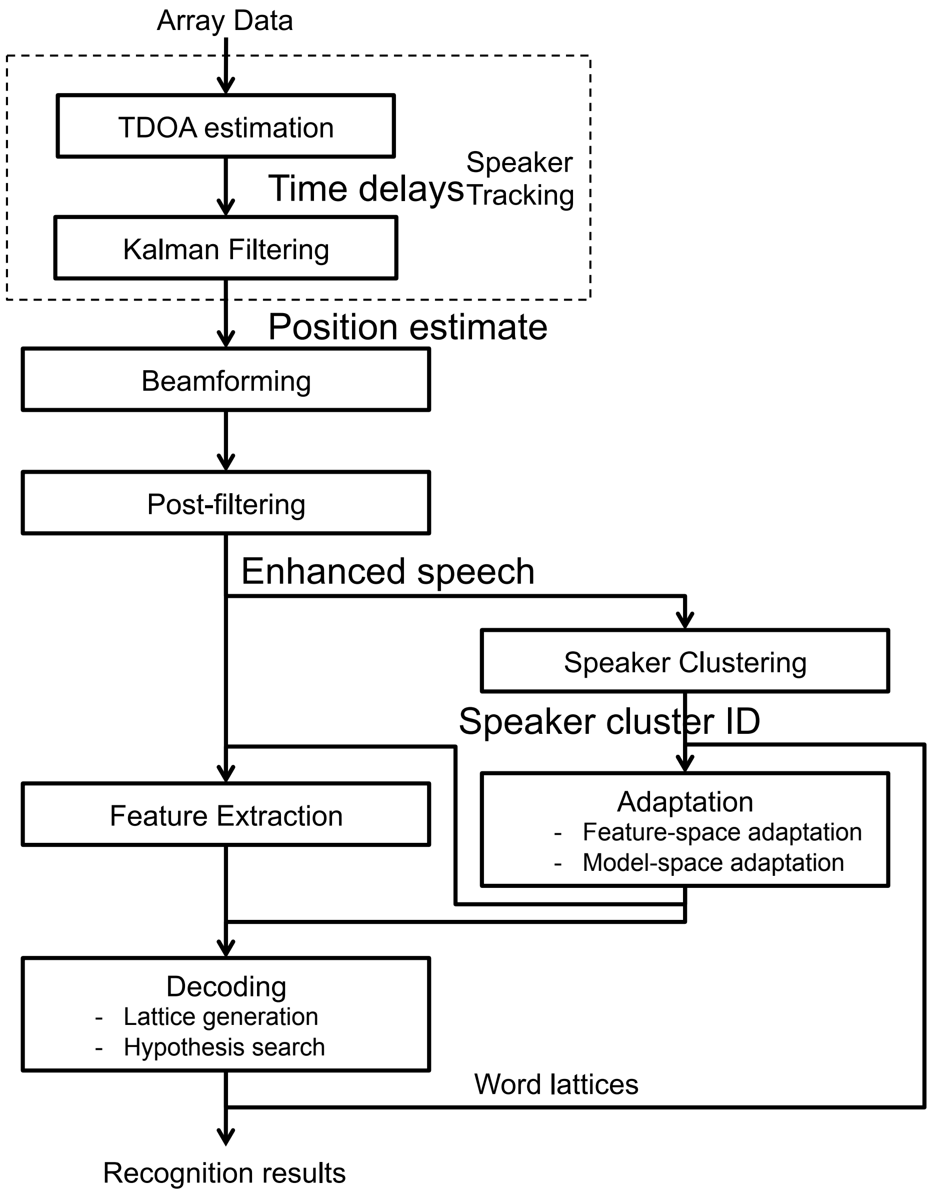

Figure 3-1: Block diagram of the multi-channel speech recognition system.

a schematic diagram of our overall system. The first key part of our system is a dynamic speaker tracking approach based on kalman filtering, which produces an accurate speaker position for spatial signal beamforming. This will be discussed in Section 3.2. The second key part of our system is beamforming, and it will be described in 3.3. We take up blind speaker clustering in Section 3.4.2, and in Section 3.4 we present the evaluation of the system on a publicly available, and well-known multi-channel corpus.

■ 3.1 Introduction

The fundamental idea behind array processing is illustrated with an example below. Let us consider two acoustic sources s_1 and s_2 located a distance r_1 and r_2, respectively, from a pair of microphones, m_1 and m_2. Source s_1 is of equal distance from both microphones, while source s_2 is closer to microphone m_2 than microphone m_1. This configuration is shown in Figure 3-2. We consider the output to be the sum of the signals received by the two microphones. Because the path lengths between source s_1 and the two microphones are equal, signals generated by s_1 will arrive in phase and their combination will amplify the signal by a factor of two. However, the path lengths between source s_2 and the two microphones are different. This difference is dependent on the angle of incidence of the source and the distance between the two microphones, as shown in Figure 3-2b. Because of the difference in path lengths, signals generated by source s_2 will arrive at the two microphones somewhat out of phase and thus combining them will cause signal s_2 to be attenuated.

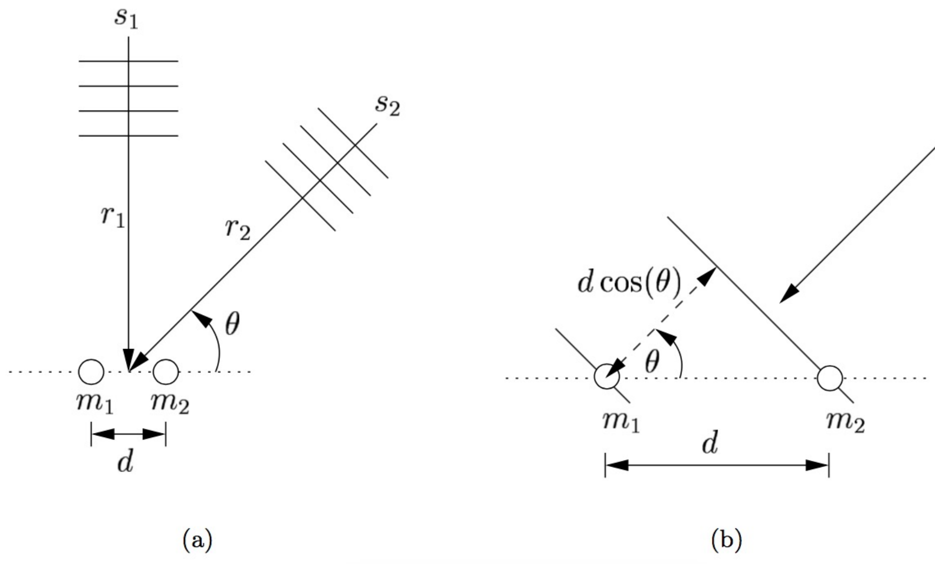

Figure 3-2: (a) Two acoustic sources propagating toward two microphones. The wavefront of source s_1 arrives at the two microphones simultaneously while the wavefront of source s_2 arrives at microphone m_2 prior to m_1. (b) The additional distance the source travels to m_1 can be determined from the angle of arrival θ and the distance between the microphones d.

We now expand this example to consider an arbitrary source $x[n]$ at some location (r, θ) from the center of an array of M microphones spaced linearly with an inter-element spacing d. If the source is assumed to be located in the far-field, the combined output of M microphones $y[n]$ can be expressed as

$$y[n] = \sum_{m=0}^{M-1} x[n - m\tau] \tag{3.1}$$

where the delay τ can be computed using the speed of sound v as

$$\tau = \frac{d\, cos(\theta)}{v} \tag{3.2}$$

. When we let $x[n] = \delta[n]$, we get the impulse response of the system $h[n] = y[n]$. The frequency response of $h[n]$ can be obtained by taking its discrete Fourier transform using 3.1:

$$H(\omega, \theta) = \sum_{m=0}^{M-1} e^{-2\pi\omega(m\tau)} = \sum_{m=0}^{M-1} e^{-2\pi\omega(m\frac{d\, cos(\theta)}{v})} \tag{3.3}$$

As Equation 3.3 indicates, the frequency response is dependent on the number of elements M, the microphone spacing d, the spectral frequency ω and the angle of incidence θ. The spatial response can be visualized by plotting the magnitude of $H(\omega, \theta)$ as a function of θ while d, M, and ω are held fixed. Such a representation, plotted in polar coordinates, is called a directivity pattern or beampattern.

By using an array of microphones rather than a single microphone, we are able to achieve spatial selectivity, reinforcing sources propagating from a particular direction, while attenuating sources propagating from other directions.

■ 3.2 Speaker Tracking

In this section, we present our speaker tracking technique, which has two components. First, time delays of arrival are estimated between pairs of microphones with a known geometry. Subsequently, a Kalman filter is used to combine these measurements and infer the position of the speaker.

■ 3.2.1 Time Delay of Arrival Estimation

Our speaker tracking system is based on estimation of *time delay of arrival* (TDOA) of the speech signal on the direct path from the speaker's mouth to unique pairs of microphones in the microphone array. TDOA estimation is performed with the well-known *phase transform* (PHAT) [31]

$$\rho_{mn}(\tau) \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{Y_m(e^{j\omega\tau}) Y_n^*(e^{j\omega\tau})}{\left|Y_m(e^{j\omega\tau}) Y_n^*(e^{j\omega\tau})\right|} e^{j\omega\tau} \, d\omega, \tag{3.4}$$

where $Y_n(e^{j\omega\tau})$ denotes the short-time Fourier transform of the signal arriving at the n^{th} sensor in the array [32]. The definition of the PHAT in Eq. 3.4 follows directly from the frequency domain calculation of the cross-correlation of two sequences. The normalization term $\left|Y_m(e^{j\omega\tau}) Y_n^*(e^{j\omega\tau})\right|$ in the denominator of the integrand is intended to weight all frequencies equally. It has been shown that such a weighting leads to more robust TDOA estimates in noisy and reverberant environments [33]. Once $\rho_{mn}(\tau)$ has been calculated, the TDOA estimate is obtained from

$$\tau_{mn} = \max_{\tau} \; \rho_{mn}(\tau). \tag{3.5}$$

■ 3.2.2 Kalman Filtering

Speaker tracking based on the maximum likelihood criterion [34] seeks to determine the speaker's position $\mathbf{x}$ by minimizing the error function

$$\epsilon(\mathbf{x}) = \sum_{s=0}^{S_2-1} \frac{\left[\hat{\tau}_s - T_s(\mathbf{x})\right]^2}{\sigma_s^2}, \tag{3.6}$$

where σ_s^2 denotes the error covariance associated with this observation, $\hat{\tau}_s$ is the observed TDOA as in Eq. 3.4 and 3.5, and $T_s(\mathbf{x})$ denotes the TDOAs predicted based on geometric considerations.

Although Eq. 3.6 implies that we should find $\mathbf{x}$ minimizing the instantaneous error criterion, we would be better advised to minimize such an error criterion over a series of time instants. In so doing, we exploit the fact that the speaker's position cannot change instantaneously; thus, both the present and past TDOA estimates are potentially useful in estimating a speaker's current position. Klee *et al.* [35] proposed a method to recursively minimize the least square error position estimation criterion in Eq. 3.6 with a variant of the *extended Kalman filter* (EKF). This was achieved by first associating the *state* $\mathbf{x}_k$ of the EKF with the speaker's position at time k, and the k^{th} observation with a vector of TDOAs. In keeping with the formalism of the EKF, Klee *et al.* [35] then postulated a *state* and *observation equation*,

$$\mathbf{x}_k = \mathbf{F}_{k|k-1}\mathbf{x}_{k-1} + \mathbf{u}_{k-1}, \text{and} \tag{3.7}$$

$$\mathbf{y}_k = \mathbf{H}_{k|k-1}(\mathbf{x}_k) + \mathbf{v}_k, \tag{3.8}$$

respectively, where $\mathbf{F}_{k|k-1}$ denotes the *transition matrix*, $\mathbf{u}_{k-1}$ denotes the *process noise*, $\mathbf{H}_{k|k-1}(\mathbf{x})$ denotes the vector-valued *observation function*, and $\mathbf{v}_k$ denotes the *observation noise*. The process $\mathbf{u}_k$ and observation $\mathbf{v}_k$ noises are unknown, but both have zero-mean Gaussian pdfs and known covariance matrices, $\mathbf{U}_k$ and $\mathbf{V}_k$, respectively. Associating $\mathbf{H}_{k|k-1}(\mathbf{x})$ with the TDOA function $T_s(\mathbf{x})$ with one component per microphone pair, it is straightforward to calculate the appropriate linearization about the current state estimate required by the EKF [11, §10.2],

$$\bar{\mathbf{H}}_k(\mathbf{x}) \triangleq \nabla_{\mathbf{x}}\mathbf{H}_{k|k-1}(\mathbf{x}). \tag{3.9}$$

By assumption $\mathbf{F}_{k|k-1}$ is known, and the *predicted state estimate* is given by $\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k|k-1}\hat{\mathbf{x}}_{k-1|k-1}$, where $\hat{\mathbf{x}}_{k-1|k-1}$ is the state estimate from the prior time step. The innovation is defined as

$$\mathbf{s}_k \triangleq \mathbf{y}_k - \mathbf{H}_{k|k-1}\left(\hat{\mathbf{x}}_{k|k-1}\right).$$

The new filtered state estimate is obtained from

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{G}_k\,\mathbf{s}_k, \tag{3.10}$$

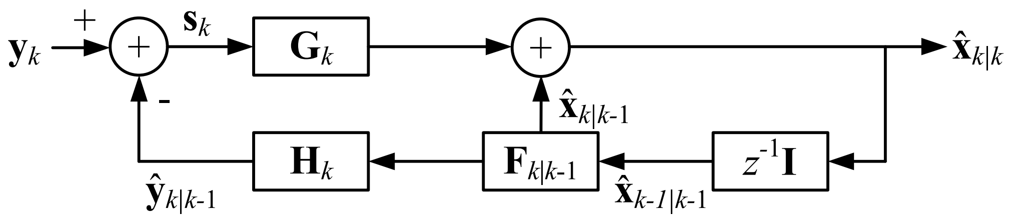

Figure 3-3: Predictor-corrector structure of the Kalman filter.

where $\mathbf{G}_k$ denotes the *Kalman gain*. A block diagram illustrating the prediction and correction steps in the state estimate update of a conventional Kalman filter is shown in Figure 3-3.

■ 3.3 Beamforming

Beamforming can then be used to mix multiple channels into one based on the speaker tracking results. The beamforming component of our system is based on the super-directive maximum negentropy (SDMN) beamformer [36, 37], which incorporates the super-Gaussianity of speech into adaptive beamforming. It has been demonstrated through ASR experiments on real array data that beamforming with the maximum negentropy (MN) criterion is more robust than conventional techniques against reverberation [38]. This is due to the fact that MN beamforming strengthens the target signal by using reflected speech; hence MN beamforming is not susceptible to signal cancellation. The post-filter used in our proposed system is a variant of the Wiener post-filter. One of the earliest and best-known proposals for estimating these quantities was by Zelinksi*et al.* [39], and a good survey of current techniques is given by Simmer *et al.* [40].

As shown in Figure 3-4, the SDMN beamformer has the generalized sidelobe canceller (GSC) architecture. The processing of SDMN beamforming can be divided into an upper branch and a lower branch. In the upper branch, the super-directive (SD) beamformer is used for the *quiescent vector* $\mathbf{w}_{\text{SD}}$. The process in the lower branch involves multiplication of the *block matrix bB* and *active weight vector* $\mathbf{w}a$. The beamformer's output for the array

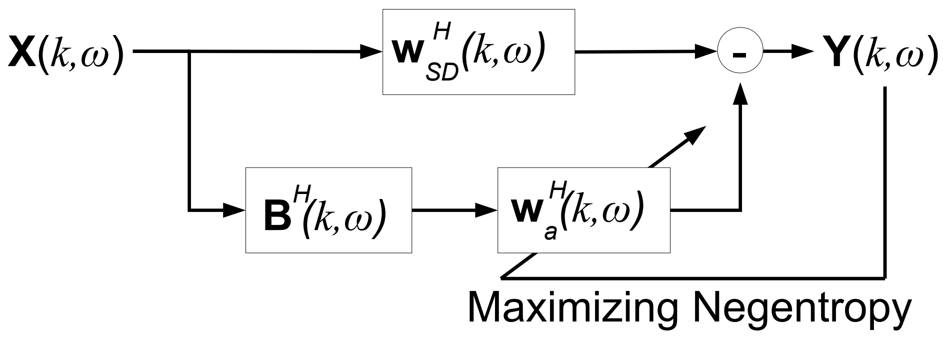
$\mathbf{X}(k,\omega)$
$\mathbf{w}_{SD}^{H}(k,\omega)$
-
$\mathbf{Y}(k,\omega)$
$\mathbf{B}^{H}(k,\omega)$
$\mathbf{w}_{a}^{H}(k,\omega)$
Maximizing Negentropy

Figure 3-4: Configuration of the super-directive maximum negentropy (SDMN) beamformer.

input vector $\mathbf{x}$ at frame k is obtained in the sub-band frequency domain as

$$Y(k,\omega) = (\mathbf{w}_{\text{SD}}(k,\omega) - \mathbf{B}(k,\omega)\mathbf{w}a(k,\omega))^H \, \mathbf{x}(k,\omega),$$

where ω is the angular frequency.

