Feature Design for Robust Speech Recognition: Nurture and Nature

Shuo-Yiin Chang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-62
http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-62.html

May 12, 2016

Feature Design for Robust Speech Recognition: Nurture and Nature

by

Shuo-Yiin Chang

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nelson Morgan, Chair
Professor Jitendra Malik
Professor David Brillinger

Spring 2016

Feature Design for Robust Speech Recognition: Nurture and Nature

Abstract

Feature Design for Robust Speech Recognition: Nurture and Nature

by

Shuo-Yiin Chang

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Nelson Morgan, Chair

As has been extensively shown, acoustic features for speech recognition can be nurtured from training data using neural networks (DNN) with multiple hidden layers. Although a large body of research has shown these learned features are superior to standard front-ends, this superiority is usually demonstrated when the data used to learn the features is very similar to the data used to test recognition performance. However, realistic environments cover many unanticipated types of novel inputs including noise, channel distortion, reverberation, accented speech, speaking rate variation, overlapped speech, etc. A quantitative analysis using bootstrap sampling shows that these trained features are easily specialized to training data and corrupted in mismatched scenarios. Gabor filtered spectrograms, on the other hand, are generated from spectro-temporal filters to model natural human auditory processing, which can be instrumental in improving generalization to unanticipated deviations from what was seen in training. In this thesis, I used Gabor filtering as feature processing or a convolutional kernel in neural networks where the former used filter outputs as DNN inputs while the latter used filter coefficients and structures to initialize a convolutional neural network (CNN). Experiments show that the proposed features perform better than other noise-robust features that I have tried on several noisy corpora. In addition, I demonstrate that inclusion of Gabor filters with lower or higher temporal modulations could be used to correlate better with human perception of slow or rapid speech. Finally, I report on the analysis of human cortical signals to demonstrate the relative robustness of these signals to the mixed signal phenomenon in contrast to a DNN-based ASR system. With a number of example tasks in the thesis, I conclude that designed feature is useful for greater robustness than just relying on DNN or CNN.

DEDICATED TO:
TO MY FAMILY

Contents

Contents ii

List of Figures iv

List of Tables vi

1 Introduction 1
1.1 Typical ASR Framework 1
1.2 Challenges in Speech Technology 2
1.3 Robust Feature Design: Nurture and Nature 3
1.4 Thesis Overview 4

2 Deep Learning in Speech Recognition 6
2.1 Fully Connected Deep Neural Network 6
2.1.1 Single-hidden Layer MLP 7
2.1.2 Deep Neural Network with Pre-training 9
2.2 Neural Network in Speech Recognition 12
2.2.1 Hybrid HMM/DNN 12
2.2.2 Tandem 13
2.3 Convolutional Neural Network 15
2.3.1 Local Connectivity and Shared Weights 15
2.3.2 Maximum Pooling 16
2.3.3 CNN in Speech Recognition 17

3 Development of Gabor Convolutional Neural Network 19
3.1 Spectrum using Power Normalization 20
3.2 Spectro-Temporal Modulation Gabor Filter 22
3.2.1 Broader Motivation of Modulation Feature 22
3.2.2 Computational Gabor Model for Auditory Receptive Fields 23
3.3 Sparse Feature Selection 26
3.4 Gabor Convolutional Neural Network 28
3.5 Experimental Results 30
3.5.1 Noisy Speech Corpus 30
3.5.2 PN Spectrum and Gabor Processing 31
3.5.3 Sparse Gabor Features 32
3.5.4 Fully Connected DNN, CNN and GCNN 35
3.5.3 CNN Trained Filter versus Gabor Filter 36
3.6 Summary 37

4 Quantifying Neural Network Feature Errors and Model Errors 38
4.1 Model Errors and Observation Mismatches 39
4.2 Simulation and Bootstrap Sampling 40

4.2.1 Simulation Pseudo Observations 40
4.2.2 Frame Resampling 41
4.2.3 Segment Resampling 42
4.3 Data Preparation 43
4.3.1 Time-Aligning the Corpora 44
4.3.2 Data Partition 46
4.4 Acoustic Models and Features 46
4.5 Results and Discussions 47
4.5.1 Analysis of Matched Near-Field Results 48
4.5.2 Analysis of Matched Far-Field Results 48
4.5.3 Analysis of Mismatched Case 49
4.5.4 Analysis of Neural Network Weights 50
4.6 Summary 51

5 Feature Designs for Speaking Rate Variability 52
5.1 CVC Data Collection 52
5.2 CVC Recognizer 54
5.3 Multi-Stream Processing for Gabor Features 55
5.3.1 Temporal Division 55
5.3.2 Combination of Temporal Modulation Streams 56
5.4 Results and Discussions 57
5.5 Summary 59

6 Phone Recognition for Mixed Speech Signal Using Human Cortical Signal 60
6.1 Neural Data 60
6.2 Neural Feature Extraction 61
6.2.1 Neural Features from High-Gamma Band 63
6.2.2 Dimension Reduction 64
6.3 Phone Recognition System 65
6.4 Results 65
6.5 Summary 66

7 Conclusion 67
7.1 Contributions 67
7.2 Future Work 68

Bibliography 70

List of Figures

1.1 *Overview of statistical speech recognition*. 2

1.2 *Technology failure in current automatic speech recognition according to expert survey reported in [54]*. 3

2.1 *Neural network perceptron.* 8

2.2 *Single-hidden layer MLP.* 8

2.3 *Restricted Boltzman machine.* 10

2.4 *Hybrid HMM/DNN.* 13

2.5 *(a) Tandem system using posterior features.* 14

2.5 *(b) Tandem system using bottleneck features.* 15

2.6 *Neural network with shared weights and pooling.* 16

2.7 *Basic convolutional neural network topology with mel spectrum.* 17

3.1 *Comparison between mel spectrum (above) and PN spectrum (below).* 20

3.2 *Frequency response of Gammatone filters according to [38]* 21

3.3 *(a) Medium-duration power subtraction of PN spectrum.* 21

3.3 *(b) Clean and noisy mel spectrogram (left) and power normalized spectrogram (right).* 22

3.4 *Frequency response of Gabor filters*. 24

3.5 *Above: a high temporal modulation filter (temporal modulation frequency: 6.2 Hz, spectral modulation frequency: -0.03 cycle/channel) and corresponding filter output; below: a low temporal modulation filter (temporal modulation frequency: 2.4 Hz, spectral modulation frequency: 0.03 cycle/channel) and corresponding filter output*.....25

3.6 *Clean and noisy mel spectrum based (above) and power normalized spectrogram based gabor features (below)* 25

3.7 *Bottleneck deep neural network tandem system using sparse Gabor features* 27

3.8 *Basic convolutional neural network topology with PN-spectrum* 28

3.9 *Gabor convolutional neural network topology* 29

3.10 *Importance of Gabor filters based on sparse PCA* 34

3.11 *A vertical and a horizontal filter example from CNN* 37

3.12 *(left): Diagonal Gabor filter, (right): GCNN tuned diagonal gabor filter* 37