Let us define the *cross-correlation coefficient* between the inputs of the m-th and n-th sensors as

$$\rho_{mn}(\omega) \triangleq \frac{\mathcal{E}\{X_m(\omega)X_n^*(\omega)\}}{\sqrt{\mathcal{E}\{|X_m(\omega)|^2\}\,\mathcal{E}\{|X_n(\omega)|^2\}}}, \tag{3.11}$$

where $\mathcal{E}\{\cdot\}$ indicates the expectation operator. The super-directive design is then obtained by replacing the spatial spectral matrix [11, §13.4] with the *coherence matrix* $\Gamma_{\mathbf{N}}$ corresponding to a diffuse noise field. The m-th and n-th component of the latter can be expressed as

$$\Gamma_{\mathbf{N},m,n}(\omega) = \text{sinc}\left(\frac{\omega\, d_{m,n}}{c}\right) = \rho_{mn}(\omega), \tag{3.12}$$

where $d_{m,n}$ is the distance between the m-th and n-th elements of the array. Given the array manifold vector $\mathbf{d}$ computed with the position estimate, the weight of the SD beamformer can be expressed as

$$\mathbf{w}_{\text{SD}} = \frac{(\Gamma_{\mathbf{N}} + \sigma_{\text{d}}\mathbf{I})^{-1}\,\mathbf{d}}{\mathbf{d}^H\,(\Gamma_{\mathbf{N}} + \sigma_{\text{d}}\mathbf{I})^{-1}\,\mathbf{d}}, \tag{3.13}$$

where $\sigma_{\rm d}$ is an amount of diagonal loading and set to 0.01 for experiments. Notice that the frequency and time indicies ω and k are omitted here for the sake of simplicity. The SD beamformer has been proven to be more suitable than delay-and-sum (DS) and minimum variance distortionless response (MVDR) beamformers in meeting room conditions [38, 41, 42].

Once the SD beamformer is fixed in the upper branch, the blocking matrix is constructed to satisfy the orthogonal condition $\mathbf{B}^H \mathbf{w}_{\rm SD} = \mathbf{0}$. Such a blocking matrix can be, for example, obtained with the modified Gram-Schmidt [43]. This orthogonality implies that the distortionless constraint for the direction of interest will be maintained for any choice of the active weight vector. In contrast to normal practice, the SD-MN beamformer seeks the active weight vector that maximizes the negentropy of the beamformer's output. Assuming that the speech subband samples can be modeled with the generalized Gaussian distribution (GGD) with shape parameter f, we can express the beamformer's negentropy as

$$\begin{aligned} J(Y) &= \log(\pi\sigma_Y^2) + 1 \\ &\quad - \left[\log\{2\pi\Gamma(2/f)B_f^2\hat{\sigma}_Y/f\} + 2/f\right], \end{aligned} \tag{3.14}$$

where

$$\sigma_Y^2 = \mathcal{E}\{|Y|^2\},$$
$$\hat{\sigma}_Y = \frac{1}{B_f}\left(\frac{f}{2}\right)^{1/f} \mathcal{E}\left\{|Y|^f\right\}^{1/f},$$
$$B_f = \sqrt{\Gamma(2/f)/\Gamma(4/f)},$$

and $\Gamma(\cdot)$ is the gamma function.

■ 3.4 Evaluation

In this section, experiments are conducted to evaluate the performance of the proposed multi-channel ASR system. The experiment details and the results are described in the following sections.

■ 3.4.1 Data

The multi-channel data we used for this work was the Reverb Challenge corpus [44]. The Reverb Challenge corpus is a well known multi-channel speech corpus in the field of distant speech recognition. It consists of a training set, a development test set, and a test set. The development test set and the test set each consist of two different parts, namely simulated data (SimData) and real recordings (RealData). The 8-channel dataset assumes scenarios in which an utterance spoken by a single spatially stationary speaker in a reverberant room is captured with eight-channel circular microphone arrays (Fig. 3-5), all of which were provided as 8-channel recordings at a sampling frequency of 16 kHz. The datasets are available through the challenge webpage [45], and the specifications of the challenge data are detailed in [44].

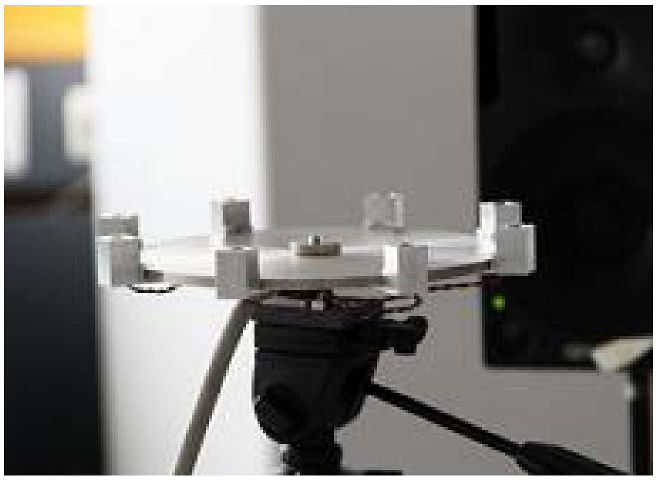

Figure 3-5: 8-channel microphone array used in the reverb challenge dataset

An overview of the datasets is given in Fig. 3-6. Details of each one are given in the following paragraphs.

SimData

SimData is comprised of reverberant utterances generated based on the WSJCAM0 corpus [46]. These utterances were artificially distorted by convolving clean WSJCAM0 signals with measured room impulse responses (RIRs) and subsequently adding measured stationary ambient noise signals with a signal-to-noise ratio (SNR) of 20 dB. SimData sim-

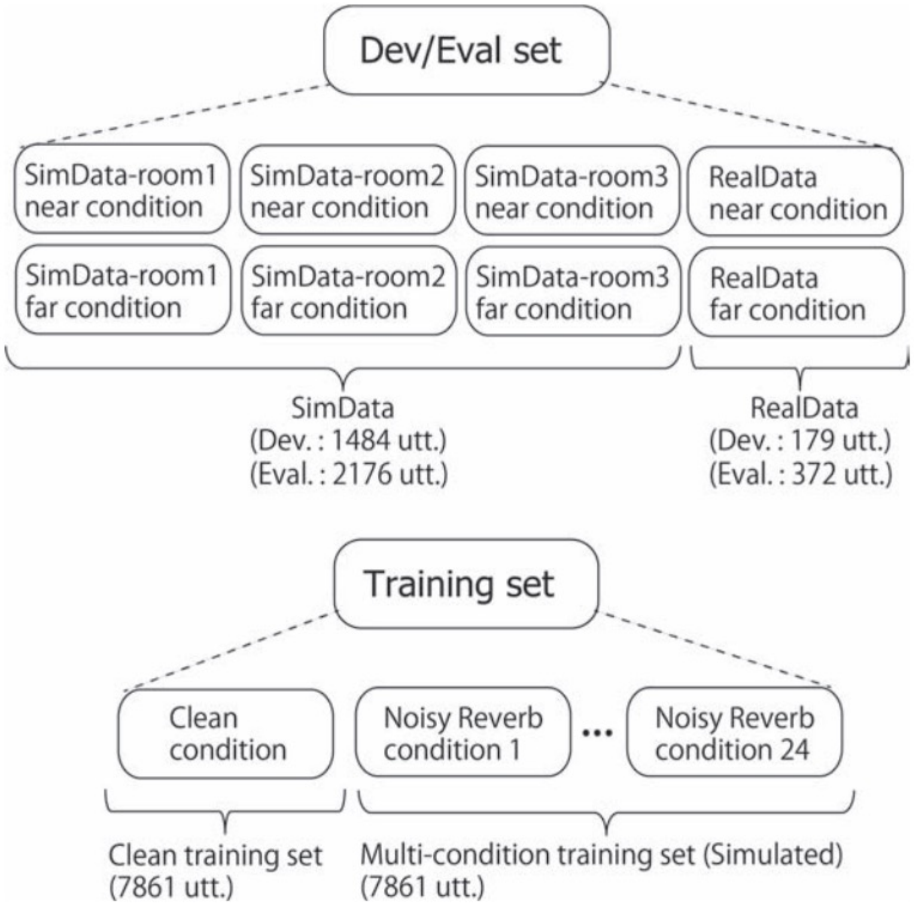
Dev/Eval set
SimData-room1 near condition
SimData-room2 near condition
SimData-room3 near condition
RealData near condition
SimData-room1 far condition
SimData-room2 far condition
SimData-room3 far condition
RealData far condition
SimData (Dev. : 1484 utt.) (Eval. : 2176 utt.)
RealData (Dev. : 179 utt.) (Eval. : 372 utt.)
Training set
Clean condition
Noisy Reverb condition 1
...
Noisy Reverb condition 24
Clean training set (7861 utt.)
Multi-condition training set (Simulated) (7861 utt.)

Figure 3-6: Overview of the datasets in the REVERB challenge. Average durations of utterances in training and test sets are about 7.5 and 6.9 s, respectively.

ulated six different reverberation conditions: three rooms with different volumes (small, medium, and large) and two distances between a speaker and a microphone array (near ∼50 cm and far ∼200 cm). Hereafter, the rooms are referred to as room1, room2, and room3. The reverberation times (i.e., $T60$) of room1, room2, and room3 are about 0.3, 0.6, and 0.7 s, respectively. The direct-to-reverberation ratios (i.e., $D50$) for room1 near and far, room2 near and far, and room3 near and far conditions are 99, 98, 95, 79, 97, and 81%, respectively. $D50$ refers to the percentage of the energy of the direct path plus early reflections up to 50 ms, relative to the total energy of the RIR. The RIRs and added noise were recorded in the corresponding reverberant room at the same position with the same microphone array, an 8-ch circular array with a diameter of 20 cm. The array is equipped with omni-directional microphones. The recorded noise was stationary diffuse background noise, which was mainly caused by the air conditioning systems in the rooms, and thus has relatively large energy at lower frequencies.

RealData

RealData, which is comprised of utterances from the MC-WSJ-AV corpus [42], consists of utterances spoken by human speakers in a noisy and reverberant room. Consequently the sound source cannot be regarded as completely spatially stationary due to the speakers head movements. The room used for the RealData recording is different from the rooms used for SimData. The rooms reverberation time was about 0.7 s [42]. The recordings contain some stationary ambient noise, which was mainly caused by the air conditioning systems. RealData contains two reverberation conditions: one room and two distances between the speaker and the microphone array (near $\sim$100 cm and far $\sim$250 cm). The recordings were measured with an array whose geometry is identical as that used for SimData. The text prompts of the utterances used in RealData and in part of SimData are the same. Therefore, we can use the same language and acoustic models for both SimData and RealData. For both SimData and RealData, we assumed that the speakers stay in the same room for each test condition. However, within each condition, the relative speaker's microphone position changes from utterance to utterance.

The training dataset consists of (i) a clean training set taken from the original WSJ-CAM0 training set and (ii) a multi-condition (MC) training set, which was generated from the clean WSJCAM0 training data by convolving the clean utterances with 24 measured room impulse responses and adding recorded background noise at an SNR of 20 dB. The reverberation times of the measured impulse responses for this dataset range roughly from 0.2s to 0.8s. Different recording rooms were used for the training, development, and evaluation sets.

■ 3.4.2 System Specifications

Speaker tracking

The primary free parameters in our speaker tracking component are $\mathbf{U}_k$ and $\mathbf{V}_k$, the known covariances matrices of the process and observation noises, $\mathbf{u}_k$ and $\mathbf{v}_k$, respectively. In our system, we set $\mathbf{U}_k = \sigma_u^2\mathbf{I}$ and $\mathbf{V}_k = \sigma_v^2\mathbf{I}$, and then tuned σ_u^2 and σ_v^2 to provide the lowest tracking error, which required a multi-channel speech corpus with ground truth speaker positions. This requirement was found in the IDIAP corpus collected by Lathoud *et al.* [47].

Shown in Figure 3-7 is a plot of radial tracking error in radians as a function of σ_u^2 and σ_v^2. This study led us to choose the final parameters of $\sigma_u^2 = 0.1$ and $\sigma_v^2 = 1 \times 10^{-8}$ for our experiment.

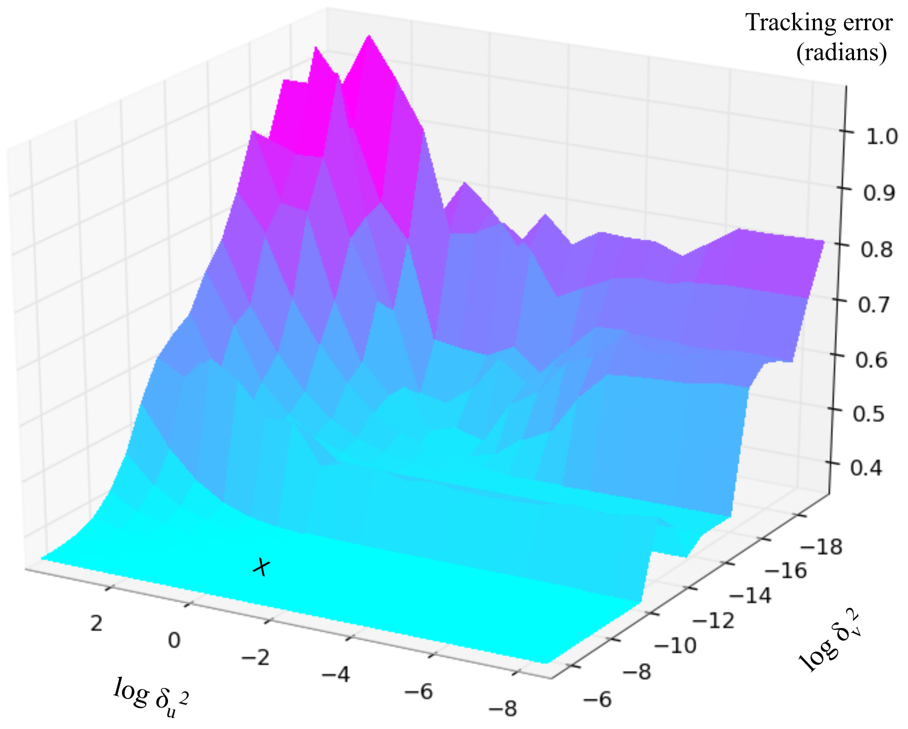

Figure 3-7: Speaker tracking error vs. process and observation noise parameters. The 'x' mark denotes our resulting choice of the parameter values.

Beamforming

For beamforming, the shape parameter of the GGD is trained with the clean WSJCAM0 data of the clean training set based on the maximum likelihood criterion as described in [37].

In order to avoid large weights, we apply the regularization term to the optimization criterion. The modified optimization criterion can be written as

$$\mathcal{J}(Y) = J(Y) - \alpha|\mathbf{w}_a|^2. \tag{3.15}$$

where α is set to 0.01 for the experiments.

Due to the absence of a closed-form solution with respect to **w**a, we have to resort to the gradient-based numerical optimization algorithm. Upon taking the partial deviation of 3.15 with respect to **w**a, we can obtain gradient information required for such a numerical optimization algorithm:

$$\frac{\partial \mathcal{J}(Y)}{\partial \mathbf{w}_a^*} = \mathcal{E}\left[\left\{\frac{1}{\sigma_Y^2} - \frac{f|Y|^{f-2}}{2\left(B_f\hat{\sigma}_Y\right)^f}\right\}\mathbf{B}^H\mathbf{x}Y^* - \alpha\mathbf{w}_a\right] \tag{3.16}$$

We use the Polak-Ribière conjugate gradient algorithm to find the solution. The post-filter used in our system is a variant of the Wiener post-filter. One of the earliest and best-known proposals for estimating these quantities was by [39].

As shown in the system diagram 3-1, after beamforming, the best output waveform possible is generated, which then gets treated as a single-channel input to the recognizer.

Feature extraction

The feature extraction of our ASR system is based on cepstral features estimated with a warped *minimum variance distortionless response* [48] (MVDR) spectral envelope of model order 30. Due to the properties of the warped MVDR, neither the Mel-filterbank nor any other filterbank was needed. The warped MVDR provides an increased resolution in low-frequency regions relative to the conventional Mel-filterbank. The MVDR also models spectral peaks more accurately than spectral valleys, which leads to improved robustness in the presence of noise. Front-end analysis involved extracting 20 cepstral coefficients per frame of speech and performing global cepstral mean subtraction (CMS) with variance normalization. The final features were obtained by concatenating 15 consecutive frames of cepstral features together, then performing Linear Discriminative Analysis (LDA) to obtain a feature of length 42.

Unsupervised Speaker Clustering

In this section, we present our approach for grouping single-speaker speech utterances into speaker-specific clusters. We start by computing supervectors, then applying factor analysis to generate i-vectors. We then train an LDA matrix based projection from the i-vectors to a speaker-discriminant subspace. The process is discussed in detail below.

A core feature of our approach lies in the approximation of speaker-conditional statistics, and training the LDA parameters for finding the optimal discriminative subspace. Figure 3-8 shows the block diagram of the speaker clustering system.

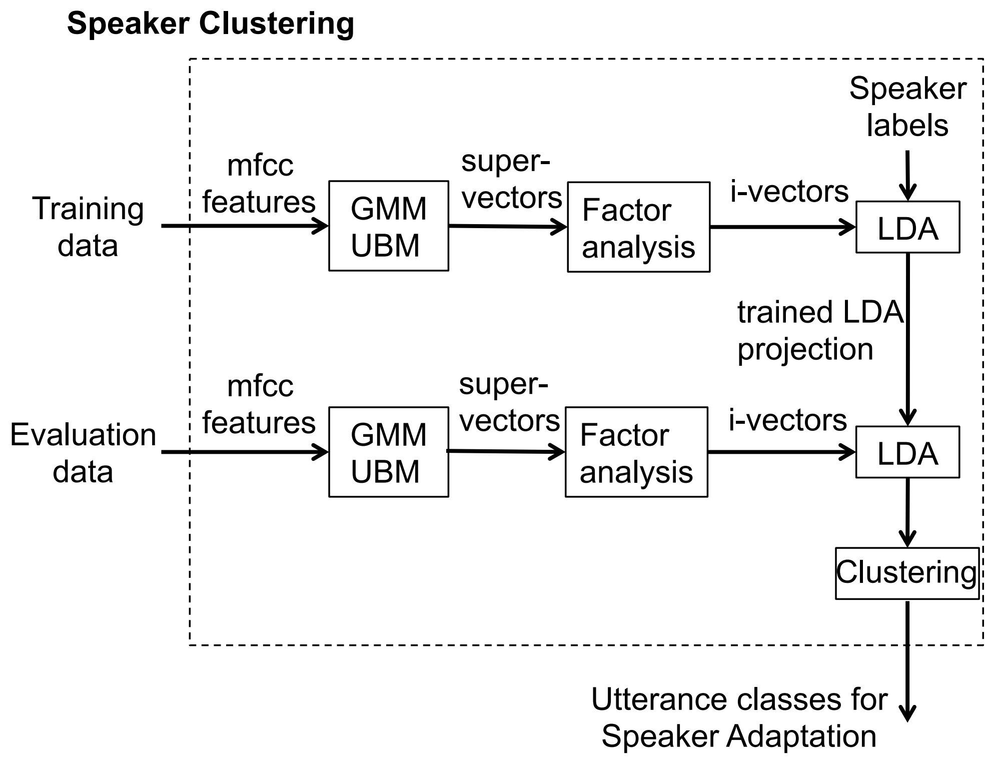

Figure 3-8: Block diagram of the speaker clustering algorithm.

For each utterance, a GMM with 512 mixtures is adapted, given appropriate front-end features (39-dimensional MFCC features) [49]. We denote the GMM mean components, which are speaker-dependent, as supervectors **M**. The Universal Background Model (UBM) [49] is a large GMM trained over all utterances to represent the speaker-independent distribution of features. We denote the UBM mean components, which are speaker-independent, as UBM vector **m**.

Speaker clusters are generated by applying the K-means clustering algorithm to this discriminant subspace, where the Euclidean distance reflects inter-speaker differences.

According to Total Variability Factor Analysis [50], given an utterance, the supervector **M** can be rewritten as follows:

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{w} \tag{3.17}$$

The key assumption in factor analysis is that the GMM supervector of the speaker- and channel-dependent **M** for a given utterance can be broken down into the sum of two supervectors where supervector **m** is the speaker- and session-independent supervector taken from a UBM, **T** is a rectangular matrix of low rank that defines the variability space and **w** is a low-dimensional (90-dimensional in our system) random vector with a normally distributed prior $\mathcal{N}(0, 1)$. These new vectors **w** are refered to as identity vectors or i-vectors for short [50].

The i-vectors **w** obtained from factor analysis contain both speaker and channel dependent information. To extract the speaker-discriminant subspace, LDA is applied to map the i-vectors to a 10-dimensional subspace. We trained our LDA projection on the simulated training data and applied the projection matrix on the evaluation set to perform unsupervised dimensionality reduction.

After LDA, the K-means algorithm is next applied on the subspace vectors, for finding the speaker clusters. K-means is a widely used clustering algorithm formulated based on a formal objective function [51]. Namely, given a set of N observation samples in $mathbbR^D$, and the number of clusters K, the objective is to determine a set of K points in $\mathbb{R}^D$, the means, so as to minimize the mean squared distance from each data point to its nearest mean. For SimData and RealData experiments, we used 40 and 20 clusters respectively.

Acoustic Model

Our system is based on two acoustic models. The first model was trained on the clean WSJCAM0 [46] and WSJ0 corpora. Training consisted of conventional HMM training, with three passes of forward-backward training followed by Gaussian splitting and more training [52]; this was followed by speaker-adapted training (SAT) [11, §8.1.3].

To train the second acoustic model, we first took the WSJ0 and WSJCAM0 corpora and "dirtied" through convolution with the multi-channel room impulse responses and addition of the multi-channel noise provided with the Reverb Challenge data. These dirty multi-channel streams were then used for speaker tracking followed by beamforming. Once we had produced the final processed single stream of data, they were once more used first for conventional HMM training and then for speaker-adapted training.

For the first pass of the system, we trained the acoustic model with noisy speech processed with SD beamforming, described in Section 3.3. For the adapted passes, we used acoustic models trained based on clean WSJ0 and WSJCAM0 corpora as described in Section 3.4.2. Our final system employed the noisy acoustic model in the first pass and then switched to the clean acoustic model in the adapted passes. Additionally, our system used the feature-space noise adaptation method based on factorized transforms [53]. Feature-space noise adaptation was performed by cascading two linear transforms: (1) noise-adapted constrained maximum likelihood linear regression (CMLLR) transform [54] estimated with all the speakers globally and (2) speaker-dependent CMLLR transform computed with speech from each speaker or cluster.

Recognition and Adaptation Passes

We performed four decoding passes on the waveforms obtained from beamforming. Each pass of decoding used a different acoustic model or speaker adaptation scheme. For all passes save the first unadapted pass, speaker adaptation parameters were estimated using the word lattices generated during the prior pass, as in Uebel *et al.* [55].

A description of the four decoding passes follows: 1) Decode with the unadapted, conventional ML acoustic model. 2) Estimate vocal tract length normalization (VTLN) [56] parameters and CMLLR for each speaker, then re-decode with the conventional ML acoustic model. 3) Estimate VTLN, CMLLR, and maximum likelihood linear regression (MLLR) [57] parameters for each speaker, then redecode with the conventional model. 4) Estimate VTLN, CMLLR, MLLR parameters for each speaker, then redecode with the ML-SAT model.

All passes used the full tri-gram LM for the 5,000 word WSJ task, which was made possible through the fast-on-the-fly composition algorithm described in [58].

■ 3.5 Results and Conclusions

	SimData							RealData		
	Room1		Room2		Room3		Ave.	Room1		Ave.
Baseline System	Near	Far	Near	Far	Near	Far		Near	Far	
1 (clean)	18.1	28.4	43.0	82.2	53.5	88.0	52.2	89.7	87.3	88.5
2 (clean)+CMLLR	14.8	18.9	24.7	64.6	33.8	78.5	39.2	82.3	80.8	81.5
3 (multi)	20.6	21.2	23.7	38.7	28.1	44.9	29.5	58.5	55.4	57.0
4 (multi)+CMLLR	16.2	18.7	20.5	32.5	24.8	38.9	25.3	50.1	47.6	48.9

Table 3.1: Baseline WER (%) on the Reverb dataset

Table 3.1 presents the baseline system performance. The Baseline systems are provided by the Reverb Challenge [59] [45] which are triphone GMM-HMM recognizers based on HTK. Baseline 1 is trained on clean training data without CMLLR adaptation. Baseline 2 is trained on clean training data with CMLLR adaptation. Baseline 3 is trained on multi-condition training data without CMLLR adaptation, and Baseline 4 is trained on multi-condition training data with CMLLR adaptation. For the rest of this chapter, we use the best performing baseline - Baseline 4 - as our baseline model.

WER of the Proposed System

Table 3.2 presents the word error rates (WERs) obtained with our systems on the reverb datasets and compares it with the baseline result.

	SimData							RealData		
	Room1		Room2		Room3		Ave.	Room1		Ave.
System	Near	Far	Near	Far	Near	Far		Near	Far	
Baseline	16.2	18.7	20.5	32.5	24.8	38.9	25.3	50.1	47.6	48.9
Proposed	8.44	8.91	9.99	13.19	10.77	20.17	11.91	16.74	19.51	18.13

Table 3.2: WER (%) of the proposed system on the Reverb dataset

We can see that our multi-channel system reduced the relative WER by 48%, 52%, 51%, 59%, 56.7% and 48% respectively for the six scenarios in SimData. Our system reduced the WER by 66.6% and 59% respectively for the two scenarios in RealData. Although we use a baseline that has already achieved high recognition performance with unprocessed distant speech, we obtained a large additional improvement using the proposed multi-channel front-end (up to 66.6% relative WER reduction). With the proposed system, ranked second among the 49 ASR systems submitted to the RealData component of the 2014 Reverb Challenge [60].

The Effect of Array Processing

To better understand the gains brought by our system, we provide further comparisons and analysis in the next paragraphs.

Table 3.3 demonstrates the effectiveness of the array processing component in our system. The results obtained with a single channel is provided as a contrast to our system's results. Both systems used the same acoustic, language model, and adaptation methods. We can see that the array processing component - speaker tracking and beamforming - significantly improved the ASR performance when going from a single-channel to an eight-channel setup.

We also built a system with conventional SD beamforming, to facilitate comparisons with the SDMN beamforming used in our proposed system. The results in Table 3.4 suggests that the beamforming method with the maximum negentropy criterion is more robust against reverberation. This is due to the fact that MN beamforming enhances the target signal by manipulating its weights so as to delay and account for reflections [38].

	SimData							RealData		
	Room1		Room2		Room3		Ave.	Room1		Ave.
System	Near	Far	Near	Far	Near	Far		Near	Far	
Proposed	8.44	8.91	9.99	13.19	10.77	20.17	11.91	16.74	19.51	18.13
Contrast 1	14.8	16.27	17.7	31.54	20.11	40.65	23.51	43.28	44.47	43.88

Table 3.3: The effect of the array processing component: comparison between the proposed system with and without front-end array processing. Contrast System is a single channel (SC) system.

	SimData							RealData		
	Room1		Room2		Room3		Ave.	Room1		Ave.
System	Near	Far	Near	Far	Near	Far		Near	Far	
Proposed	8.44	8.91	9.99	13.19	10.77	20.17	11.91	16.74	19.51	18.13
Contrast 2	8.73	9.49	10.69	15.39	15.72	29.54	14.92	18.75	21.03	19.89

Table 3.4: The effect of beamforming method: proposed system uses SDMN beamforming and contrast system uses conventional SD beamforming.

The Effect of Speaker Adaptation

Table 3.5 shows a WERs obtained without the application of our speaker adaptation method. The proposed system used speaker labels produced by the K-means clustering algorithm on i-vectors. For reference, we also show WERs obtained using the true speaker identities. We observed that adaptation using the speaker labels created by the proposed method reduced WERs by 5.3% absolute and 1.6% absolute, for the SimData and RealData, respectively.

	Simulated Data							Real Data		
	Room 1		Room 2		Room 3			Room 1		
System	Near	Far	Near	Far	Near	Far	Ave.	Near	Far	Ave.
Proposed	8.44	8.91	9.99	13.19	10.77	20.17	11.91	15.97	18.67	17.33
Contrast 3	13.23	15.12	14.77	20.97	14.92	24.65	17.27	17.45	20.42	18.94
Oracle	7.74	8.68	9 .33	12.81	9.54	19.74	11.31	13.41	15.06	14.50

Table 3.5: Comparison of the ASR performance with and without speaker adaptation. Oracle system uses the true speaker labels for adaptation.

It is also seen from Table 3.5 that the use of the true speaker labels yielded a reduction in error rate of approximately 1.0% absolute for the simulated data; the reduction was larger, approximately 4.5% absolute for the real data. This difference in behavior is ascribed to the fact that the simulated WSJCAM0 training data, which was used to estimate the LDA transformation on the i-vectors prior to *K*-means clustering, matched the simulated evaluation set much better than the real evaluation set. Hence, the separation of speaker classes was better for the simulated data than for the real data.

Chapter 4
Deep Learning based Front End Denoising: Deep Denoising Autoencoder

In the previous chapter, we presented a robust speech recognition system that improves system performance by utilizing multi-channel information. From this chapter on, we focus mainly on single channel systems and demonstrate how to boost system robustness in this scenario.

The major problem in distant-talking speech recognition, as discussed in the previous chapters, is the corruption of speech signals by both interfering sounds and reverberation caused by the far speaker-to-microphone distance. Since the beginnings of speech recognition research, a range of techniques have been developed to combat additive and convolutional noise. These methods focused on generating robust features based on traditional signal processing techniques [23, 61–64].

Along with the growing popularity of DNN-HMMs for ASR, are the use of DNN based methods to generate robust speech features. For example, Sainath *et al.* explored deep bottleneck autoencoders to produce features for GMM-HMM based ASR systems [65]. Vinyals *et al.* investigated the effectiveness of DNNs for detecting articulatory features, which combined with MFCC features were used for ASR tasks [66].

In this chapter, we describe an alternative front-end method for robust feature generation via deep denoising autoencoders (DDAs)[1]. The proposed DDA framework involved layers of affine+sigmoid encoding followed by affine decoding to recover speech features from noisy reverberant speech features. The DDA was fine-tuned with clean features, which resulted in learning a stochastic mapping from noisy to clean features.

Denoising autoencoders (DAs) have been explored by Vincent *et al.* and Bengio [68, 69], and were stacked to form stacked denoising autoencoder (SDA) to generate robust

[1]Portions of this work have been published in [67].

features for images [70]. Meanwhile, Ishii *et al.* applied DAs to reconstruct clean speech spectra from reverberant speech [59].

Our work is differentiated by a) denoising at the feature level, b) mapping multiple frames of noisy features directly to a single frame output of features, and c) a training procedure that utilized decoder layers that were asymmetric with respect to the encoder layers.

■ 4.1 Background

A distorted speech signal has two aspects: additive distortion and the convolutional distortion. We denote the clean speech signal as $x[n]$ where n is the discrete sampling index, then the distant speech recording signal $y[n]$ can be written as:

$$y[n] = x[n]h[n] + v[n] \tag{4.1}$$

where $h[n]$ is the convolutional distortion (commonly referred to as reverberation), and $v[n]$ is the additive distortion (referred to as noise).

■ 4.1.1 Reverberation

For a speech recognition system in a reverberant space, where speech is captured with one or more distant microphones, the wavefront of the speech signal is repeatedly reflected off the walls and other objects in the space. These reflections, perceived as reverberation, alter the acoustic characteristics of the original speech. The speech signals captured by the microphones are fed into the recognizer, whose processing steps are usually grouped into two broad units: the front end and the back end. The goal of reverberant speech recognition is to correctly transcribe speech corrupted by reverberation regardless of the severity of the reverberation. The question that immediately arises is: "Why do we need fundamentally new techniques for handling reverberation? To answer this question, this section describes the fundamental problem in reverberant speech recognition and discusses the properties of reverberation that can be leveraged to overcome the problem.

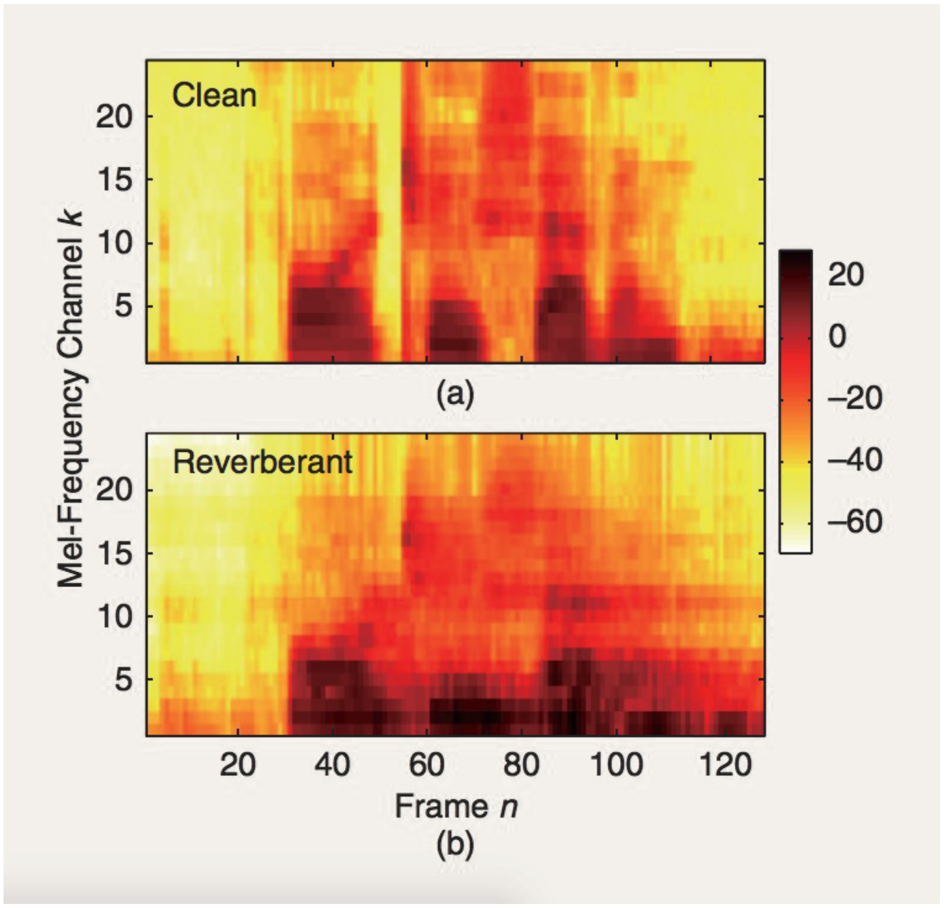

Figure 4-1: Log Mel-frequency filterbank features corresponding to the utterance ”four, two, seven” in dB color scale, extracted from (a) clean and (b) reverberant speech.

The effect of reverberation on feature vector sequences is illustrated in Figure 4.1.1, which compares clean and reverberant log Mel-frequency filterbank features extracted from an utterance ”four, two, seven”. Each log Mel-frequency filterbank feature is a frequency-warped and dimension-reduced version of each spectral bin obtained by short-time Fourier transform (STFT). As can be seen from the figure, energies from past frames get carried over and attenuates to current and later frames, causing aliasing in the spectra.