4.1 *Simulation process* 41

4.2 *Pseudo test data ("hi") generated using frame resampling* 42

4.3 *Pseudo test data ("hi") generated using state resampling* 43

4.4 *Pseudo test data (“hi”) generated using phone resampling* 43

4.5 *Time alignment: (a) near-field (blue) and far-field (green) signals (b) cross-correlation between the signals*. 45

4.6 *Discarded example: near-field (blue) and far-field (green) signals*.....45

4.7 *Initial weights and final weights of first hidden layer for MFCC input (a) and PNS-Gabor input (b)* 50

4.8 *Initial weight (black line) and final weight (red dashed line) learned from 2 hidden nodes for 9 frames of a MFCC (c1) (above: (a) and (b)) and a PNS-Gabor input (below: (c) and (d))* 51

5.1 *Accuracy for each consonant in clean condition* 53

5.2 *Word net for decoding CVC data*.....53

5.3 *Classification system for CVC data* 53

5.4 *Combination of neural network outputs using inverse entropy. Low temporal modulation streams (red) and high temporal modulation streams are split.* 57

6.1 *MRI reconstruction of subject's cortex, with electrocorticogram (ECog) electrodes (16x16 grid, 4 mm spacing) indicated by dots. The red and yellow regions (temporal lobe) are (temporal lobe) are expected to contribute most to speech processing.* 61

6.2 *Example of raw cortical signal from one electrode and the corresponding audio signal*.....62

6.3 *Neural feature generation process*.....62

6.4 *256-d high gamma feature* 63

6.5 *Spatial weightings for 4 out of 48 of the convex NMF neural features used. These are chosen for their loadings onto temporal lobe sites typical examples, chosen for their loadings onto temporal lobe sites important for speech important for speech processing.* 64

List of Tables

3.1 *Temporal and spectral modulation frequencies for Gabor function.* 24

3.2 *Comparison between Gabor-DNN, CNN and GCNN.*.....30

3.3 *Noisy channels added to WSJ.* 31

3.4 *Aurora2 WER of Gabor features with different spectro-temporal representations. The baseline is a 39-dimensional MFCC plus the first 2 derivatives with mean and variance normalization.* 32

3.5 *MFCC, AFE and PNCC baseline.* 33

3.6 *WERs for DNN features based on PNCC, PNS-Gabor and dimensional reduced PNS-Gabor using sparse PCA or classical PCA.* 34

3.7 *WERs for neural network features, clean training, noisy test.* 35

3.8 *WER for multi-condition training set*.....36

4.1 *Training and test statistics for both near-field and far-field set.*.....46

4.2 *Results for matched near-field data.*.....48

4.3 *Results for matched far-field data*.....49

4.4 *Results for mismatched scenario.* 50

5.1 *Consonant accuracy from the subjects for clean and noisy condition.* 53

5.2 *Temporal division of Gabor feature streams.*.....55

5.3 *Number of features for each spectral modulation frequency*.....55

5.4 *Front end effects for clean CVCs, all speech*.....57

5.5 *Front end effects for clean CVCs, slow speech*.....58

5.6 *Front end effects for clean CVCs, rapid speech.* 58

5.7 *Comparison between using low modulation filters and only the high ones for slow and rapid speech, using the GT-Gabor MLP front end, consonant accuracies.* 58

5.8 *Comparison between using low modulation filters and only the high ones for slow and rapid speech, using the GT-Gabor MLP front end, correlation with perception.*.....58

6.1 *Phone error rate for complete CRM utterances.* 66

6.2 *Phone error rates within the color target word (red, green, or blue).* 66

6.3 *Phone error rates within the number target word (two, five, or seven).* 66

Acknowledgement

When I look back over my Ph.D. career, I feel blessed to be a member of the speech group at International Computer Science Institute where I got a great many people helping me.

First, I would like to express my special appreciation and thanks to my advisor Nelson Morgan who have been a tremendous mentor for me. For the 5-year Ph.D. career, Morgan is very patient, encouraging and allowing me to grow as a research scientist. By working with him for four projects, I was always benefited with his view on either good or unexpected results. His intuition about the new techniques also allows me thinking out of mathematical formulations. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Second, I would like to thank to Steven Wegmann, the head of the Speech group at the International Computer Science Institute. Steve has a solid statistical background and experience on speech recognition system. His advice on both research as well as on my career have been priceless. The work benefited greatly from the discussions with Dan Ellis and Adam Janin. Their comments and suggestions are extremely helpful when I got stuck. Adam has the answers to all my questions. Thanks to Abeer Alwan and Edward Chang for sharing invaluable data and helpful advises. Thanks to Chanwoo Kim and Richard Stern for the use of PNCC. Thanks to Andreas Stolcke for the help on setting up language models. I would like to thanks to my coworkers, Erik Edwards, Bernd Meyer, Michael Kellman, Anirudh Raju, Hari Parthasarathi, Frantisek Grezl and Hai Do. They are very smart and nice to work with.

I would like to thank to Howard Lei, Dan Gillick and Arlo Faria who teach me a lot in my first year when I know little about the infrastructures and the tools at ICSI. Hang Su has been a wonderful friend and one of the smartest people in my life. He always catches up new techniques and tools very quickly. TJ Tsai always gave perfect presentations that is better than most of instructors from my studying. Suman Ravuri is the nicest guy ever and knows speech techniques broadly and deeply.

I would also like to thank my committee members, professor Jitendra Malik and professor David Brillinger for serving as my committee members even at hardship. I also want to thank for their brilliant comments and suggestions. Thank to all administrative staff and computer gurus who help to make ICSI a great place to develop interesting research. I would especially like to thank to Maria Eugenia Quintana who carefully read the thesis and make it easier to read. I would also thank to my master's advisor, Lin-Shan Lee who lead me to this fantastic research area.

A special thank to my family. I am grateful to my grandmother, grandfather, aunts and father for all of the sacrifices on my behalf. I would also thank to my friends Chi Pang Lam, Chung-Yen Lin, Junkai Lu and Yiwen Laio who gave me a lot of useful suggestions on my career and future plans.

This material is based on work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. D10PC20024, National Science Foundation under Grant No. IIS-1450916 and Grant IIS: 1320260.

Chapter 1

Introduction

Research on automatic speech recognition (ASR) flourished in the ‘70s and ‘80s when the hidden Markov model (HMM) [5] offered substantial improvements over other systems (e.g. dynamic time wrapping [55]). Since then, commercial versions of speech technology has been successfully used for mobile technology (e.g. voice dialing and interactive voice response), hands-free computing (e.g. voice search and video game with voice command), dictation products (real time speech writing), etc. Today, voice-activated digital assistants are an increasingly important feature for smart phones where high quality audio systems and substantial computing power are available. Accurate automatic speech recognition is also very useful for people who find it difficult to interact with computers using a keyboard: e.g., the elderly, the physically disabled, or the vision impaired.

While speech recognition works reasonably well in some environments, it often fail in difficult environments where the input may include noise, reverberation or overlapped speech. With the rapid growth of speech applications, robust processing is an important and challenging problem. Recently, robust speech processing leverages the expertise of machine learning techniques to nurture a more robust acoustic representation from a training phase. Modern machine learning techniques could potentially model a very broad distribution of all the input in the training set. However, many unanticipated types of novel input still appear because it is hard to enumerate all noise types, signal to noise ratios (SNRs), speaking rates and accents encountered in test environments. Given the difficulty of training, it is arguably necessary to work on technologies that directly deal with the noise, channel, and speaking style impacts. This thesis demonstrates this necessity using a number of example tasks that we have worked with.