The room impulse response can be divided into three portions as shown in Figure 4.1.1. After the arrival of the direct sound, several strong reflections, called early reflections, occur within 50 ms. After that comes a series of numerous indistinguishable reflections, called late reverberations. The characteristics of the early reflections depend strongly on the positions of the speaker and microphone. By contrast, the magnitude

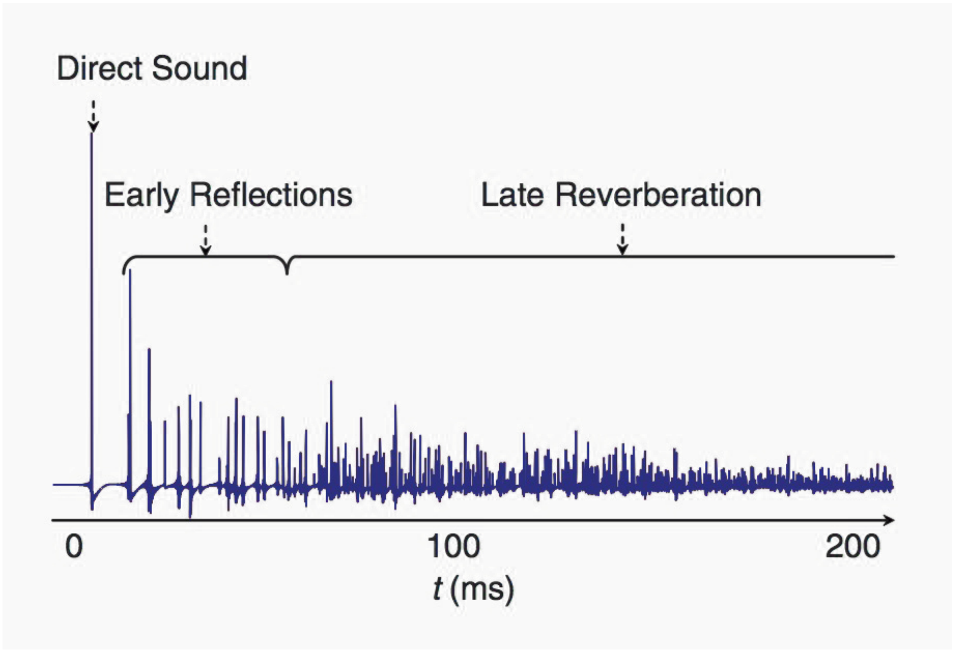

Figure 4-2: Room impulse response generated by the image method [2]. Room impulse responses actually measured in rooms are usually noisier due to microphone characteristics, band limitation with analog-to-digital conversion, and measurement errors.

of the late reverberation decays approximately exponentially and the decay rate is independent of the positions. The time required for the late reverberation to decay by 60 dB relative to the level of direct sound is called the reverberation time T_{60} . For typical office and home environments, the reverberation time ranges from 200 ms to 1,000 ms.

■ 4.2 Autoencoder

In this section, we begin with the simple notion of encoder-decoder models that retain information and progress to formally introduce the traditional autoencoder paradigm from this more general vantage point.

■ 4.2.1 Introduction

We are interested in learning a (possibly stochastic) mapping from input X to a novel representation Y. To make this more precise, let us restrict ourselves to parameterized mappings $q(Y|X) = q(Y|X;\theta)$ with parameters θ representing what we want to learn.

First we give an operational definition of a “good” representation as one that will be useful for addressing tasks of interest, in the sense that it will improve system performance compared to a baseline. Based on the objective measure typically used to assess algorithm performance, this might be phrased as ”A good representation is one that will yield a better performing classifier”. Final classification performance will typically be used to objectively compare algorithms. However, if a lesson is to be learned from the recent breakthroughs in deep learning training techniques, it is that the error signal from a single narrowly defined classification task should not be the only, nor primary criterion used to guide the learning of representations. For example, it has been shown experimentally that beginning by optimizing with an unsupervised criterion, oblivious of the specific classification problem, can greatly help in eventually achieving superior performance for that classification problem . It can also be argued that the capacity for humans to quickly become proficient in new tasks builds on much of what they have learned prior to confronting the new task.

One natural criterion that we may expect any good representation to meet, at least to some degree, is to retain a significant amount of information about the input. It can be expressed in information-theoretic terms as maximizing the mutual information $I(X;Y)$ between an input random variable X and its higher level representation Y. This is the infomax principle put forward by Linsker (1989) [71].

Mutual information can be decomposed into an entropy and a conditional entropy term in two different ways. A first possible decomposition is $I(X;Y) = H(Y)H(Y|X)$ which lead Bell and Sejnowski (1995) to their infomax approach to Independent Component Analysis [72]. Here we will start from another decomposition: $I(X;Y) = H(X)H(X|Y)$. Since observed input X comes from an unknown distribution q(X) on which θ has no influence, this makes $H(X)$ an unknown constant. Thus the infomax principle reduces to:

$$\begin{aligned} \underset{\theta}{\operatorname{argmax}}\, \mathbf{I}(X;Y) &= \underset{\theta}{\operatorname{argmax}} -\mathbf{H}(X|Y) \\ &= \underset{\theta}{\operatorname{argmax}}\, \mathbf{E}_{q(X,Y)}[\log q(X|Y)] \end{aligned} \tag{4.2}$$

Now for any distribution $p(X|Y)$ we will have

$$\mathbf{E}_{q(X,Y)}[\log p(X|Y)] \leq \mathbf{E}_{q(X,Y)}[\log q(X|Y)] \tag{4.3}$$

as can be shown starting from the property that for any two distributions p and q we have Kullback-Leibler (KL) divergence $D_{KL}(q||p) \geq 0$, and in particular $D_{KL}(q(X|Y = y)||p(X|Y = y)) \geq 0$.

Let us consider a parametric distribution $p(X|Y;\theta')$, parameterized by θ', and the following optimization:

$$\max_{\theta,\theta'} \mathbf{E}_{q(X,Y;\theta)}[\log p(X|Y;\theta')]. \tag{4.4}$$

From Equation 4.3, we see that this corresponds to maximizing a lower bound on $H(X|Y)$ and thus on the mutual information. We would end up maximizing the exact mutual information provided that there exists a θ' such that $q(X|Y) = p(X|Y;\theta')$.

If, as is done in infomax ICA, we further restrict ourselves to a deterministic mapping from X to Y, that is, representation Y is to be computed by a parameterized function $Y = f_\theta(X)$ or equivalently $q(Y|X;\theta) = \delta(Y - f_\theta(X))$ (where δ denotes Dirac-delta), then this optimization can be written:

$$\max_{\theta,\theta'} E_{q(X)}[\log p(X|Y = f_{\theta(X)};\theta')]. \tag{4.5}$$

This again corresponds to maximizing a lower bound on the mutual information. Since $q(X)$ is unknown, but we have samples from it, the empirical average over the training samples can be used instead as an unbiased estimate (i.e., replacing $E_{q(X)}$ by $E_{q^0(X)}$

$$\max_{\theta,\theta'} E_{q^0(X)}[\log p(X|Y = f_{\theta(X)};\theta')]. \tag{4.6}$$

We will see in the next section that this equation corresponds to the reconstruction error criterion used to train autoencoders.

■ 4.2.2 Traditional Autoencoder

Autoencoders consist of an encoder and a decoder. The encoder is the deterministic mapping f_θ that transforms an n-dimensional input vector $\mathbf{x}$ into a hidden representation $\mathbf{y}$.

For traditional autoencoders, the typical form is an affine mapping followed by a non-linearity:

$$f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

with parameter set $\theta = \{\mathbf{W}, \mathbf{b}\}$, where $\mathbf{W}$ is a $d \times d$ weight matrix and b is an offset vector of dimensionality d. The resulting hidden representation $\mathbf{y}$ is then mapped back to a reconstructed d-dimensional vector $\mathbf{z}$ in input space, with $\mathbf{z} = g'_\theta(\mathbf{y})$.This mapping is called the decoder. Its typical form is an affine mapping optionally followed by a squashing non-linearity, that is, either

$$g_{\theta'}(\mathbf{y}) = \mathbf{W}'\mathbf{y} + \mathbf{b}',$$

or

$$g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}').$$

with appropriately sized parameters $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. In general, $\mathbf{z}$ is not to be interpreted as an exact reconstruction of $\mathbf{x}$, but rather in probabilistic terms as the parameters (typically the mean) of a distribution $p(X|Z = \mathbf{z})$ that may generate $\mathbf{x}$ with high probability. This yields an associated reconstruction error to be optimized with respect to loss $L(\mathbf{x}, \mathbf{z}) = -\log p(\mathbf{x}|\mathbf{z})$. For real-valued $\mathbf{x}$, this requires $X|\mathbf{z} \sim \mathcal{N}(\mathbf{z}, \sigma^2\mathbf{I})$, which yields $L(\mathbf{x}, \mathbf{z}) = C(\sigma^2)||\mathbf{x} - \mathbf{z}||^2$, where $C(\sigma^2)$ denotes a constant that depends only on σ^2 and thus can be ignored for the optimization. This is the squared error objective found in most traditional autoencoders. In this setting, due to the Gaussian interpretation, it is more natural not to use a squashing nonlinearity in the decoder. For the rest of this work, we used an **affine+sigmoid encoder** and an **affine decoder** with **squared error loss**.

■ 4.2.3 Denoising Autoencoder

The denoising autoencoder (DA) is a variant of the basic autoencoder. Figure 4-3 shows a standard structure of a denoising autoencoder.

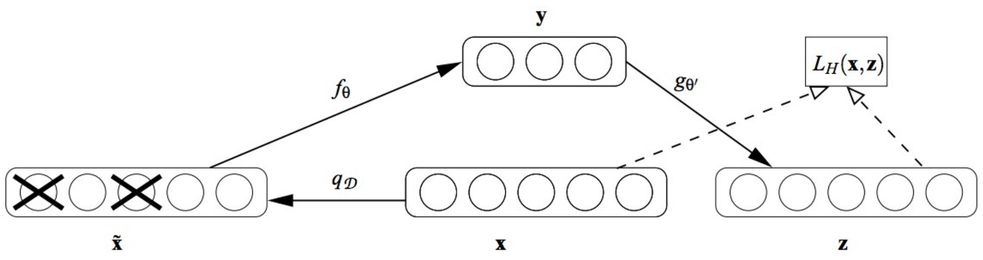

Figure 4-3: Denoising Autoencoder. $\hat{x}$ is a corrupted version of x, f_θ encodes $\hat{x}$ to y while $g_{\theta'}$ decodes y to z. The reconstruction error $L_H(x,z)$ is measured between x and z.

A DA is trained to reconstruct a clean input $\mathbf{x}$ from a corrupted version of it. The corrupted input $\tilde{\mathbf{x}}$ is mapped, as with the basic autoencoder, to a hidden representation $f_\theta(\tilde{\mathbf{x}}) = \text{sigmoid}(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b})$ from which we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y}) = (\mathbf{W}'y + \mathbf{b}')$. Instead of minimizing the loss function $L(\tilde{\mathbf{x}}, \mathbf{z})$ between the input and the output, parameters θ and θ' are trained to minimize the average reconstruction error over a clean training set, that is, to have $\mathbf{z}$ as close as possible to the uncorrupted input $\mathbf{x}$, with $L(\mathbf{x}, \mathbf{z}) \propto ||\mathbf{x} - \mathbf{z}||^2$. Denoising autoencoder experiments in computer vision, noise, and corruption were often artificially generated, e.g. by setting some values to zero, so that the denoising autoencoder learns the noise pattern. However in the speech field, the corrupted speech features already exist, and we can use the close-talking microphone speech features to represent the non-corrupted (clean) version.

■ 4.3 Deep Denoising Autoencoder

By using multiple layers of encoders and decoders, the DA can form a deep architecture and become a Deep Denoising Autoencoder (DDA). Note that since we used an affine decoder without nonlinearity, one can easily join the layers of decoders to form one single decoder layer. The system work flow is illustrated in Figure 4.3. Specifically, with parallel clean and noisy speech data available, a DDA can be pre-trained on noisy reverberant speech features and fine-tuned by clean speech features. The rich nonlinear structure in the DDA can be used to learn an efficient transfer function which removes noise in speech while keeping enough phonetically discriminative information to generate good reconstructed features.

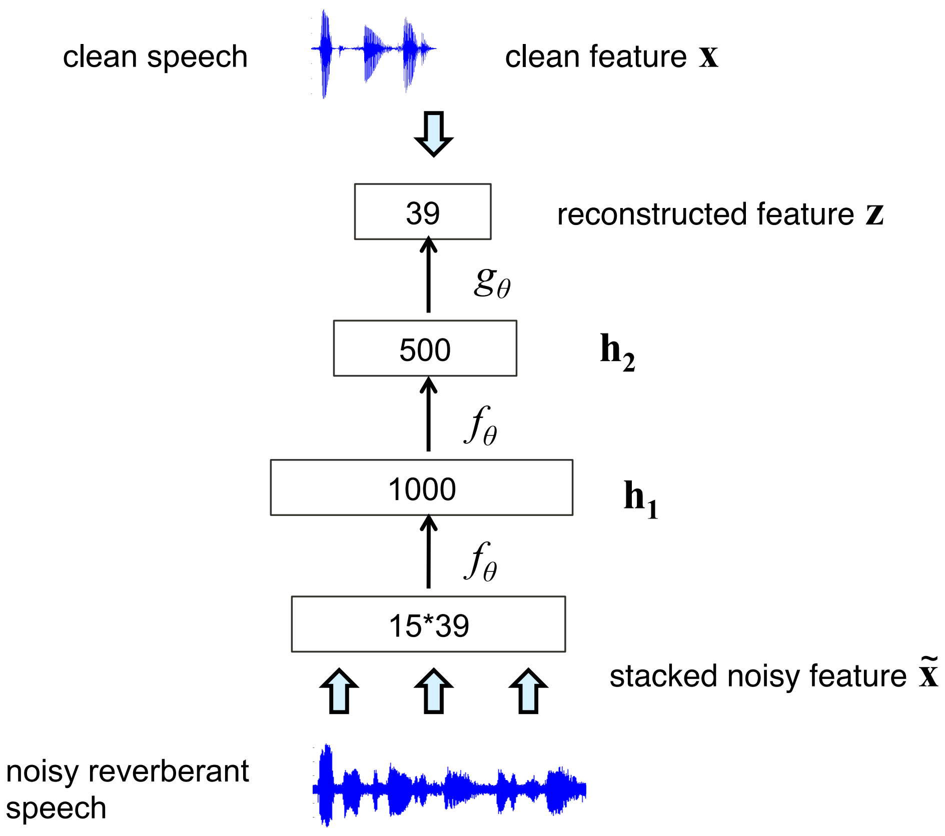

Figure 4-4: Deep Denoising Autoencoder Architecture. This figure gives an example of a DDA containing two encoder layers, with 1000 nodes in the first layer and 500 nodes in the second.

Due to the reverberations, information from previous frames is leaked to the current frame, and with the presence of noise, adjacent frame features become less independent. Note that for our speech denoising and dereverberation task, $\tilde{\mathbf{x}}$ is not of the same dimension as $\mathbf{x}$. We use concatenated features from contiguous frames as $\tilde{\mathbf{x}}$ to encode, and use only the corresponding center frame feature from the clean speech as $\mathbf{x}$. The surrounding frames in $\tilde{\mathbf{x}}$ provide context for denoising and dereverberating the input data.

■ 4.3.1 Training DDA for Robust ASR

Pre-training

Instead of initializing hidden weights with little guidance, we perform pre-training by adopting an efficient approximation learning algorithm proposed by Hinton *et al.* called one-step contrastive divergence (CD-1) [73]. The generative pre-training not only requires no supervised information, but can also put all hidden weights into a proper range which can be used to avoid converging to local optima in the supervised back-propagation based fine-tuning.

Figure 4-5(a) illustrates the pre-training of our DDA. Pre-training consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. After learning one RBM, the status of the learned hidden units given the training data can be used as feature vectors for the second RBM layer. The CD-1 method can be used to learn the second RBM in the same fashion. Then, the status of the hidden units of the second RBM can be used as the feature vectors for the third RBM, and so on. This layer-by-layer learning can be repeated for many times. After pre-training, the RBMs are unrolled to create a deep autoencoder, which is then fine-tuned using back-propagation of error derivatives [74].

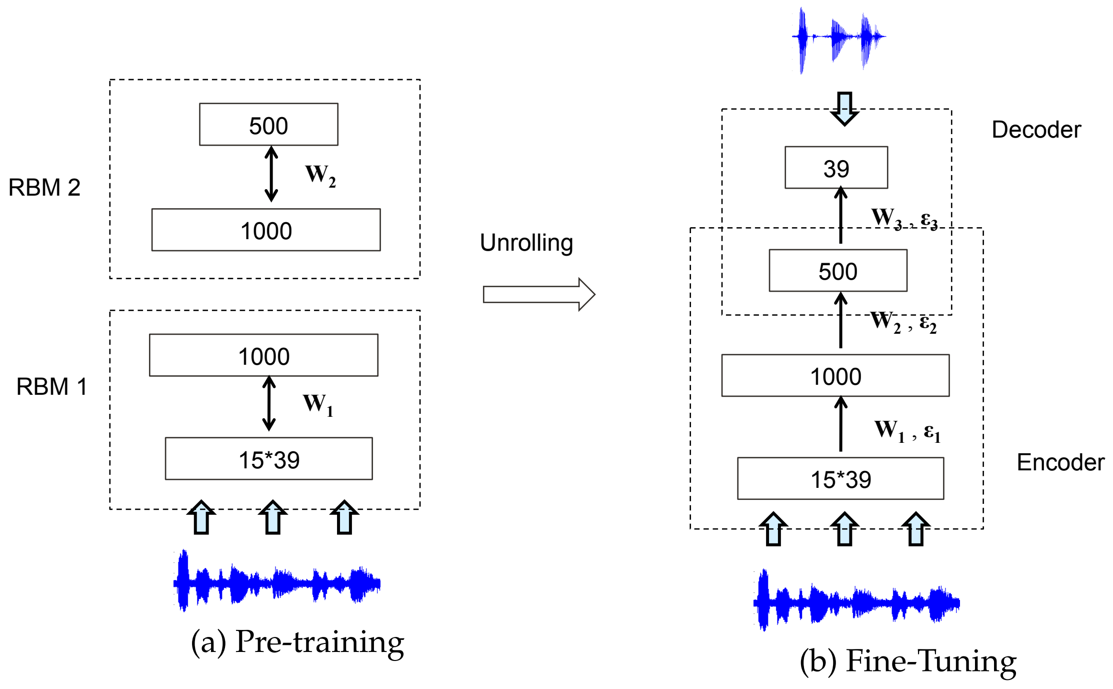

Figure 4-5: Pre-training consists of learning a stack of restricted Boltzmann machines (RBMs). After pre-training, the RBMs are unrolled to create a deep autoencoder, which is then fine-tuned using back-propagation of error derivatives.

Fine-tuning

Figure 4-5(b) demonstrates the fine-tuning state of a DDA. The goal of back-propagation fine-tuning is to minimize the squared error loss on the entire dataset between the reconstructed and clean vectors, and is defined as follows:

$$F = \sum_{i=1}^{U} ||\mathbf{x}^i - \mathbf{z}^i||^2 \tag{4.7}$$

where U is the total number of training cases, $\mathbf{z}^i$ is the i-th reconstructed feature vector, and $\mathbf{x}^i$ is the corresponding clean feature vector. Suppose we have a DDA with M hidden layers and a decoder layer with N outputs (e.g., 39). By taking partial derivatives, the gradient of weights for the decode layer are

$$\frac{\partial F}{\partial \mathbf{W}_l} = \sum_{i=1}^{U} [Z_l(i)E_l(i)]^T \mathbf{v}_l^i \tag{4.8}$$

where each $\mathbf{v}_l^i$ represents the output of the l-th hidden layer for the i-th input. and the gradients for the bias are

$$\frac{\partial F}{\partial \mathbf{b}_l} = \sum_{i=1}^{U} [Z_l(i)E_l(i)]^T \tag{4.9}$$

where Z is the transfer function and E is the error function. For the decoder layer

$$Z_{M+1}(i) = \mathbf{1} \tag{4.10}$$

$$E_{M+1}(i) = \mathbf{x}^i - \mathbf{z}^i. \tag{4.11}$$

For the l-th hidden layer ($l \in [1..M]$),

$$Z_l(i) = (\mathbf{W}_l \mathbf{v}_l^i + \mathbf{ffl}_l) \cdot (1 - \mathbf{W}_l \mathbf{v}_l^i - \mathbf{ffl}_l) \tag{4.12}$$

$$E_l(i) = \mathbf{W}_l Z_l(i) E_{l+1}(i) \tag{4.13}$$

After calculating these gradients, stochastic gradient descent (SGD) is used to update the parameters [69].

■ 4.4 Evaluation

To evaluate the effectiveness of the proposed framework, experiments were conducted. The data corpus used for experimentation and the system specifications are presented in the following sections.

■ 4.4.1 Dataset

ChiME2-WSJ0 is a 5K-vocabulary task in a reverberant and noisy environment, whose utterances are taken from the Wall Street Journal database (WSJ0). The training data set (si_tr_s) contains 7,138 utterances from 83 speakers, the evaluation data set (si_et_05) contains 330 utterances from 12 speakers (Nov'92), and the development set (si_dt_05) contains 409 utterances from 10 speakers. Acoustic models were trained using si_tr_s and some of the parameters (e.g., language model weights) were tuned based on the WERs of si_dt_05. This database simulates a realistic environment. We used the type of data called *Isolated*, which was created as follows: First, clean speech was convolved with binaural room impulse responses corresponding to a frontal position at a distance of 2m from the microphones in a family living room. Second, real-world noises recorded in the same room were added, with the noise excerpts selected to obtain signal-to-noise ratios (SNRs) of -6, 3, 0, 3, 6, and 9 dB without rescaling. Noises were non-stationary such as other speakers, utterances, home noises, or background music.

■ 4.4.2 Experimental Setup

A DDA was first trained on the training set. Next, raw test speech features were processed by the trained DDA. An acoustic model was retrained using the processed features, and the retrained model was utilized for ASR.

Acoustic features

Both the clean and noisy reverberant speech waveforms were parameterized into a sequence of standard 39-dimensional Mel-frequency cepstral coefficient (MFCC) vectors: 12 Mel-cepstral coefficients processed by cepstral mean normalization (CMN), plus logarithmic frame energy and delta and acceleration coefficients. The MFCCs were extracted from 25ms time frames with a step size of 10ms. Prior to feature extraction, the input binaural signals were down-mixed to mono by averaging the two channels together. Although this down-mixing operation lead to a small degradation in WER, we decided to use it in order to focus on the evaluation of the front-end processing technique.

DDA Configuration

All DDA configurations had an input layer with 15*39 units and an affine encoder output layer with 39 units. The pre-training in each configuration was set to stop at the 25th iteration with a learning rate of 0.004 and a batch size of 256. The fine-tuning using backpropagation was set to stop at the 50th iteration using the line search stochastic gradient decent method with a batch size of 256.

Acoustic Model

The HMM/GMM training followed the recipe in [75]. The number of phonemes was 41: 39 phones plus 1 silence (sil) and 1 short pause (sp) model. The output distributions of *sp* and *sil* had their parameters tied. The number of clustered triphone HMM states was 1,860 and was relatively smaller than the conventional setup (more than 2,000 states). Each HMM had three output states with a left-to-right topology with self-loops and no skip. Each HMM state was represented by a GMM with 8 components for phoneme-based HMMs and 16 for silence-based HMMs. The standard WSJ 5K non-verbalized closed bigram language model was used. We only re-estimated the HMM/GMM parameters from a clean speech acoustic model, and did not change the model topology for simplicity. Decoding was performed using HVite with a pruning threshold [76].

■ 4.4.3 Results and Comparison

This section presents the experimental results and analysis on the performance of our proposed DDA denoising method. We first start by varying the DDA configurations to find the appropriate size of the network.

Influence of Number of Layers, Hidden Units per Layer

Table 4.1 presents the results for several DDA configurations and their resulting average WER over 6 SNR scenarios. In the first column, 500 indicates a DDA that had one encoder layer with 500 hidden units, while 500x500 denotes a DDA with two hidden layers each of which had 500 hidden units.

As we observe in Table 1, the average WER is not very sensitive to the DDA configurations. Following this exploration, we utilized the top performing configuration (500x500) for the next set of experiments.

DDA	Average WER
500	35.7%
500x500	34.0%
1000x1000	34.5%
500x500x500	34.2%

Table 4.1: WER vs. different DDA configurations.

Comparison of WER Improvement

Table 4.2 and 4.3 report the results of the system trained on the ChiME2-WSJ0 dataset, as a function of the SNR. The denoising autocoder had two encoder layers with 500 nodes each, had 15x39 input nodes, and emitted a 39-dimensional output. The subsequent HMM-GMM training followed the recipe in [75]. We compared the WER with and without the proposed front-end processing. We found that across the six SNR scenarios, the proposed method improved the ASR accuracy by 16.7%, 19.9%, 25.0%, 22.9%, 21.8%, and 20.5% respectively in absolute difference. The baseline system was trained on matching noisy reverberant data with the exact same settings. These results shows the impact of front-end denoising and dereverberation on improving ASR performance. The improvement is most dramatic in the 0 dB and 3 dB cases. This should be affected by the fact that we did not train a separate DDA for various SNR degrees. Thus the feature mapping generalize well on the middle range cases compared to the extremely high or low SNR cases.

SNR	WER	
	Baseline using MFCC features	Using proposed DDA reconstructed feature
-6dB	70.4%	53.8%
-3dB	63.1%	44.2%
0dB	58.4%	33.4%
3dB	51.1%	28.2%
6dB	45.3%	23.5%
9dB	41.7%	21.2%

Table 4.2: WER under different SNRs with and without the proposed DDA on MFCC features.

SNR	WER	
	Baseline using PLP features	Using proposed DDA reconstructed feature
-6dB	72.7%	55.6%
-3dB	64.5%	45.9%
0dB	58.9%	35.0%
3dB	52.7%	29.6%
6dB	46.8%	24.7%
9dB	42.5%	22.3%

Table 4.3: WER under different SNRs with and without the proposed DDA on PLP features

In comparing our results against those obtained by the participants of the CHiME Challenge [77], ours was among the top two. We note that the CHiME challenge participants employed strategies at the spatial signal, feature, and model levels while we only focus on the front-end feature denoising part [77] . If we were to combine our proposed method with advanced acoustic model and back-end techniques, we anticipate that the results would most likely have improved futher.

Comparison with Other DDA Structures

A variant of our proposed DDA front-end is a symmetric feature mapping from the spliced features to the spliced features. For each frame, there will be multiple resulting reconstructed features. The final reconstructed output takes an average over these multiple copies. Table 4.4 compares this DDA variant with our proposed model.

	WER	
SNR	Averaging DDA	proposed DDA
-6dB	60.37%	53.75%
-3dB	52.10%	44.21%
0dB	46.84%	33.37%
3dB	41.07%	28.19%
6dB	35.72%	23.52%
9dB	30.26%	21.22%

Table 4.4: Comparison between the proposed DDA architecture and averaging DDA architecture. Denoising is performed on MFCC features and the recognition settings are the same for both systems.

■ 4.5 Summary

In this chapter we presented a front-end speech feature denoising and dereverberation method based on DDAs. The proposed framework is unsupervised and learns a stochastic mapping from the corrupted features to the clean ones. We also showed how to train an asymmetric denoising autoencoder to learn from adjacent frames and demonstrated that this improved ASR system performance in the domain of reverberant ASR using the 5K noisy reverberant CHiME-WSJ0 corpus. Our presented method showed a 16% to 25% absolute improvement compared to the baseline. Our results were also among the top two for task 2 compared to the second CHiME Challenge in 2013, and without using any backend technique.

Since this feature denoising method does not preclude the use of many other front-end or back-end methods, this approach can be combined together with various other front-end techniques such as array speech processing, in addition to back-end model adaptation methods. Our presented method is language independent, making it applicable to large vocabulary tasks, and languages beyond English.

Chapter 5

Harvesting Additional Information for Robustness: Heterogenous DNN

Most ASR systems incorporate only a single source of information about their input, namely, features and transformations derived from the speech signal. However, in many applications (such as vehicle-based speech recognition), sensor data and other environmental information are often available to complement audio information.

In the previous two chapters, we pursued ASR robustness from a feature-based perspective, that is, to reconstruct features and compensate noise distortions in the feature domain. We now investigate a model-based approach for robust ASR by training a noise-adaptive acoustic model. In this chapter, we present methods that show how non-speech data sources can be used to improve DNN-HMM ASR systems, and specifically explore this in the context of a vehicle-based ASR task[1].

■ 5.1 DNN-based Acoustic Model

Using neural networks as acoustic models for HMM-based speech recognition was introduced over 20 years ago [79, 80]. Much of this original work developed the basic ideas of hybrid DNN-HMM systems which are used in modern, state-of-the-art ASR systems. However, until much more recently, neural networks were not a standard component in the highest performing ASR systems. Computational constraints and the amount of available training data severely limited the pace at which it was possible to make progress on neural network research for speech recognition.

A DNN is a multi-layer perceptron with many hidden layers between its inputs and outputs. In a modern DNN hidden Markov model (HMM) hybrid system, the DNN is trained to provide posterior probability estimates for context-dependent HMM states.

[1]Portions of this work have been published in [78].

Starting with a visible input $\mathbf{x}$, each hidden layer models the posterior probabilities of a set of binary hidden variables h, given the input visible variables, while the output layer models the class posterior probabilities. The output y for the class s_i is given by

$$\mathbf{y}(s_i) \triangleq P(s_i|\mathbf{x}) \tag{5.1}$$

The networks are trained by optimizing a given training objective function using the standard error back-propagation procedure. For a given objective $\mathbb{L}(\mathbf{x}, \mathbf{s})$, the network weights are updated by

$$(\mathbf{W}_{l,j}, b_{l,j}) \leftarrow (\mathbf{W}_{l,j}, b_{l,j}) + \alpha \frac{\partial \mathbb{L}(\mathbf{x}, \mathbf{s})}{\partial (\mathbf{W}_{l,j}, b_{l,j})} \tag{5.2}$$

where l denotes the l-th layer, and j denotes the j-th node in each layer. α is the learning rate.

The state emission likelihoods $p(\mathbf{x}|s)$ are obtained via Bayes' rule using the posterior probabilities computed by the DNN $p(s|\mathbf{x})$ and the class priors $p(s)$.

$$p(\mathbf{x}|s) \propto \frac{p(s|\mathbf{x})}{p(s)} \tag{5.3}$$

■ 5.1.1 Training Criterion

The default choice for DNN acoustic models is the cross entropy loss function, which corresponds to maximizing the likelihood of the observed label given the input. The cross entropy loss function does not consider each utterance in its entirety. Instead it is defined over individual samples of acoustic input x and senone label y. The cross entropy objective function for a single training pair (x, y) is:

$$-\sum_{k=1}^{K} \mathbb{1}\{y = k\} \log \hat{y}_k \tag{5.4}$$

where K is the number of output classes, and $\hat{y}_k$ is the probability that the model assigns to the input example taking on label k.

Cross entropy is the standard choice when training DNNs for classification tasks, but it ignores the DNN as a component of the larger ASR system. To account for more aspects of the overall system, discriminative loss functions were introduced for ASR tasks. Discriminative loss functions were initially developed for GMM acoustic models [81,82], but were recently applied to DNN acoustic model training [83]. Discriminative training of DNN acoustic models begins with standard cross entropy training to achieve a strong initial solution. The discriminative loss function is used either as a second step, or additively combined with the standard cross entropy function. We can view discriminative training as a task-specific loss function which produces a DNN acoustic model to better act as a sub-component of the overall ASR system.

Nowadays, DNNs have become a competitive alternative to GMMs [84]. DNNs provide an interesting path forward for acoustic modeling as neural networks offer a direct path for increasing representational capacity, provided it is possible to find a good set of DNN parameters. For the rest of this chapter, we investigate DNNs that incorporate heterogeneous information for improving ASR robustness.

■ 5.2 Harvesting Heterogeneous Information

Many researchers have reported different ways of using DNNs to augment ASR robustness. For example, noise-aware training (NAT) was proposed in [85] to improve noise robustness of DNN-based ASR systems. It uses a crude estimate of noise obtained by averaging the first and the last few frames of each utterance as input to the DNN acoustic model. Similarly, [86] uses speech separation to obtain a more accurate estimate of noise. In the above prior work, the additional features are generally derived from the speech signals, and there are limited studies on utilizing existing environmental information.

In this section, we explore DNNs using features extracted from available heterogeneous data, in addition to the features derived from speech signals. In many ASR tasks, e.g., vehicle-based speech recognition and robotic communication systems, various data from the motor sensors, devices and camera sensors, are available, all of which may provide additional clues for ASR. We propose a DNN-based method to incorporate such information by augmenting the input speech features with additional features extracted from the heterogeneous data.

Specifically, vehicle-based speech recognition is a perfect example for such research: it is a noisy reverberant environment, with distant talking from the driver or other passengers, and what's more, it has a large amount of heterogeneous data from modalities beyond just speech. Our approach will be presented via a vehicle-based speech recognition setting.

■ 5.2.1 Heterogeneous Data

The heterogeneous data we explored included engine speed, HVAC fan status, wiper status, vehicle type, signal light status, and many others. These data can all be automatically reported by the vehicle in real time in parallel with the audio, and require no human supervision. Table 5.1 lists the values of the additional data used in our experiments.

Data	Type	Values
speed	real-valued	0 MPH, 35 MPH, 65 MPH
acceleration	real-valued	$\in R(-\infty, \infty)$
cabin_volume	real-valued	15000, 17000, ..
wiper	status	On/Off
window	status	Up/Down
signal light	status	On/Off
weather	status	Rain/No_rain
vehicle_type	categorical	Fiesta, Escape, Mustang, Focus..
ac_fan	categorical	Off/Low/Mid/High

Table 5.1: A list of a subset of the available heterogeneous data

■ 5.2.2 Feature pre-processing

In order to fit this information into the DNN model, pre-processing was conducted on these data. All real-valued feature vectors were normalized globally to zero mean and unit variance, while all status feature vectors were mapped to binaries of 0/1. Therefore, speed was normalized globally to zero mean and unit variance. AC fan status and wiper status were mapped to binary values. Vehicle types were mapped to five distinct values according to the size of the vehicle model, and further normalized globally to zero mean and unit variance. For categorial features, we experimented with one-hot encoding, e.g., [0 0 1], [0 1 0], [1 0 0]. These extracted additional features were concatenated with the spliced corresponding speech features.

■ 5.2.3 Heterogeneous DNN

Background noise and acoustic scenes contain certain acoustic distortion factors of the speech signals. However, such a relationship is highly nonlinear. Because the DNN is composed of multiple layers of nonlinear processing, the network has the capacity to learn and model this relationship directly from the data. It also allows an easy way of combining diverse features, including both discrete and continuous features.

For the DNN with heterogeneous data, the input speech features are computed as in the conventional system. However, these features are now augmented with various combinations of additional features computed from the heterogeneous data, by concatenating the speech features with the additional features. The features are augmented for both training and decoding.