1.1. Typical ASR Framework

Speech recognition is defined as the science of recovering words from an acoustic signal meant to convey those words to a human listener. Fig 1.1 illustrates an overview of the speech recognition framework, which involves acoustic processing, acoustic modeling, pronunciation modeling, language modeling and decoding. Conventional speech recognition is formulated as a pattern classification problem using the *maximum a posterior* decision rule to find the best word sequence W^* based on the parameterized observations O, which is transformed from the speech waveform:

$$W^* = \text{argmax}_{w \in W} P(W \mid O) \tag{1.1}$$

The system could be factorized into several smaller models:

$$W^* = \text{argmax}_{w \in W} P(O \mid Q) P(Q \mid W) P(W) \tag{1.2}$$

where Q is a sequence of phones: [$phone_1$, $phone_2$...., $phone_n$]

The prior probability of words $P(W)$ is evaluated with a language model. $P(O|Q)$ is the likelihood of observing O by assuming Q as the underlying phone sequence. $P(Q|W)$ is obtained from the pronunciation model that estimates the pronunciations given a sequence of words, which is typically specified manually. While more sophisticated language models [49] and pronunciation models [12] have been studied, here we are focusing on acoustic features and models in the thesis.

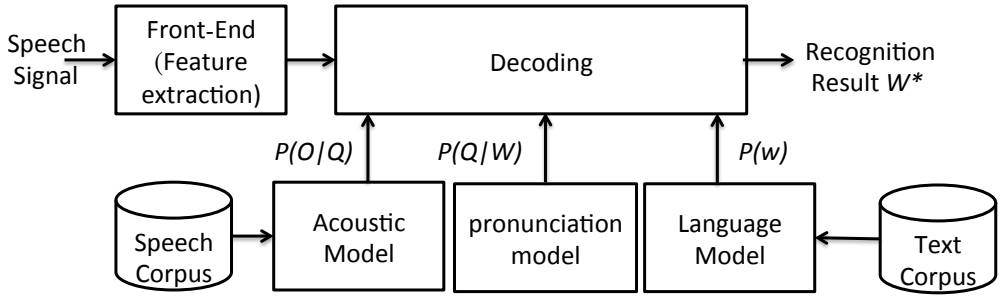

Figure 1.1, Overview of statistical speech recognition

1.2. Challenges in Speech Technology

Speech recognition technology has decades of history with several major innovations. While there is a great deal of progress and commercial applications, automatic speech recognition is far from being a solved problem. In particular, automatic speech recognition is much worse than human recognition in noisy environments, for novel speakers, in far-field or other unusual acoustic conditions, in accented speech, and for speech in which other signals or noises share the acoustic channel.

In a recent survey, Fig. 1.2 [54], major participants in speech and language technology were asked to identify where the current technology has failed. While several components in the framework were pointed out, many of the informants identified the lack of robustness as a primary failure of speech and language technology. In addition to the general robustness issue, many responses identified the particular characteristic of speech or language that caused this lack of robustness, such as noise and variability in the speaker population.

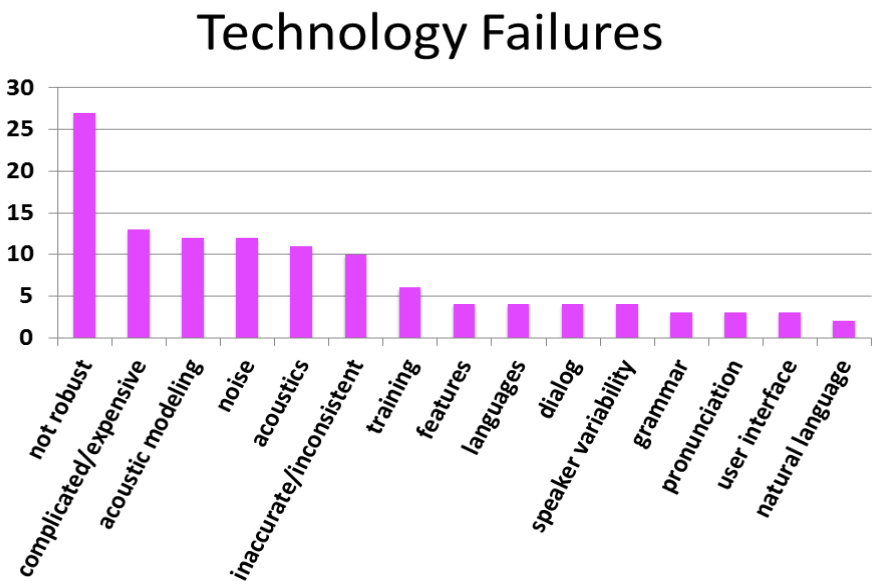

Figure 1.2, Technology failures in current automatic speech recognition according to expert survey reported in [54]

1.3. Robust Feature Design: Nurture and Nature

A large body of research on robust speech processing focuses on generating a more robust feature representation. Traditional robust feature design is based on the expertise of the mammalian auditory system, prior knowledge of acoustic distortion, normalization approaches, etc. For example, many acoustic features are obtained from filters mimicking "natural" hearing systems in order to duplicate the robust properties of human speech recognition (e.g., cocktail party effect, word recognition in noise, insensitivity to a wide range of speaking rates, etc.). Typically, these features are carefully hand-crafted leading to state-of-the-art speech recognition results.

Exploiting machine learning algorithm to "train" features from data provides an alternative solution to the difficult task of feature handcrafting. In this case, the features are "nurtured" automatically from the training set. For example, the probabilistic feature (class probabilities) of a neural network is trained to discriminate between the phone classes. These trained features could represent underlying acoustic cues well if the classification is reliable. With the advent of new training algorithms and model architectures, trained features could model more complicated distributions from the input, which led to a recent breakthrough in speech community.

Recently, because of the success of deep learning, neural network features have become a dominant paradigm for speech recognition. In [53], fewer perceptually based designs are suggested for better performance. In [31], a neural network is able to achieve comparable performance by generating feature directly from speech signals, which requires no human-inspired design on the feature generation process. In these studies,

most of the work of feature extraction has been done in neural networks. These studies suggest questions about the role of feature representations beyond that.

Although a large amount of research has shown that acoustic features nurtured from data using neural networks can be superior to a standard front-end, this superiority is usually demonstrated when the data used to learn the features is very similar to the data used to test recognition performance. A speech recognizer may be robust in one environment and yet be inappropriate for another. The main reason for this is that the performance of existing recognition systems, which assume noise-free environment degrade rapidly in the presence of noise, distortion, etc. On the other hand, carefully hand-crafted feature designs develop robust features based on auditory processing provide a more general solution than trained features alone.

In the thesis, we incorporate biological model-inspired features into neural networks by adapting an existing architecture. We investigate the robustness of the proposed approaches with several input variations including noise, reverberation, speaking rates and overlapped speech. The goal of the thesis is to facilitate the development of robust speech recognition in two areas: (1) development of a robust representation that actually improves recognition accuracy, and (2) analysis of robustness benefited from biological models. We also explore these issues:

1. How can a speech feature benefit from a biological model under (a) noise, (b) reverberation and (c) speaking rate variation?