Figure 5-1 gives a diagram of our DNN framework that incorporates additional features. The speech features are derived from speech signals, while the additional features are provided by various sources of sensor information, e.g., camera sensor, motion sensor, speed sensor, fan power etc. As discussed in Section 2.2, in our experiment, these additional features include vehicle speed, HVAC fan status, windshield wiper status, and vehicle type. Thus, the DNN's input is a super vector with the additional features appended to the speech features. At time t, the input is given by

$$\mathbf{v}_{0t} = [\mathbf{x}_{t-\tau}, ..., \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, ..., \mathbf{x}_{t+\tau}, \mathbf{c}_t]. \tag{5.5}$$

Each observation is propagated forward through the network, starting with the lowest layer v_0 . The output variables of each layer become the input variables of the next layer. In the final layer, the class posterior probabilities are computed using a softmax layer. Specifically, instead of 440 MFCC-LDA-MLLT-fMLLR speech features, the DNN input layer had 440 speech features and n-dimensional additional features $\mathbf{c}$ as input, where n

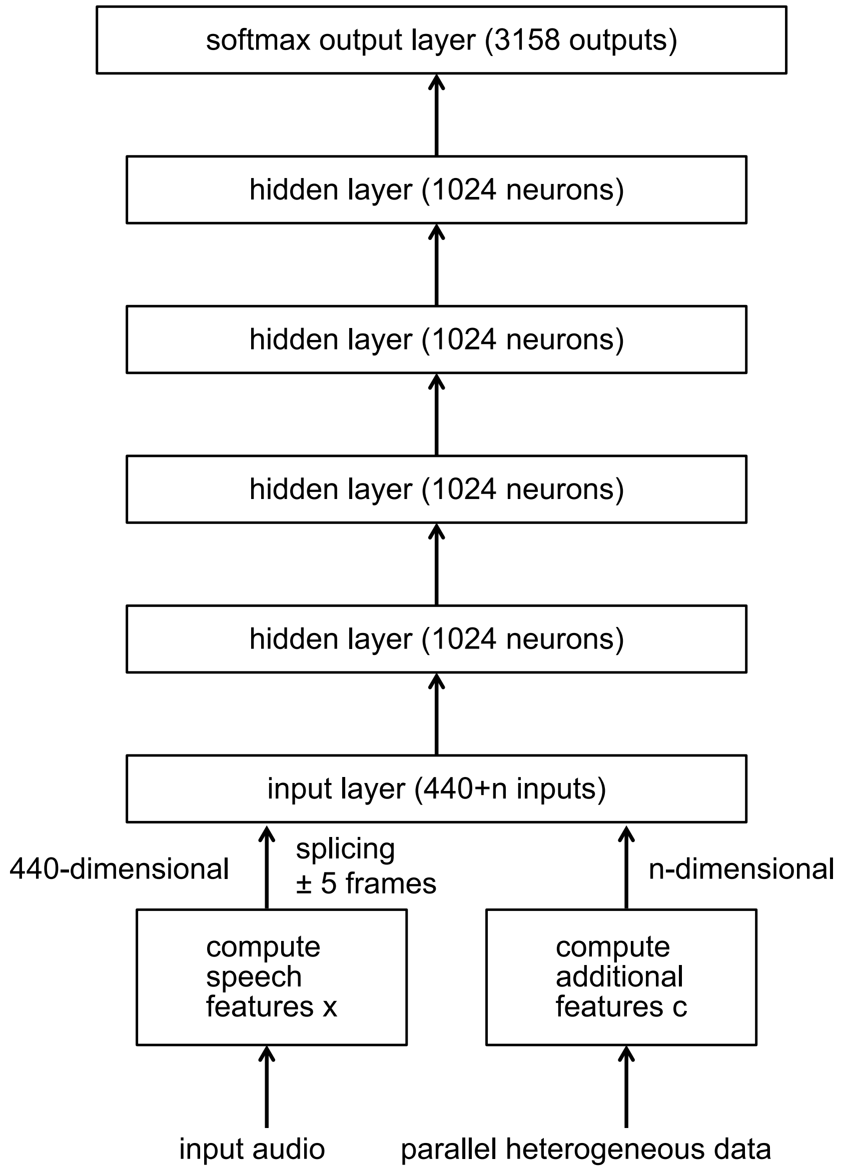
softmax output layer (3158 outputs)
hidden layer (1024 neurons)
hidden layer (1024 neurons)
hidden layer (1024 neurons)
hidden layer (1024 neurons)
input layer (440+n inputs)
440-dimensional
splicing ± 5 frames
n-dimensional
compute speech features x
compute additional features c
input audio
parallel heterogeneous data

Figure 5-1: Diagram of the proposed Heterogeneous DNN system.

was the number of additional features used. During training and testing, these additional features were extracted at the frame level from the parallel heterogeneous data provided by the vehicles.

■ 5.2.4 Feature Selection

We were interested in evaluating the effectiveness of the additional features we extracted from the heterogeneous information and selecting the best performing subsets. Comparison of feature subsets amounts to a combinatorial problem (there are $2^k - 1$ possible subsets for k variables), which becomes computationally infeasible, even for moderate input size. Branch and Bound exploration allows reducing the search for monotonic criteria [87], however, the complexity of these procedures is still prohibitive in most cases. Due to these limitations, we adopted the common (albeit sub-optimal) search method, forward feature selection.

The forward feature selection procedure begins by evaluating all feature subsets which consist of only one input attribute. In other words, we started by measuring the Leave-One-Out Cross Validation (LOOCV) error of the one-component subsets, X_1, X_2, ..., X_M, where M was the input dimensionality; so that we could find the best individual feature, X^1.

Next, forward selection finds the best subset consisting of two components, $X(1)$ and one other feature from the remaining M-1 input attributes. Hence, there are a total of $M - 1$ pairs. Lets assume X^2 is the other attribute in the best pair besides X^1. Afterwards, the input subsets with three, four, and more features are evaluated. According to forward selection, the best subset with m features is the m-tuple consisting of X^1, X^2, ..., X^m, while overall the best feature set is the winner out of all the M steps. Assuming the cost of a LOOCV evaluation with i features is $C(i)$, then the computational cost of forward selection searching for a feature subset of size m out of M total input attributes will be

$$M * C(1) + (M - 1) * C(2) + ... + (M - m + 1) * C(m).$$

■ 5.3 Evaluation

To evaluate the performance of our proposed approach, experiments were conducted. The system specifications as well as results are presented in detail in this section.

■ 5.3.1 Dataset

We evaluated the proposed method on a 30-hour 2K-vocabulary dataset collected by the Ford Motor Company in actual driving, reverberant, and noisy environments. The utterances were recorded in vehicles of varying body styles (e.g., small, medium, large car, SUV, pick-up truck) with talkers (drivers) of varying gender, age, and dialects, under different ambient noise conditions (blower on/off, road surface rough/smooth, vehicle speed 0-65 MPH, windshield wipers on/off, vehicle windows open/closed, etc.). For our experiments, the data were randomly partitioned into three sets with non-overlapping speakers. The training set contained 17,183 utterances from 90 speakers, the development set contained 2,773 utterances from 14 speakers, and the evaluation set contained 1,763 utterances from 9 speakers. The OOV rate was 5.03%. Except the speakers, all other recording conditions were found in all three data sets.

■ 5.3.2 Speech Features

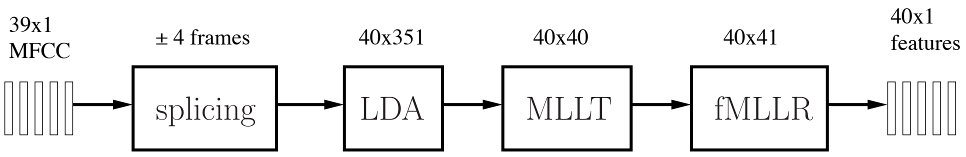

Figure 5-2: Generation of baseline speech features.

As shown in Figure 5-2, the speech waveforms were first parameterized into a conventional sequence of 39-dimensional MFCC vectors based on a 25ms Hamming window, and computed every 10ms. Cepstral mean subtraction was applied on a per speaker basis. The MFCCs were then spliced across 9 frames to produce 351 dimensional vectors. LDA was used to reduce the dimensionality to 40 by using context-dependent HMM states as classes for LDA estimation.An MLLT was then applied to the MFCC-LDA features to better orthogonalize the data [88]. Finally, global fMLLR was then applied to normalize inter-speaker variability [61]. In our experiments, fMLLR was applied both during training and test (also known as speaker-adaptive training (SAT)) [89].

■ 5.3.3 Baseline DNN-HMMs

Experiment Setup

For the baseline speech-only system, we first flat-start trained 26 context-independent monophone acoustic models using MFCC features, then used these models to bootstrap the training of a context-dependent triphone GMM-HMM system. The triphone GMM-HMM system was then retrained by MFCC-LDA-MLLT features. The resulting models contained 3,158 tied triphone states, and 90K Gaussians. This GMM-HMM system was then used to generate fMLLR feature transforms for training and test speakers. The resulting transformed features were input to the DNN. We use the Kaldi toolkit for these experiments [90]. For decoding, a trigram language model with modified Good-Turing smoothing was used. The trigram language model was generated from the 27-hour training data using the sriLM toolkit [91]. The perplexity of the trigram search LM on the Ford development text was 14. Given that we had a small vocabulary task, experiments showed that results using a quadgram LM do not differ much from that of a trigram LM. Therefore, trigram LM was used for all following experiments.

The DNN baseline was trained on the fMLLR transformed MFCC-LDA-MLLT features, except that the features were globally normalized to have zero mean and unit variance. The fMLLR transforms were the same as those estimated for the GMM-HMM system during training and testing. The DNN had 4 hidden layers, where each hidden layer had 1024 units, and 3,158 output units. The input to the network consisted of 11 stacked frames (5 frames on each side of the current frame). We performed pre-training using one-step contrastive divergence [73], whereby each layer was learned one at a time, with subsequent layers being stacked on top of the pre-trained lower layers.

After learning one RBM, the status of the learned hidden units given the training data can be used as feature vectors for the second RBM layer. The CD-1 method can be used to learn the second RBM in the same fashion. Then, the status of the hidden units of the second RBM can be used as the feature vectors for the third RBM, etc. This layer-by-layer learning can be repeated many times.

For the DNNs in this work, we used two different loss objectives. One was Cross Entropy (CE), which minimizes frame error. The other was sequence-discriminative training using state-level minimum Bayes risk (sMBR) criterion.

The state emission likelihoods were obtained via Bayes rule using the posterior probabilities computed by the DNN and the class priors. First we trained the 4-layer DNN using back propagation with CE objective function. After calculating the gradients for this loss objective, stochastic gradient descent (SGD) was used to update the network parameters [69]. For SGD, we used minibatches of 256 frames, and an exponentially decaying schedule that started with an initial learning rate of 0.008 and halved the rate when the improvement in frame accuracy on a cross-validation set between two successive epochs fell below 0.5%. The optimization terminated when the frame accuracy increased by less than 0.1%. Cross-validation was done on a set of 180 utterances that were held out from the training data.

The resulting DNN was then used for sequence training. We used the state-level minimum Bayes risk (sMBR) criterion for the sequence-discriminative training. After calculating the gradients for this loss objective, SGD was used to update the network parameters. The SGD back propagation parameters were the same as with the DNN-CE baseline.

Baseline Results

The ASR performance on the evaluation set is shown in Table 5.2. We observed an 0.5% absolute improvement with discriminative training versus cross entropy. Both baseline systems will be used in the next section.

Baseline systems	WER
DNN-CE	7.0%
DNN-sequence	6.5%

Table 5.2: WER of baseline systems using CE training, followed by sequence training.

■ 5.3.4 Heterogeneous DNN-HMMs

In this section we present the experimental results of our proposed Heterogeneous DNN-HMM system. DNNs augmented with different combinations of the additional features were trained. For each distinct additional feature combination, we trained a separate DNN. The network configurations and training/decoding procedures remained the same as in Section 5.3.1.

Table 5.3 provides the baseline WER results, as well as results from our proposed system using different additional feature combinations. Our experiments show that systems with an increasing number of additional features improved WERs. Here we only list the results from seven feature combination candidates that improved WER the most in Table 5.3.

System	Additional Feature	WER	WERR
GMM	(Baseline)	11.86	-
DNN	(Baseline)	7.04	-
	+ speed	6.71	4.7
	+ ac_fan	6.72	4.5
	+ wiper	6.81	3.3
	+ vehicle	6.89	2.1
	+ speed, ac_fan	6.63	5.8
	+ speed, ac_fan, wiper	6.61	6.1
DNN+sMBR	(Baseline)	6.53	-
	+ speed	6.46	1.1
	+ ac_fan	6.45	1.2
	+ wiper	6.29	3.7
	+ vehicle	6.42	1.7
	+ speed, ac_fan	6.19	5.2
	+ speed, ac_fan, wiper	6.22	4.7

Table 5.3: Word error rate (WER) and Word error reduction rate (WERR) of the proposed systems with additional features.

We observed that speed provides the lowest WER among the individual additional features. This might be related to the fact that speed is a dominant contributor to noise. With more additional features included, the WER continues to drop. The complete additional feature set of speed, ac_fan, wiper_status and vehicle_type gave the lowest WER among all additional feature combinations. This demonstrates that having such information is helpful for noise robustness, and that the DNNs are able to learn useful relationships between the additional heterogeneous features and noisy speech features, which enables the model to generate more accurate posterior probabilities.

Table 5.4 displays a comparison between the performance of the baseline systems against our systems that include additional features. The additional features used here were $\mathbf{c} = (speed, ac\_fan, wiper, vehicle)$. We compared the WER with and without the additional features. We can see that WER was reduced by 6.3% relative to the the DNN-CE baseline. WER was reduced by 5.5% relative to the DNN-sequence baseline. The absolute gain doesn't seem to be huge due to our comparatively small baseline WER, but the 6.3% comparative gain suggests that if this approach is used on a larger-vocabulary task, the absolute gain might be more substantial.

	without AFs	with AFs	WERR
DNN-CE	7.0%	6.6%	6.3%
DNN-sequence	6.5%	6.2%	5.5%

Table 5.4: WER comparison of with v.s. without the additional features, and the Word error reduction rate (WERR).

To further demonstrate the effectiveness of the heterogeneous data, in Table 5.5 we provide the noise-adaptive training (NAT) results using signal-to-noise ratios (SNRs) derived from speech signals. SNRs were computed using the NIST metric. The results demonstrate that the additional features we used outperformed the SNRs computed from the the speech signals in both CE and sequence training cases. This indicates that the heterogeneous data contained richer information about the environment than the SNRs computed from the speech signals. These additional features are better alternatives to SNR for noise-adaptive training or environment-aware training. Moreover, the additional features and SNRs were not exclusive. It could be that using them jointly would lead to a better adaptation scenario.

	with AFs	with SNR
DNN-CE	6.60%	6.91%
DNN-sequence	6.17%	6.51%

Table 5.5: WER comparison of using additional features vs. using SNRs

We also compared the impact of model depth and size on the performance. Table [to be inserted] shows the experimental results obtained from different model sizes.

■ 5.4 Summary

We have shown that DNNs can adapt to environment characteristics if we augment standard acoustic features by appending features extracted from heterogeneous data. The DNN learns useful relationships between these heterogeneous data and noisy speech features, which enables the model to generate more accurate posterior probabilities. This was motivated by the success of noise-aware training where SNRs had been found to be useful for noise robustness because it served to characterize the noise level.

Our experiments demonstrated that WER could be reduced by 6.3% relatively with the additional features compared to the baseline DNN-HMM hybrid system. Moreover, this outperformed the improvement brought by noise-aware training using SNRs, by a large margin. This indicates that the heterogeneous data contained richer information about the environment than the SNRs. If other heterogeneous data and representations contained similar information about the environment, then they can possibly also be used to do environment-adaptation of DNN-HMM system in the same way, for better ASR robustness.

This framework can also be generalized to incorporate other features, both continuous and discrete, for various ASR tasks. For example, visual information, acoustic sensor data, and machine status can be explored using this approach. Moreover, different DNN structures can also be investigated, by feeding the additional features at different layer levels in the network.

Chapter 6
E-vector: Blind Feature for Noise Robustness via Bottleneck DNN

In the previous chapter, we investigated how to build a noise-aware model that was able to adapt to the environment by harvesting heterogeneous data. The results showed significant improvements over the conventional model. However, it has a drawback that occurs when the type or dimensionality of the feature extracted from the heterogeneous data changes. The model, once trained, is fixed to a specific number of heterogeneous feature vectors, and thus is not flexible enough to deal with missing features, or accommodate a different feature dimensionality.

In this chapter, we propose an alternative method from the feature-based perspective, to generate a novel environment-aware feature that can be used for adaptation[1]. Once trained, feature generation does not rely on the supply of any additional information, and thus is independent of feature dimensionality or even the type of the additional features. It is derived from an i-vector, and we refer to it as an e-vector.

In contrast to the i-vector's success and popularity in speaker related tasks, there has been little research on its usefulness in channel and environment applications. i-vectors are extracted in a way that makes no distinction between channel and speaker variability. Inspired by this fact, we propose features derived from i-vectors in the total variability space to capture environmental variability only.

Methods for extracting the proposed environment-aware feature will be described, and the effectiveness of the proposed feature will be evaluated using the same multi-modal vehicle-based speech corpus as the previous chapter. Additionally, we also found that the proposed feature can be applied to blind noise condition classification.

[1]Portions of this work have been published in [92].