2. Given the different feature designs, what is the quantitative effect of the model residual on recognition accuracy?

3. What can we learn directly from a human cortical signal, which can handle multiple source signals well?

1.4. Thesis Overview

The thesis is organized as follows:

Chapter 2 is an introduction to (deep) neural networks and their application to automatic speech recognition.

Chapter 3 proposes an architecture that incorporates biologically inspired Gabor features and (convolutional) neural network. The final feature, which benefited from human auditory processing, is more robust to other trained features.

Chapter 4 performs a quantitative analysis on model residuals and robustness of the proposed features and conventional trained features for noisy speech recognition.

Chapter 5 explores the effect of feature design to speaking rate variation.

Chapter 6 investigates the robust property of human cortical signals for phone recognition on mixed speech signals.

And Chapter 7 summarizes the conclusions of the work, and points to future work.

Chapter 2

Deep Learning in Speech Recognition

Multi-layer perceptrons (MLP) have been used successfully for HMM-based speech recognition for more than two decades [9]. In that approach, MLP outputs were used as posteriors to derive emission probabilities for hidden Markov models (HMMs). Later, a number of researchers (e.g., [Hermansky et al.]) made use of MLP outputs as features for HMM observations (tandem) [28, 79]. Both of these approaches have been used in more recent "deep learning" methods that have been designed to effectively incorporate a larger number of layers, and in particular have been successfully applied to automatic speech recognition (ASR).

There has been a lot of success using deep learning in the machine learning community, with many applications to difficult problems, for example in computer vision. There are two major components to deep learning: one is the architecture of the MLP and the other involves the training algorithm. First of all, in a deep learning architecture, MLPs with more than one hidden layer are used. Second, since MLPs with more than one hidden layer are hard to train, the learning algorithm involves better initialization than simple back-propagation. To provide a better initialization, a generative model, called a restricted Boltzmann machine (RBM), has been used with an unsupervised learning algorithm, called pre-training [29], to build up a multi-layer network, called a deep belief network (DBN). This DBN is then converted to an MLP, called a deep neural network (DNN), and final passes of back-propagation are used for "fine-tuning" the weights. In additional to RBM, other pre-training approaches, for example, layer-by-layer discriminative pre-training technique with back-propagation [75], are also commonly used.

Beyond the traditional fully connected DNNs, convolutional neural networks (CNN) has been successfully introduced for speech recognition task. Unlike typical fully connected networks, CNN is more robust to translation variance, which is desirable for object recognition and robust speech recognition.

2.1. Fully-Connected Deep Neural Network

The single-hidden layer MLP has been used for several decades. While some researchers (e.g. Chen et al [78]) use two hidden layers to achieve better results, the "depth" is still

much smaller than a typical deep neural network structure. The challenges against moving the single-hidden layer MLP toward large scaled deep networks are (1) the cost of training and (2) poor local minima in the highly nonlinear optimization model. The computational capacity has been benefited from the parallel, distributed tools e.g., GPU and MapReduce. In parallel computing, data parallelism and model parallelism are two commonly used approaches to speed up the training procedure. The former splits training samples into min-batches in each thread and updates the gradients after each min-batch iteration. The latter splits the weights among the threads and synchronizes their outputs. Another issue is the non-convex optimization problem. While an increasing number of hidden layers allows it to represent complicated transformations, it is highly nonlinear and the stochastic gradient decent (SGD) based training algorithm with random initialization is easily converged into poor local minimum. In DNN, a training algorithm based on restricted Boltzman machine is sometimes used to provide a better initialization. In the remainder of the section, we will first introduce the basic mechanism of MLP and then DNN using RBM pre-training.

2.1.1. Single-hidden Layer MLP

MLP consists of basic processing units called perceptrons. Perceptrons were developed in 1950s and 1960s by Frank Rosenblatt [66, 67]. Fig. 2.1 shows an example of a perceptron where the inputs are x_1, x_2 … x_n. The weights, w_1, w_2, … w_n are used to express the importance of the respective input to the output. The activation function $\theta(.)$ is used to model neuron firing activity. The widely used activation function is hyperbolic tangent:

$$y = \tanh(\sum_i w_i x_i + b_i) = \frac{2}{1+\exp(\sum_i w_i x_i + b_i)} - 1 \tag{2.1}$$

and sigmoid function:

$$y = sig(\sum_i w_i x_i + b_i) = \frac{1}{1+\exp(-\sum_i w_i x_i - b_i)} \tag{2.2}$$

As shown in the equation 2.2, a perceptron with sigmoid activation can be viewed as a logistic regression classifier, called a logistic unit.

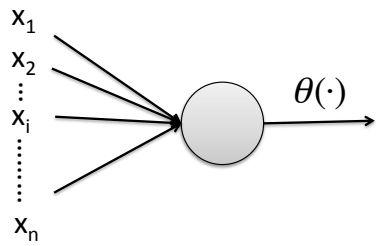

Figure 2.1, Neural network perceptron

More recently, a rectified linear unit (ReLU) based neural network has been introduced [57] where the activation function is a half-rectification non-linearity which is linear for positive values and zero otherwise:

$$y = \max(0, \sum_i w_i x_i + b_i) \tag{2.3}$$

It has been observed that the rectified linear unit is better than a logistic unit or hyperbolic tangent in vision and other applications. A rectified linear unit is better than a logistic unit using sigmoid activation in several aspects. First of all, rectified linear unit is piece-wise linear. If we focus our attention to the units that are non-zero, the whole system reduces to a linear convex system whose optimization is straightforward even using first order optimizers. Second, the activated value is sparser. Unlike logistic units that produce small positive values when the input is not aligned with the internal weights, rectified linear units often output exact zeros, for example, in a randomly initialized network, only about 50% of hidden units are activated. The increased sparsity of the internal representation can be seen as the effect of regularization, which improves the generalization. Third, traditional activation functions have gradients less than one, meaning the gradients decrease exponentially with number of layers and early layers train very slowly. The problem could also be avoided using rectified linear units.

A single-hidden-layer MLP consists of 3 layers (i.e. input, hidden and output layer) of neurons with each layer fully connected to the next one. Except for the input layer, each neuron is the perceptron described above. Fig. 2.2 illustrates an example of MLP.

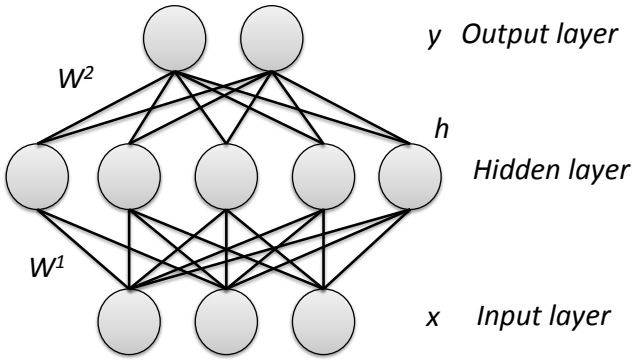

Figure 2.2, Single-hidden layer MLP

The parameters of MLP are two matrices $W^1$and W^2 where hidden units h is computed by $\theta(W^1 x)$ while the output units y is $\theta(W^2 h)$. The cost function is typically defined as mean square error:

$$E(n) = \frac{1}{K}\sum_{k=1}^{K}\left[t_k(n) - y_k(n)\right]^2 \tag{2.4}$$

or cross entropy error:

$$E(n) = \frac{1}{K}\sum_{k=1}^{K}\left\{t_k(n)\ln y_k(n) + \left[(1-t_k(n)\right]\ln\left[(1-y_k(n)\right]\right\} \tag{2.5}$$

where t_k is target value and y_k is actual neural network output. In practice, the cross entropy error criterion is more common than mean square error. By using cross entropy error, the error signal propagating back from output units is directly proposition to the difference of target value and t_k and the actual value y_k , which lead to a faster convergence and better performance. To minimize the cost function, the standard gradient decent algorithm is applied to determine the weights:

$$w_{ij}^1(n+1) = w_{ij}^1(n) + \mu\frac{\partial E(n)}{\partial w_{ij}^1(n)} \tag{2.6}$$

$$w_{ij}^2(n+1) = w_{ij}^2(n) + \mu\frac{\partial E(n)}{\partial w_{ij}^2(n)} \tag{2.7}$$

2.1.2. Deep Neural Network with Pre-training

While back-propagation theoretically allows training a MLP with many layers, researchers did not have widespread success training MLPs with more than one hidden layers. It is difficult to benefit from increasing the number of hidden layers without using some kind of initialization, particularly because the layers far from the target are little changed by the usual stochastic gradient learning algorithms (i.e. vanishing gradient problem). Hinton et al [29] proposed an unsupervised training algorithm based on restricted Boltzman machine moving the parameters to a good initial. RBMs are trained in one more layer at a time in a greedy manner and then stacked to build up a hierarchy multi-layer network, so-called deep belief networks.

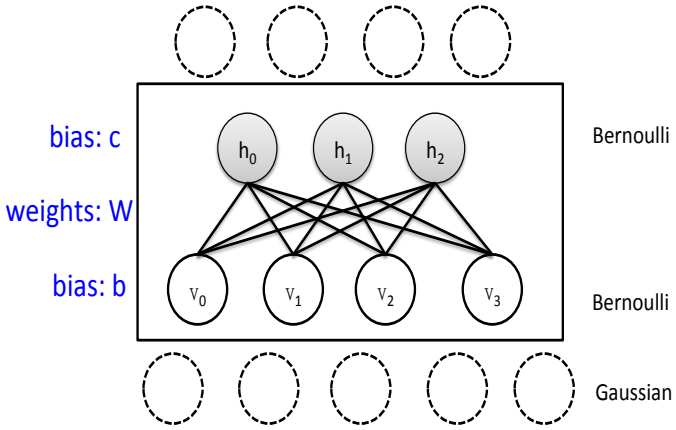

Figure 2.3, restricted Boltzman machine

RBM is a bipartite graph modeling the joint distribution of a layer of stochastic visible units and hidden units as shown in Fig. 2.3. Only visible-hidden connections are allowed. For Bernoulli-Bernoulli RBM, the joint distribution is defined as:

$$P(v,h) = \frac{e^{-E(v,h)}}{Z}, E(v,h) = -b^T v - c^T h - v^T W h \tag{2.8}$$

where Z is the normalization term, $E(v,h)$ is energy function. b and c are the bias for visible layer v and hidden layer h while W is a symmetric weight matrix between v and h. To compute the posterior of hidden units, it can be obtained as:

$$P(h \mid v) = \frac{P(v,h)}{P(v)} = \frac{e^{-E(v,h)}}{\sum_{h'} e^{-E(v,h')}} \tag{2.9}$$

By plugging in the energy function $E(v,h)$, we could obtain:

$$P(h \mid v) = \frac{e^{b^T v + c^T h + v^T W h}}{\sum_{h'} e^{b^T v + c^T h' + v^T W h'}} = \frac{e^{c^T h + v^T W h}}{\sum_{h'} e^{c^T h' + v^T W h'}} \tag{2.10}$$

,which is the multiplication of the posterior of each unit:

$$P(h \mid v) = \prod_i \frac{e^{c_i h_i + v^T W_i h_i}}{\sum_{h_i'} e^{c_i h_i' + v^T W_i h_i'}} = \prod_i P(h_i \mid v) \tag{2.11}$$

Let h_i be one, we can rewrite the posterior as:

$$P(h_i = 1 \mid v) = \frac{e^{c_i + v^T W_i}}{e^{c_i + v^T W_i} + 1} = sigmoid(c_i + v^T W_i) \tag{2.12}$$

Formula (2.12) is essentially the forward propagation procedure of neural networks. Hence, RBM can be applied to model each pair of layers of a neural network appropriately. Similarly, we could compute $P(v|h)$ by symmetric:

$$P(v_i = 1 \mid h) = sigmoid(b_i + h^T W_i) \tag{2.13}$$

Unlike the Bernoulli-Bernoulli distributed RBM, the input layer usually consists of real-valued variables, so a Gaussian-Bernoulli RBM is employed. The energy function for this is revised as:

$$Energy = \frac{1}{2}(v - b)^T (v - b) - c^T h - v^T W h \tag{2.14}$$

By replacing the energy function as defined above, we still obtain the posterior of hidden units as:

$$P(h_i = 1 \mid v) = sigmoid(c_i + v^T W_i) \tag{2.15}$$

and the posterior of visible units is:

$$P(v_i \mid h) = N(v_i; b + h^T W^T, I) \tag{2.16}$$

The parameters of the neural network, therefore, get good initialization by maximizing the log likelihood of the RBM where the objective function is:

$$\hat{c}, \hat{b}, \hat{W} = \arg\max_{c,b,W} P(v \mid c, b, W) = \arg\max_{c,b,W} \sum_h P(v, h \mid c, b, W) \tag{2.17}$$

The detail of RBM training can be found in [29].

A deep belief network is built by stacking RBMs that are trained in a greedy layer-wise manner. The first hidden layer is trained using RBM with raw input as visible units. After training is done, the first hidden layer is viewed as one visible layer for another RBM with another hidden layer on top of that. We repeat the process to increase the number of layers and build the network with multiple hidden layers. The deep belief network is then fine tuned using back-propagation training as the conventional MLP training. The final network is called a deep neural network (DNN).

While unsupervised pre-training using RBM was historically the turning point for deep neural networks, RBM training doesn't have a good stop criterion, and the training procedure can be expensive with the increase of the number of layers. Alternatively, layer-by-layer discriminative pre-training technique is presented in [75]. In general, a one-hidden-layer neural network is trained first using labels discriminatively with error back-propagation. Then, after discarding an output layer in the previous one-hidden-layer neural network, another randomly initialized hidden layer is added on top of the previously trained hidden layer along with a new output layer that represents the targets for classification or recognition. The resulting multiple-hidden-layer DNN is then discriminatively trained using the same strategy, and so on until the desired number of hidden layers is reached. Other approaches have been developed [51], but their explanation is beyond the scope of this thesis.