The rest of this chapter is organized as follows. Section 6.1 briefly introduces i-vectors. Next, the e-vector is proposed in Section 6.2, and two extraction methods are presented. We evaluate e-vector adaptation in Section 6.3, and e-vector environment identification in Section 6.4. We conclude with Section 6.5.

■ 6.1 Introduction

i-vectors have been extracted by factor analysis, and have demonstrated success for speaker recognition and verification [50, 93, 94]. Recently, the i-vector method has been successfully applied to speaker and channel adaptation in speech recognition.

To compute the i-vector, suppose we are given recordings that consist of the speech of a single speaker. Each recording is assumed to be represented by a Gaussian mixture model (GMM), and principal components analysis is applied to the GMM supervectors. Thus, the basic assumption is that all utterance supervectors are confined to a low dimensional subspace of the GMM supervector space so that each utterance supervector can be specified by a small number of coordinates. For a given recording, these coordinates of the corresponding supervector define the i-vector representation. The i-vector approach models supervector adaptation to a given sequence of frames in a low dimensional space called the total variability space. In the i-vector framework, each speech utterance can be represented by a GMM supervector, which is assumed to be generated as follows:

$$\boldsymbol{s} = \boldsymbol{m} + \boldsymbol{T}\boldsymbol{w}$$

where $\boldsymbol{s}$ is the supervector defined by an adapted universal background model (UBM), $\boldsymbol{m}$ is the speaker independent and channel independent supervector, $\boldsymbol{T}$ is a rectangular matrix of low rank, and $\boldsymbol{w}$ is a random vector having a standard normal distribution prior N(0, 1). The i-vector is a Maximum A Posteriori (MAP) point estimate of the latent variable $\boldsymbol{w}$ adapting the UBM (supervector m) to a given recording.

■ 6.2 E-vector Extraction Method

Inspired by i-vectors, we propose to extract a feature in the total variability space, which we refer to as e-vectors, that specifically captures the environmental noise variability.

One way of focusing on the environment related information in an i-vector is via a dimensionality reduction method, e.g., LDA. We refer to the environmental feature obtained by this method as an LDA-evector.

We also propose a second method to extract environmental features from i-vectors, by training a bottleneck neural network (BN-NN). Both methods will be described in the following section.

■ 6.2.1 LDA-based Projections

The i-vector representation we obtained so far is speaker and channel dependent. In order to compensate the within class inter-speaker variability and the session variability, a supervised dimensionality reduction (LDA), can be used to find a low dimensional channel-dependent subspace.

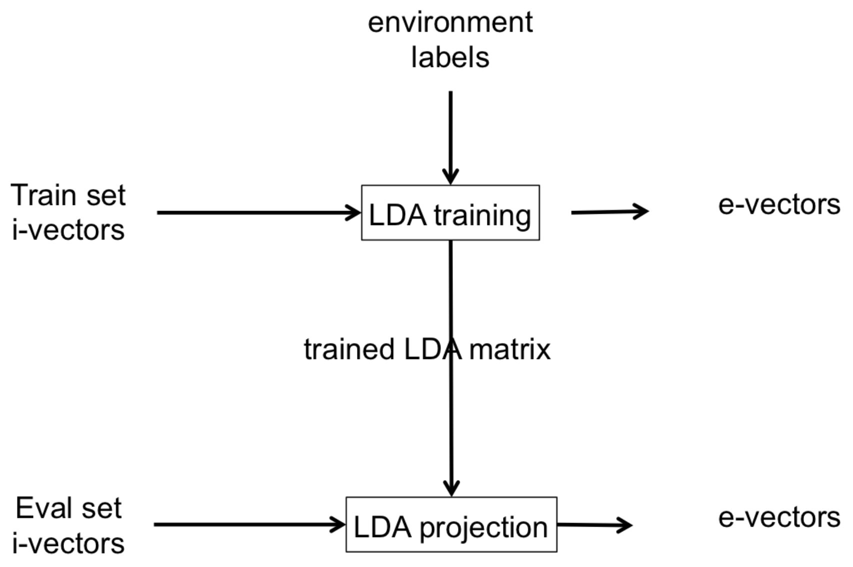

Figure 6-1: Evector extraction via LDA using noise labels.

The idea behind this approach is to seek new orthogonal axes to better discriminate among different noise environments. The axes found must satisfy the requirement of maximizing noise condition variance but minimizing within class variance. The within class variance $\boldsymbol{S_w}$ and between class variance $\boldsymbol{S_b}$ can be written as

$$\begin{aligned} \boldsymbol{S_b} &= \sum_{c=1}^{C} \left(\boldsymbol{i}_c - \bar{\boldsymbol{i}}\right) \left(\boldsymbol{i}_c - \bar{\boldsymbol{i}}\right)^t \\ \boldsymbol{S_w} &= \sum_{c=1}^{C} \frac{1}{n_c} \sum_{n=1}^{n_c} \left(\boldsymbol{i}_n^c - \bar{\boldsymbol{i}}\right) \left(\boldsymbol{i}_n^c - \bar{\boldsymbol{i}}\right)^t \end{aligned} \tag{6.1}$$

where $\bar{i}$ is the mean of i-vectors for each class, C is the number of classes, and n_c is the number of utterances for each class.

In the case of i-vectors, the speaker population mean vector is equal to the null vector since, in factor analysis, these i-vectors have a standard normal distribution, which has a zero mean vector. This maximization is used to define a projection matrix A composed of the best eigenvectors (those with highest eigenvalues) of the general eigenvalue equation:

$$\boldsymbol{S_b v} = \boldsymbol{\Lambda S_w v}, \tag{6.2}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. The i-vectors are then submitted to the projection matrix obtained from LDA.

After the projection matrix is trained, the same LDA projection will be applied on all training, development, and evaluation sets. Figure 6-1 depicts an overview of this process.

■ 6.2.2 BN-NN based

A Bottleneck Neural-Network (BN-NN) refers to a particular topology of a NN, such that one of the hidden layers has a significantly lower dimensionality than the surrounding layers. It is assumed that such a layer, referred to as the bottleneck layer, compresses the information needed for mapping the NN input to the NN output. A bottleneck feature vector is the vector of values at the bottleneck layer, as a by-product of forwarding a primary input feature vector through the BN-NN. In other words, after a BN-NN is

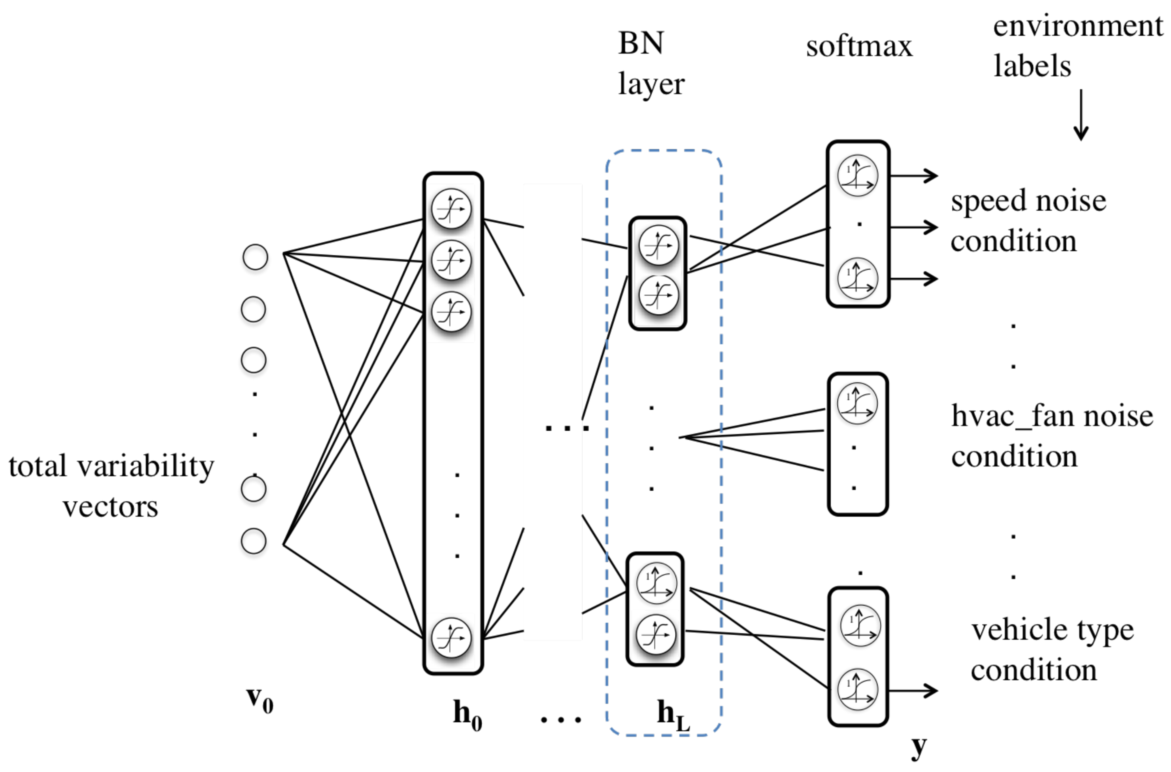

Figure 6-2: BN-evector extraction via bottleneck DNN using noise labels.

trained for its primary task, the bottleneck layer is declared to be the output layer and all succeeding layers are ignored.

In 2011, Yu and Seltzer applied a DNN for extracting BN features, with the bottleneck being a small hidden layer placed in the middle of the network [95]. Bottleneck features have shown success in speaker adaptation and language identification [96,97].

Following Sainath *et al.* we applied a low-rank approximation to the weights of the softmax layer of the network. This was done by replacing the usual softmax layer weights by a linear layer with a small number of hidden units, followed by a softmax layer. More specifically, a new BN output layer with r linear hidden units was inserted into the last weight matrix with a hidden layer of size h, and a softmax layer with s state posterior outputs. This changes the number of parameters from $h * s$ to $r * (h + s)$. There are two benefits of using this method. First, it ensures the best achievable frame accuracy even with a relatively small r. Second, the linearity of the output for the BN layer prevents any loss of information when we treat the DNN as a feature extractor.

The configuration for our BN-NN was 487x1024x1024xMxN, where M was the size of the bottleneck, and N was the number of targets. One of the key questions is what targets to train the BN features on. Since our goal was to obtain a bottleneck feature that was able to separate different noise conditions, the noise conditions were used as the targets.

Instead of a single bottleneck layer for all mixed noise conditions, our proposed BN-NN structure extracted separate BN-evectors for each individual noise condition, with the concatenated BN-evector as the final bottleneck feature output. The extraction method is given in Figure 6-2. The bottleneck layers shared the same weights for the preceding hidden layers, and each had its own softmax output with corresponding noise condition targets. Once we trained the BN-evector extracting network using a certain noise environment label, we used the same network to generate a BN-evector for the eval set.

■ 6.3 ASR Environment Adaptation using E-vector

To evaluate the effectiveness of the proposed e-vector, we performed environment adaptation experiments by augmenting an ASR system with an e-vector.

■ 6.3.1 Experiment Setup

I-vector Extraction

For this work, an i-vector extractor of dimension 92 was trained, using a UBM consisting of 512 mixtures. The features used to train the UBM was a 40 dimensional LDA-transformed feature of nine stacked MFCC frames of dimension 13. We trained the i-vector and the underlying UBM using the same training set utterances.

Acoustic Model

For recognition, a hybrid DNN-HMM was used as our acoustic model. The network configurations and training/decoding procedures remained the same as in Chapter 5. The speech features were also extracted in the same way. The fMLLR-adapted [61] MFCC-

LDA-MLLT features were used, and fMLLR was applied both during training and test. For completion, the baseline results are presented in Table 6.1 on the vehicle test corpus.

Model	WER (%)
GMM	11.86
hybrid DNN-CE	7.04
hybrid DNN-sMBR	6.53

Table 6.1: ASR baseline WER.

6.3.2 LDA-evector Adaptation

The LDA-evectors were extracted by first training the LDA projection matrix for each corresponding noise condition. In our modeling, each class was made up of all the recordings of each noise condition. After training, all i-vectors were transformed using the learnt projections for each noise condition.

Using the LDA-evector generated, the system was adapted as follows. The speech features were derived from speech signals in the same way as the baseline model. The DNN acoustic models input was a supervector with the e-vector appended to the speech features. In this way, the e-vector provides conditioning information about the background environment.

Results

LDA Projection Label used	Evector dim	WER
speed	2	6.3
hvac_fan	1	5.8
wiper	1	6.3
vehicle_type	4	5.9
speed+hvac_fan	5	5.7
speed+hvac_fan+wiper	11	5.5
speed+hvac_fan+wiper+vihecle_type	20	5.5

Table 6.2: ASR WER using LDA-evectors

The results of environment adaptation using LDA-evector are displayed in Table 6.2. Our experiments show that augmenting the system with e-vectors extracted from different noise targets improved recognition accuracies. Here we only list the results from seven feature combination candidates that improved WER the most in Table 6.2. We can see that the model trained with the LDA-evector extracted from {speed, hvac_fan, wiper, vehicle_type} improved the relative WER by 16% compared to the DNN-sMBR baseline, reducing the WER from 6.5% to as low as 5.5%.

■ 6.3.3 BN-evector Adaptation

The results of environment adaptation using the BN-evector are displayed in Table 6.3. We can see that a 17% relative improvement is achieved, reducing WER from 6.5% to as low as 5.4%. Comparing the results in Table 6.2 and Table 6.3, we observe that the BN-evector performs slightly better than the LDA-evector.

BN-NN output target	BN-evector dim	WER
speed	10	6.3
hvac_fan	10	5.7
wiper	10	6.1
vehicle	10	5.8
speed+hvac_fan	20	5.7
speed+hvac_fan+wiper	30	5.5
speed+hvac_fan+wiper+vehicle_type	40	5.4

Table 6.3: ASR WER using BN-evectors

BN-NN output target	BN-evector dim	WER
speed	2	6.3
	5	6.28
	10	6.28
hvac_fan	2	5.76
	5	5.74
	10	5.74
wiper	2	6.17
	5	6.14
	10	6.14
vehicle	2	5.94
	5	5.82
	10	5.81

Table 6.4: The effect of the size of the BN Layer.

We also investigated the effect of the size of the bottleneck in the BN-NN, which directly influences the dimensionality of the resulting BN-evector. Results in Table 6.4 show that the performance gain stops increasing at around 10. In Table 6.3 and the rest of this chapter, we keep our BN size to be 10 for each noise condition.

6.3.4 Comparison and Fusion

To better understand the performance, in Table 6.5 we evaluated the WER obtained by augmenting the original acoustic features with raw i-vectors. We can see that i-vector adaptation gives a 2.5% relative improvement. We also varied the i-vector dimensionality to evaluate its effect, and found that 92-dimensions gave the best performance, although this effect was very mininal.

I-vector dim	WER(%)
92	6.37
50	6.38

Table 6.5: ASR WER using raw i-vectors, with different i-vector dimensionalities.

Table 6.6 compares the improvement brought by i-vector adaptation versus e-vector adaptation. Both the LDA-evector and BN-evector outperformed the i-vector by a large margin. This indicates that the e-vector was able to capture accurate information about the noise environment. The reason that the i-vector was not as good might be due to redundancy in the non-environmental information given the with speech features and could lead to a biased adaptation.

System	WER(%)
baseline	6.5
+i-vector	6.3
+LDA-evector	5.5
+BN-evector	5.4

Table 6.6: ASR WER comparison using i-vector and proposed e-vectors.

Table 6.7 reports the ASR performance of concatenating i-vectors and e-vectors for adaptation. The fusion result is slightly better than using the i-vector or e-vector individually. This indicates that the i-vector and the e-vector are complementary, although the e-vector is trained from an i-vector. In practice, the i-vector and e-vector can be used in conjunction with each other to achieve better adaptation results.

System	WER(%)
baseline	6.5
+ivector+LDA-evector	5.4
+ivector+BN-evector	5.3

Table 6.7: ASR WER fusing i-vector and e-vector.

■ 6.4 E-vector for Noise Environment Identification

In the previous section, we demonstrated the effectiveness of using e-vector for environment adaptation. This indicates that the e-vector characterized the underlying noise conditions.

Visualization

As we can see from the scatter plot 6-3, trained from the speed target, the e-vector tended to separate according to the underlying speed condition.

In this section, we are interested in evaluating its usefulness in identifying different noise conditions.

Preliminary Study

We used the trained BN network to extract BN-evectors from the eval audio files, and then use the BN-evector to blindly classify certain noise condition of the corresponding audio. Classification was done using an SVM classifier with a Radial Basis Function kernel. The LIBSVM toolkit was used for training and testing [98]. Table 6.8 reports the classification equal error rate (EER) on the speed condition using a BN-evector. Table 6.9 reports the classification accuracy on hvac fan status using a BN-evector. Table 6.10 reports the classification accuracy on wiper status using a BN-evector. Table 6.11 reports the classification accuracy on vehicle type using a BN-evector. We can see that BN-evector is capable of classifying different noise conditions. Similar results can be observed in Tables 6.12, 6.13, 6.14, and 6.15.