2.2. Neural Network in Speech Recognition

In this section, we introduce two of the most common approaches that incorporate neural networks in the HMM based acoustic model: the hybrid and tandem systems. The approaches were first proposed in the 1990's and 2000 using MLPs and extended to deep neural networks in recent years. The more recently developed neural network end-to-end (HMM free) system was not included in the section, but is described in [23].

2.2.1. Hybrid HMM/DNN

Research on hybrid neural network HMM based automatic speech recognition started since 1990's. The basic idea was to replace GMM for the hidden states's marginal distributions in the standard HMM system with a single neural network. In typical processing, the neural network used the softmax function for the output non-linearities, while the inputs to the network were a context window of adjacent MFCC or PLP frames.

The neural network was used as a classifier that discriminated between all of the HMM's hidden states. Earlier work [8] had shown that the output of such an neural network could be interpreted as the posterior probabilities of the states given the inputs, which meant that dividing the posterior probabilities by the state priors yielded scaled likelihoods that were suitable to use for the state's marginal or output probabilities. For the training scheme, instead of using only frame-level DNN training independent of a HMM and language model, some approaches jointly optimize DNN weights, HMM state-

to-state transition parameters, and language model scores using the sequential discriminative training criterion [51].

There are some advantages that distinguish a hybrid HMM/DNN over the standard HMM/GMMs. First of all, neural networks are inherently discriminative, although at the state rather than the word or utterance level. Second, a neural network has fewer assumptions about the data. Third, using a context window as the inputs to the neural network captures the long term temporal information.

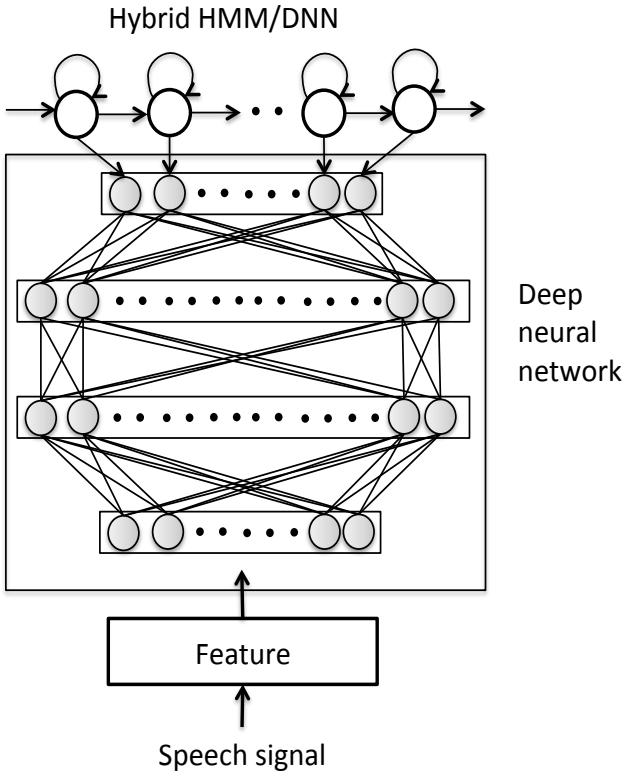

Figure 2.4, Hybrid HMM/DNN

2.2.2. Tandem

Starting in 2000, researchers have investigated features that were learned from data using neural networks. In the tandem approach, these neural network based features were then used as extra acoustic features for the standard HMM/GMM (Fig. 2.5(a)). One motivation for the tandem approach was to take advantage of the discriminant property from neural networks by incorporating the resulting discriminative features into HMM/GMM-based systems. Another advantage to the tandem approach is that it makes it easier to include non-traditional features within the HMM/GMM; for example TRAPS [27] use narrow spectral subbands and long temporal windows as inputs to multiple

MLPs whose outputs are combined using a MLP. A variant of TRAPS, called HATS [11], led to a roughly 10% relative improvement in recognition accuracy.

Again, the input layer of a neural network is a context window of adjacent frames of features (e.g., MFCC or PLP) and the softmax outputs discriminate between context independent phones. The frame-level labels for training are produced using forced alignment in HMM/GMM system. To apply the posteriors from neural network outputs as acoustic features, a post-processing step is preferred. In typical tandem processing, the posterior phone probabilities are processed by logarithm and principle component analysis (PCA) to yield features that could be modeled by Gaussian mixtures appropriately. The logarithm transformation makes the distribution more Gaussian-like, and PCA performs de-correlation and dimension reduction for the diagonal Gaussian distributions. In conventional fashion, they are appended to the standard feature set, such as MFCC or PLP.

One variant of tandem processing is taking the raw hidden layer outputs instead of using the posteriors as features, as shown in Fig. 2.5(b). In this case, the hidden layer is often designed as a narrow-dimensional layer, called a bottleneck layer e.g., in [24] a five-layer network is used with a constriction in the middle layer and the (processed) output of the constricted layer is taken as the feature vector. The motivation is that these hidden layer representations should be a good feature for HMM in the way they were used for classification in neural network. There are several advantages to using bottleneck features rather than posteriors. Because the feature size is not determined by output layer, neural networks could be trained to discriminate context dependent tied tri-phone states. Also, the logarithm and PCA are not necessary as the bottleneck feature can be viewed as nonlinear dimensional reduction.

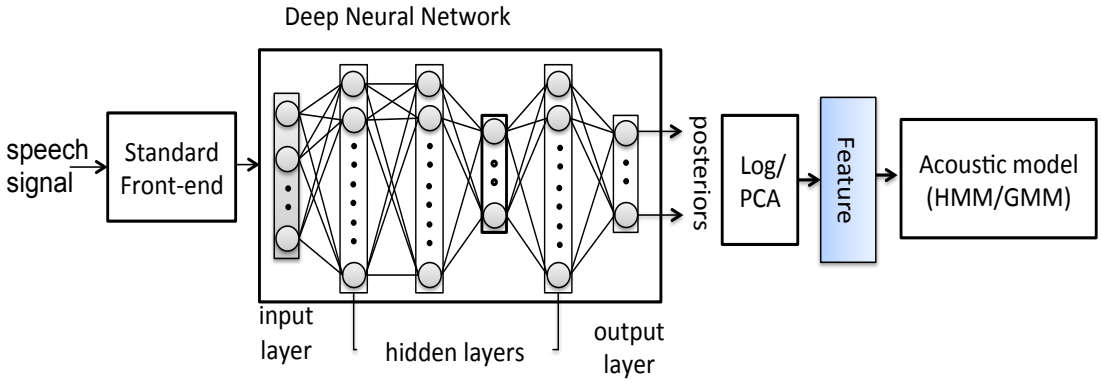

Figure 2.5(a), Tandem system using posterior features

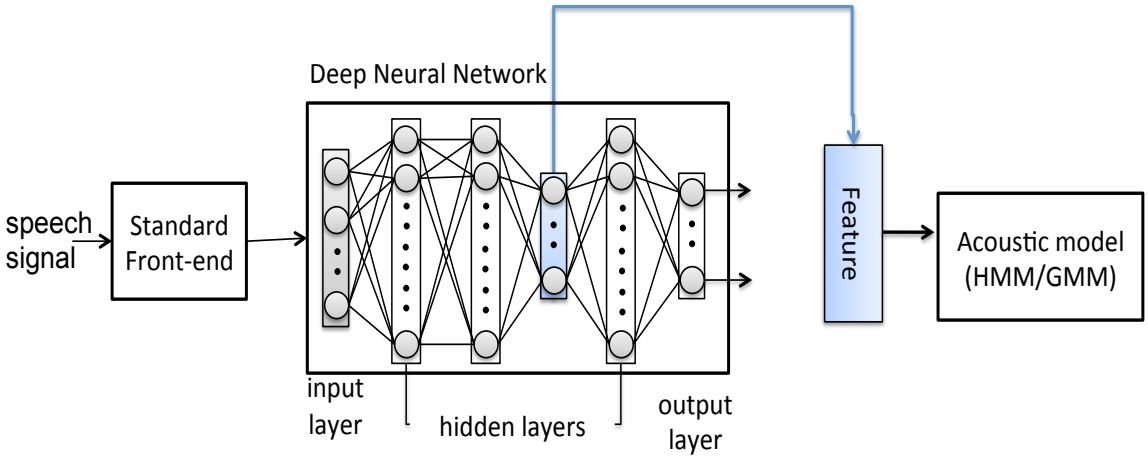

Figure 2.5(b), Tandem system using bottleneck features

2.3. Convolutional Neural Network

A convolutional neural network is a biologically inspired variant of a deep neural network. A common CNN topology was proposed by LeCun et al. [40] and has garnered wide attention with the success of deep learning. It has been successfully applied in digit recognition and object recognition. Later, Sainath et al. [68] applied a similar concept to speech recognition. The keys concepts that distinguish CNN from a typical fully connected neural network are local connectivity, shared weights and pooling.

2.3.1. Local Connectivity and Shared Weights

Fig. 2.6 depicts a CNN where the hidden units are connected to only a small, localized region of the input instead of the entire field. In the example, each hidden activation h_i is computed by multiplying local input $([x_i, x_{i+1}, x_{i+2}])$ against the weights W. The weights W are shared across the entire input space, as indicated in the figure. In other words, the hidden layer h is obtained by convolution of input x with a linear filter W, adding a bias term and applying an activation function. The function of the hidden layer is, therefore, often called convolution.

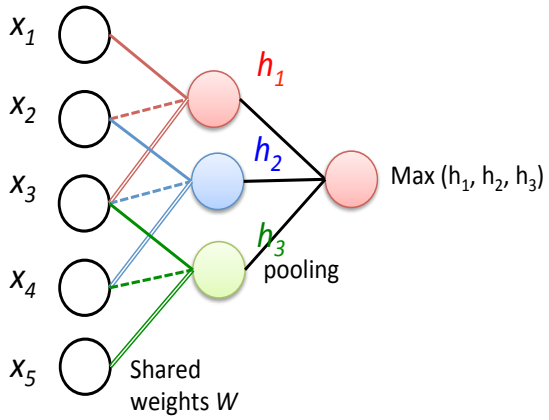

Figure 2.6, Neural network with shared weights and pooling

The local connectivity is inspired by biological evidence where a number of experiments on different mammal species have revealed that sensory neurons are sensitive to a particular sensory space, a so-called receptive field. For example, Hubel and Wiesel's early work on the cat's visual cortex shows that the cells in the visual cortex are sensitive to small sub-regions of the visual field. These sub-regions are tiled to cover the entire field. Thus, by connecting to a small receptive field and weight-tying, CNN learns local structure of the input (e.g. image or spectrogram of speech), which (1) models the local receptive field in human sensory neurons and (2) is more computationally efficient to scale up well.

2.3.2. Maximum Pooling

Another important concept of CNNs is max pooling, which is a form of down-sampling. After computing the hidden units, a max pooling layer helps to remove variability in the hidden units (i.e. convolutional activations) from the upper layer. Max pooling partitions the input into a set of overlapping or non-overlapping units and, for each such sub-region, outputs the maximum value. In the example of Fig. 2.6, each max-pooling unit receives 3 activations from the convolutional layer, and outputs the maximum of the activations from these activations. Convolution and max pooling over neighboring units allows translational invariance to the input. For example, a single shift from the input would lead to invariant convolution activations with a single shift. As the max pooling is done over a window of 3, it will produce exactly the same output. Since it provides additional robustness to minor difference of positions, CNN can be better for vision and robust speech recognition tasks where the robustness to object translation and speaker/channel variability is desirable.

Typically, CNN consists of one or more convolution/max pooling layer pairs and fully connected layers on top. The top fully connected layers finally combine inputs from all the sub-regions to do the classification of the overall inputs. This hierarchical organization generates good results in image and speech processing tasks.

2.3.3. CNN in Speech Recognition

The typical convolutional neural network for speech recognition using Mel spectrum was shown in Fig. 2.7, which consists of a convolutional layer, a pooling layer, and fully connected layers.

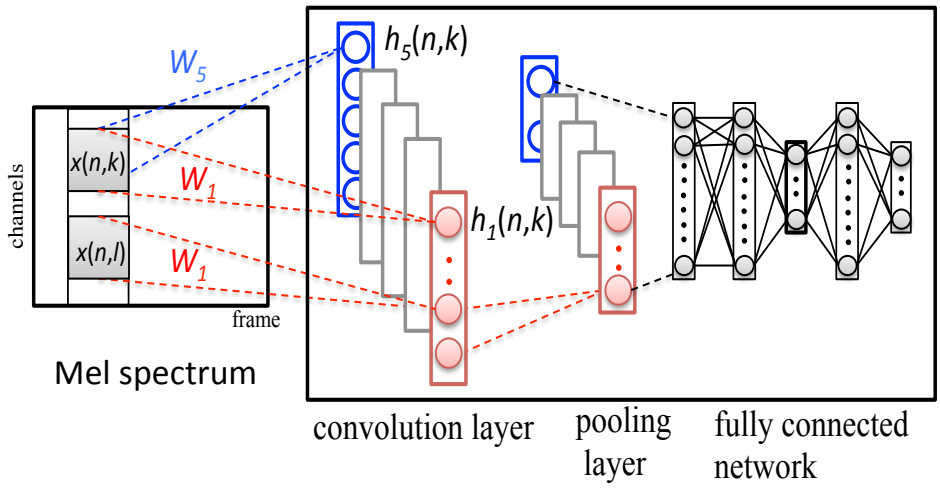

Figure 2.7: Basic convolutional neural network topology with Mel spectrum

Each CNN training case consists of entire frequency bands of successive frames. In the convolutional layer, the receptive field of each neuron is connected to a local subset of frequency bands. A set of neurons with receptive fields shifted in frequency share the same kernel (weights). Stacks of these neurons that cover features of the kernel along entire frequency bands constitute a feature vector. The convolutional layer is composed of multiple feature vectors determined by different kernels, as depicted in Fig. 2.7. Activation of each neuron is computed by multiplication of a local receptive field with the weights, adding a bias and applying a nonlinear function:

$$\begin{aligned} h_m(n,k) &= \theta\Big(\sum_{i=-N}^{N}\sum_{j=-K}^{K} W_m(i,j)\cdot x(n+i,k+j)+b_m\Big) \\ &= \theta(W_m(-n,-k) * x(n,k)+b_m) \end{aligned} \tag{2.18}$$

where $h_m(n,k)$ represents the neuron of m^{th} feature vector, whose receptive field is $2K+1$ (bands) by $2N+1$ (frames) matrix centered at current band of the frame, $x(n,k)$. The connection weights W_m perform filtering on the receptive field where the indices of the filter coefficients are flipped from the weight indices, both vertically and horizontally as shown in Eq. 3.14. b_m and $\theta(.)$ are the bias term and sigmoid function respectively.