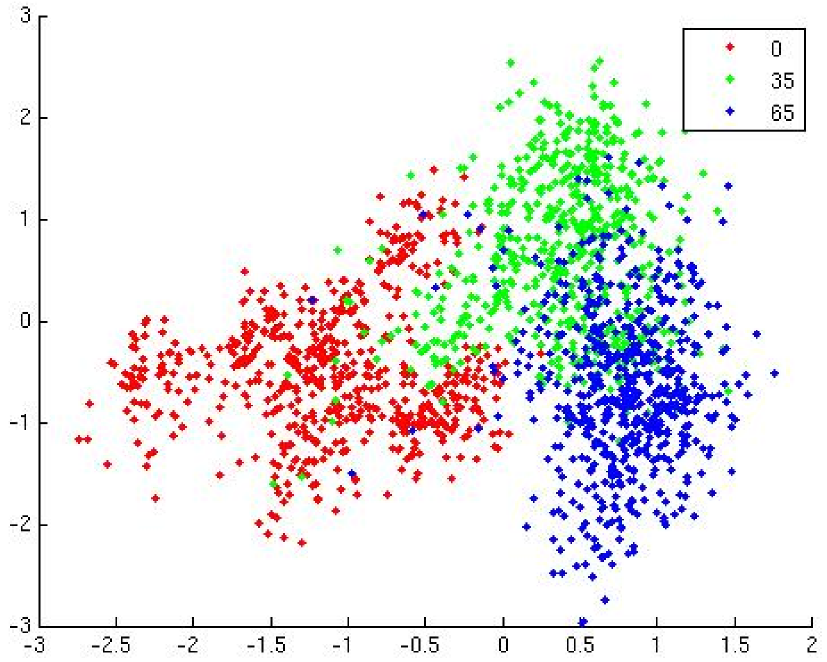

Figure 6-3: A low dimensional projection of an LDA-evector, color-coded by the underlying recording's speed condition, 0/35/65 MPH respectively.

BN-evector extracted with labels	classification accuracy (%)
speed	80.5
speed+hvac_fan	81

Table 6.8: Speed classification accuracy using BN-evector.

BN-evector extracted with labels	classification accuracy (%)
hvac_fan	83.5
speed+hvac_fan	84.5

Table 6.9: hvac_fan status classification accuracy using BN-evector.

BN-evector extracted with labels	classification accuracy (%)
wiper	82
wiper+speed_fan	82.5
wiper+speed+hvac_fan	83.5

Table 6.10: Wiper status classification accuracy using BN-evector.

BN-evector extracted with labels	classification accuracy (%)
vehicle	78.5
vehicle+speed	79.0
vehicle+hvac_fan	79.9

Table 6.11: Vehicle type classification accuracy using BN-evector.

LDA-evector extracted with labels	classification accuracy (%)
speed	78.8
speed+hvac_fan	79.7

Table 6.12: Speed classification accuracy using LDA-evector.

LDA-evector extracted with labels	classification accuracy (%)
hvac_fan	81.3
speed+hvac_fan	82.2

Table 6.13: hvac_fan status classification accuracy using LDA-evector.

LDA-evector extracted with labels	classification accuracy (%)
wiper	80.1
wiper+speed_fan	80.4
wiper+speed+hvac_fan	81.2

Table 6.14: Wiper status classification accuracy using LDA-evector.

LDA-evector extracted with labels	classification accuracy (%)
vehicle	77.6
vehicle+speed	78.3
vehicle+hvac_fan	79.2

Table 6.15: Vehicle type classification accuracy using LDA-evector.

Results show that the both the LDA-evector and BN-evector are able to identify most of the noise conditions correctly. This indicates that the e-vectors contain useful information about the noise environment. We also observe that, in all four experiments, classification accuracy is improved using e-vectors training with more noise labels.

■ 6.5 Summary

In this chapter, we presented a novel feature representation trained from i-vector, which we refer to as e-vector, that specifically captures the channel and environment variability. We have given two e-vector extraction methods, one via an LDA projection, and the other via a modified Bottleneck DNN. Our experiments on environment adaptation using the proposed e-vectors brought a 17% relative WER improvement on real-recorded noisy corpora. This outperformed raw i-vector adaptation improvement by a large margin. We also showed that the result can be further improved when fusing i-vectors and e-vectors together. The extracted environment features can also be applied to the blind environment classification problem, and our experiment demonstrated the capability of separating noise conditions using the proposed e-vector.

Bibliography

[1] Nelson Morgan, Jordan Cohen, and Sree Hari Krishnan, "Ouch project (outing unfortunate characteristics of hmms)," 2013.

[2] Jont B Allen and David A Berkley, "Image method for efficiently simulating small-room acoustics," *The Journal of the Acoustical Society of America*, vol. 65, no. 4, pp. 943–950, 1979.

[3] KH Davis, R Biddulph, and Stephen Balashek, "Automatic recognition of spoken digits," *The Journal of the Acoustical Society of America*, vol. 24, no. 6, pp. 637–642, 1952.

[4] James Baker, "The dragon system–an overview," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 24–29, 1975.

[5] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton, "Deep belief networks for phone recognition," in *Nips workshop on deep learning for speech recognition and related applications*. Vancouver, Canada, 2009, vol. 1, p. 39.

[6] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[7] B. Schrauwen and E. Antonelo, "Timit benchmark results," 03 2010.

[8] C. Nikias and J. Mendel, "Signal processing with higher-order spectra," *Signal Processing Magazine*, vol. 10, no. 3, pp. 10–37, 1993.

[9] R. A. Wiggins, "Minimum entropy deconvolution," *Geoexploration*, vol. 16, no. 1, pp. 21–35, 1978.

[10] Li Deng and Xuedong Huang, "Challenges in adopting speech recognition," *Communications of the ACM*, vol. 47, no. 1, pp. 69–75, 2004.

[11] Matthias Wölfel and John McDonough, *Distant Speech Recognition*, Wiley, London, 2009.

[12] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*, Prentice hall PTR, 2001.

[13] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Jour. of ASA*, vol. 87, no. 4, pp. 1738–1752, Apr. 1990.

[14] Kai-Fu Lee, "On large-vocabulary speaker-independent continuous speech recognition," *Speech communication*, vol. 7, no. 4, pp. 375–379, 1988.

[15] Steve J Young, Julian J Odell, and Philip C Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 307–312.

[16] 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.

[17] Yariv Ephraim and David Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 6, pp. 1109–1121, 1984.

[18] Steven Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 27, no. 2, pp. 113–120, 1979.

[19] Mohamed Afify, "Accurate compensation in the log-spectral domain for noisy speech recognition," *IEEE transactions on speech and audio processing*, vol. 13, no. 3, pp. 388–398, 2005.

[20] Li Deng, Jasha Droppo, and Alex Acero, "Enhancement of log mel power spectra of speech using a phase-sensitive model of the acoustic environment and sequential estimation of the corrupting noise," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 2, pp. 133–143, 2004.

[21] Fu-Hua Liu, Richard M Stern, Xuedong Huang, and Alejandro Acero, "Efficient cepstral normalization for robust speech recognition," in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1993, pp. 69–74.

[22] Angel De La Torre, Antonio M Peinado, José C Segura, José L Pérez-Córdoba, M Carmen Benítez, and Antonio J Rubio, "Histogram equalization of speech representation for robust speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 355–366, 2005.

[23] Hynek Hermansky and Nelson Morgan, "Rasta processing of speech," *IEEE transactions on speech and audio processing*, vol. 2, no. 4, pp. 578–589, 1994.

[24] Chia-Ping Chen and Jeff A Bilmes, "Mva processing of speech features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 257–270, 2007.

[25] Mark JF Gales and Steve J Young, "Cepstral parameter compensation for hmm recognition in noise," *Speech communication*, vol. 12, no. 3, pp. 231–239, 1993.

[26] Jean-Luc Gauvain and Chin-Hui Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Trans. on SAP*, vol. 2, no. 2, 1994.

[27] C. J. Leggetter and P. C. Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models,” *Jour. on CSL*, pp. 171–185, 1995.

[28] Richard M Stern, Bhiksha Raj, and Pedro J Moreno, “Compensation for environmental degradation in automatic speech recognition,” in *Robust Speech Recognition for Unknown Communication Channels*, 1997.

[29] Jinyu Li, Li Deng, Dong Yu, Yifan Gong, and Alex Acero, “High-performance hmm adaptation with joint compensation of additive and convolutive distortions via vector taylor series,” in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007, pp. 65–70.

[30] Xue Feng, Kenichi Kumatani, and John McDonough, “The cmu-mit reverb challenge 2014 system: description and results,” in *Proceedings of REVERB Challenge Workshop, p1*. Citeseer, 2014, vol. 9.

[31] G. C. Carter, “Time delay estimation for passive sonar signal processing,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 463–469, 1981.

[32] M. Omologo and P. Svaizer, “Acoustic event localization using a crosspower–spectrum phase based technique,” in *Proc. of ICASSP*, 1994, vol. II, pp. 273–6.

[33] J. H. DiBiase, H. F. Silverman, and M. S. Brandstein, “Robust localization in reverberant rooms,” in *Microphone Arrays*, M. Brandstein and D. Ward, Eds., chapter 4. Springer Verlag, Heidelberg, Germany, 2001.

[34] S. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice–Hall, Englewood Cliffs, NJ, 1993.

[35] Ulrich Klee, Tobias Gehrig, and John McDonough, “Kalman filters for time delay of arrival–based source localization,” *Journal of Advanced Signal Processing, Special Issue on Multi–Channel Speech Processing*, August 2005.

[36] Kenichi Kumatani, Liang Lu, John McDonough, Arnab Ghoshal, and Dietrich Klakow, "Maximum negentropy beamforming with superdirectivity," in *European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, 2010.

[37] Kenichi Kumatani, John McDonough, Barbara Rauch, and Dietrich Klakow, "Maximum negentropy beamforming using complex generalized gaussian distribution model," in *Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Pacific Grove, CA, USA, 2010.

[38] Kenichi Kumatani, John McDonough, Dietrich Klakow, Philip N. Garner, and Weifeng Li, "Adaptive beamforming with a maximum negentropy criterion," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, pp. 994–1008, July 2009.

[39] R. Zelinski, "A microphone array with adaptive post-filtering for noise reduction in reverberant rooms," in *Proc. ICASSP*, New York, NY, USA, April 1988.

[40] K. Uwe Simmer, Joerg Bitzer, and Claude Marro, "Post-filtering techniques," in *Microphone Arrays*, M. Branstein and D. Ward, Eds., pp. 39–60. Springer, Heidelberg, 2001.

[41] Ivan Himawan, Iain McCowan, and Sridha Sridharan, "Clustered blind beamforming from ad-hoc microphone arrays," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 19, pp. 661–676, 2011.

[42] M. Lincoln, I. McCowan, I. Vepa, and H. K. Maganti, "The multi–channel Wall Street Journal audio visual corpus (MC–WSJ–AV): Specification and initial experiments," in *Proc. of ASRU*, 2005, pp. 357–362.

[43] H. L. Van Trees, *Optimum Array Processing*, Wiley, New York, 2002.

[44] Keisuke Kinoshita, Marc Delcroix, Takuya Yoshioka, Tomohiro Nakatani, Emanul Habets, Reinhold Häb-Umbach, Volker Leutnant, Armin Sehr, Walter Kellermann, Roland Maas, Sharon Gannot, and Bhiksha Raj, "The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech,"

in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, October 2013.

[45] "Reverb challenge webpage," http://reverb2014.dereverberation.com.

[46] T. Robinson, J. Fransen, D. Pye, J. Foote, and S. Renals, "WSJCAMO: a British English speech corpus for large vocabulary continuous speech recognition," in *Proc. ICASSP*, 1995.

[47] Guillaume Lathoud, Jean-Marc Odobez, and Daniel Gatica-Perez, "AV16.3: an audio-visual corpus for speaker localization and tracking," in *Proceedings of the MLMI'04 Workshop*, 2004.

[48] M. Wölfel and J.W. McDonough, "Minimum variance distortionless response spectral estimation, review and refinements," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 117–126, Sept. 2005.

[49] Douglas A Reynolds, Thomas F Quatieri, and Robert B Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing*, vol. 10, no. 1, pp. 19–41, 2000.

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

[51] John A Hartigan and Manchek A Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[52] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey V. Valtchev, and P. C. Woodland, *The HTK Book*, Cambridge University Engineering Department, 3.4 edition, 2006.

[53] Michael L. Seltzer and Alex Acero, "Separating speaker and environmental variability using factored transforms," in *Proc. of Interspeech*, 2011, pp. 1097–1100.

[54] Mark J. F. Gales, "The generation and use of regression class trees for MLLR adaptation," Tech. Rep. CUED/F–INFENG/TR263, Cambridge University, 1996.

[55] L. Uebel and P. Woodland, "Improvements in linear transform based speaker adaptation," in *Proc. of ICASSP*, 2001.

[56] L. Welling, H. Ney, and S. Kanthak, "Speaker adaptive modeling by vocal tract normalization," *IEEE Trans. on SAP*, vol. 10, no. 6, pp. 415–426, Sep. 2002.

[57] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 9, pp. 171–185, April 1995.

[58] John McDonough and Emilian Stoimenov, "An algorithm for fast composition with weighted finite–state transducers," in *Proc. of ASRU*, Kyoto, Japan, 2007.

[59] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa, "Reverberant speech recognition based on denoising autoencoder," *Proc. INTERSPEECH*, 2013.

[60] 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.

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

[62] Li Deng, Alex Acero, Mike Plumpe, and Xuedong Huang, "Large-vocabulary speech recognition under adverse acoustic environments.," in *INTERSPEECH*, 2000, pp. 806–809.

[63] Olli Viikki and Kari Laurila, “Cepstral domain segmental feature vector normalization for noise robust speech recognition,” *Speech Communication*, vol. 25, no. 1, pp. 133–147, 1998.

[64] H. K. Maganti and M. Matassoni, “An auditory based modulation spectral feature for reverberant speech recognition.,” in *Proc. INTERSPEECH*, 2010.

[65] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, “Auto-encoder bottleneck features using deep belief networks,” in *Proc. ICASSP*, 2012.

[66] O. Vinyals and S. V. Ravuri, “Comparing multilayer perceptron to deep belief network tandem features for robust asr,” in *Proc. ICASSP*, 2011.

[67] Xue Feng, Yaodong Zhang, and James Glass, “Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1759–1763.

[68] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proc. ICML*, 2008.

[69] Y. Bengio, “Learning deep architectures for ai,” *Foundations and trends in Machine Learning*, vol. 2, no. 1, 2009.

[70] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[71] Ralph Linsker, “An application of the principle of maximum information preservation to linear systems,” in *Advances in neural information processing systems*, 1989, pp. 186–194.

[72] Anthony J Bell and Terrence J Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[73] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[74] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[75] K. Vertanen, "Baseline WSJ acoustic models for HTK and Sphinx: Training recipes and recognition experiments," Tech. Rep., Technical report, University of Cambridge, 2006.

[76] Steve Young *et. al.*, "The HTK book (Version 3.2)," *Cambridge Univ. Press*, 2002.

[77] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni, "The second 'chime' speech separation and recognition challenge: An overview of challenge systems and outcomes," in *Proc. ASRU*, 2013.

[78] Xue Feng, Brigitte Richardson, Scott Amman, and James Glass, "On using heterogeneous data for vehicle-based speech recognition: A dnn-based approach," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4385–4389.

[79] Herve Bourlard and Nelson Morgan, "Connectionist speech recognition. a hybrid approach, vol. 247 of the kluwer international series in engineering and computer science," 1993.

[80] James L McClelland and Jeffrey L Elman, "The trace model of speech perception," *Cognitive psychology*, vol. 18, no. 1, pp. 1–86, 1986.

[81] Scott Axelrod, Vaibhava Goel, Ramesh Gopinath, Peder Olsen, and Karthik Visweswariah, "Discriminative estimation of subspace constrained gaussian mix-

ture models for speech recognition," *IEEE Trans. on ASLP*, vol. 15, no. 1, pp. 172–189, 2007.

[82] C. Liu, H. Jiang, and X. Li, "Discriminative training of CDHMMs for maximum relative separation margin," in *Proc. of ICASSP*, Philadelphia, Pennsylvania, USA, 2005.

[83] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *INTERSPEECH*, 2013, pp. 2345–2349.

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

[85] Michael L Seltzer, Dong Yu, and Yongqiang Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP*. IEEE, 2013, pp. 7398–7402.

[86] Arun Narayanan and DeLiang Wang, "Joint noise adaptive training for robust automatic speech recognition," *Proc. ICASSP*, 2014.

[87] Patrenahalli M. Narendra and Keinosuke Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on Computers*, vol. 9, no. C-26, pp. 917–922, 1977.

[88] Ramesh A Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," in *Proc. ICASSP*. IEEE, 1998, vol. 2, pp. 661–664.

[89] Spyros Matsoukas, Rich Schwartz, Hubert Jin, and Long Nguyen, "Practical implementations of speaker-adaptive training," in *DARPA Speech Recognition Workshop*. Citeseer, 1997.

[90] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *Proc. ASRU*, 2011, pp. 1–4.

[91] Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash, "Srilm at sixteen: Update and outlook," in *Proc. ASRU*, 2011, p. 5.

[92] Xue Feng, Brigitte Richardson, Scott Amman, and James Glass, "An environmental feature representation for robust speech recognition and for environment identification," in *Proc. of Interspeech*. IEEE, 2017, pp. 3078–3082.

[93] Najim Dehak, Reda Dehak, Patrick Kenny, Niko Brümmer, Pierre Ouellet, and Pierre Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification.," in *Interspeech*, 2009, vol. 9, pp. 1559–1562.

[94] Mohammed Senoussaoui, Patrick Kenny, Najim Dehak, and Pierre Dumouchel, "An i-vector extractor suitable for speaker recognition with both microphone and telephone speech.," in *Odyssey*, 2010, p. 6.

[95] Dong Yu and Michael L Seltzer, "Improved bottleneck features using pretrained deep neural networks.," in *Interspeech*, 2011, vol. 237, p. 240.

[96] Fred Richardson, Douglas Reynolds, and Najim Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[97] Patrick Cardinal, Najim Dehak, Yu Zhang, and James Glass, "Speaker adaptation using the i-vector technique for bottleneck features," in *Proceedings of Interspeech*, 2015, vol. 2015.

[98] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.