In the above example, convolution was applied along the frequency axis. the CNN architecture can also be applied along the time axis to reduce temporal variability, which was known as time-delay neural network (TDNN) in [74]. In [1], the network combines convolution along both frequency and time axes to generate a 2D CNN similar to the

ones used for image analysis. While both case offered limited performance improvement, perhaps it is because the speech temporal variability has been handled by the HMM in either the tandem or hybrid framework.

Chapter 3

Development of Gabor Convolutional Neural Network

Although a large body of research has shown that acoustic features for speech recognition can be nurtured from data using neural networks with multiple hidden layers and that these learned features are superior to standard features (e.g., MFCCs), this superiority is usually demonstrated when the data used to learn the features is very similar in character to the data used to test recognition performance. An open question is how well these learned features generalize to realistic data that is different in character to their training data; in particular the robustness to unexpected noise environments is highly desirable.

Several existing robustness methods focus on compensating the difference between clean training data and noisy testing speech in different aspects such as model-based [37] or feature-based approaches [50, 38, 62]. Model based approaches focus on compensating the difference of clean and noisy speech by adapting a recognition model into a noisy speaker condition. For feature-based approaches, signal processing techniques are proposed to suppress the impact of noise or distortion prior to feature extraction or generate feature parameters, which are less sensitive to noise or distortion. Alternately, unlike ASR systems, human listeners rely on attention-driven (cognitive) selection of a specific speaker, e.g., in a high-noise cocktail party situation, which results in high recognition scores for human listeners, and which inspires researchers to find more robust features based on biological models about the auditory system.

In this chapter, we describe the development of robust feature representation based on Gabor filters that mimics “natural” (human) auditory processing. The filter design is then incorporated into fully connected deep neural networks or convolutional neural networks in several different ways to improve the robustness of automatic speech recognition. From our results, we conclude that signal processing technqiues based on prior knowledge of auditory models can improve the noise-robust speech recognition significantly while the perfromance of a deep neural network based on a standard front-end could degrade rapidly in the presence of noise.

3.1. Spectrum using Power Normalization

In [38], Kim and Stern propose the power normalized cepstral coefficient (PNCC) algorithm using gammatone filters followed by power bias subtraction and power nonlinearity compression. PNCC is relatively insensitive to constant background level noise. Therefore, spectra generated by the PNCC algorithm could be a better choice as the time frequency representation than Mel spectrum. We refer to the spectra generated from PNCC as the power normalized spectrum (PNS). The comparsion between MFCC and PNCC is depicted in Fig. 3.1.

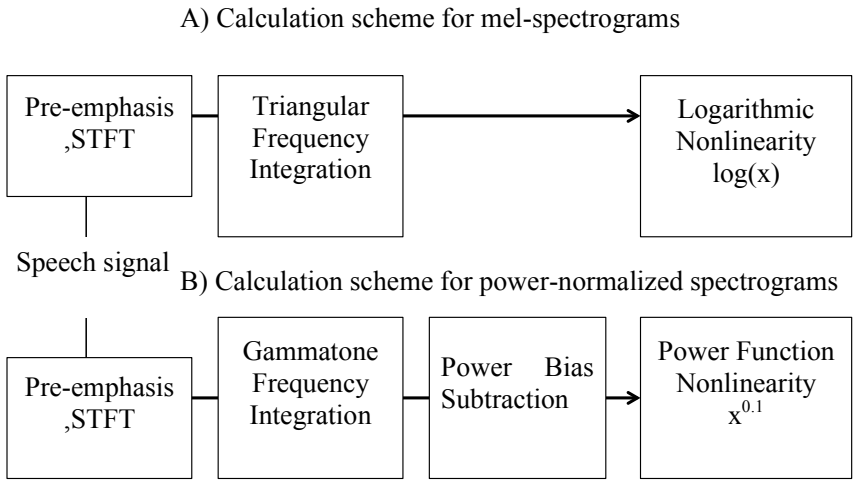

Figure 3.1, Comparison between Mel spectrum (above) and PN spectrum (below)

As in the case of MFCC, a pre-emphasis filter of the form $H(z) = 1–0.97z–1$ is applied. A short-time Fourier transform (STFT) is performed using Hamming windows of duration 25.6 ms, with 10 ms between frames, using a DFT size of 1024. Spectral power in 40 analysis bands is obtained by weighting the magnitude-squared STFT outputs for positive frequencies by the frequency response associated with a 40-channel gammatone-shaped filter bank whose center frequencies are linearly spaced in Equivalent Rectangular Bandwidth (ERB) between 200 Hz and 8000 Hz, using the implementation of gammatone filters in Slaney's Auditory Toolbox [70] (Fig. 3.2), which is derived from psychophysical observations of the auditory periphery, i.e., the filter bank represents a model of cochlear filtering.

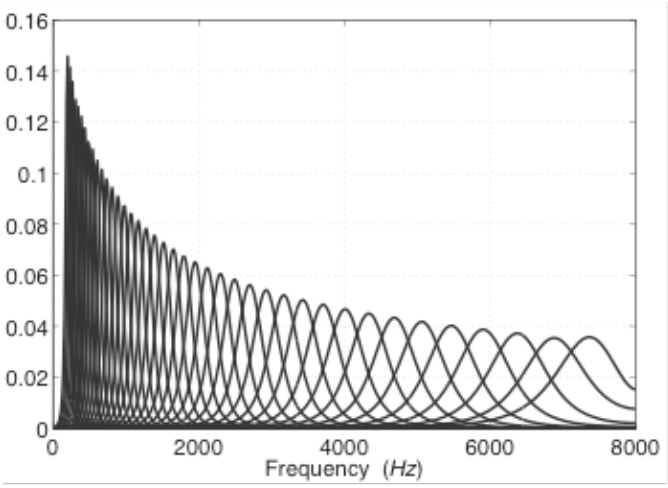

Figure 3.2, Frequency response of gammatone filter according to [38]

For the second step, medium duration power is used to estimate and subtract the noise background level. It is commonly observed that longer analysis windows provide better performance for noise modeling and environment analysis as the power associated with most background noisy conditions changes more slowly than the instantaneous power associated with speech. Thus, a medium-duration power obtained by computing the running average of the power of consecutive analysis frames is used to estimate the noise bias in this processing. The noise bias level was calculated based on the ratio of arithmetic mean and geometric mean (AM-to-GM ratio) of the medium duration power, which is motivated by a decrease of the noise power for a decreasing AM-to-GM ratio. In this step, a subtraction of the medium-duration power bias is carried out, which makes AM-to-GM ratio the same as that of clean speech.

Finally, power nonlinearity with an exponent of 0.1 replaces the logarithm nonlinearity for compression. The output of the logarithm would be dominated by noise when the intensity of the input signal is low, thus the power nonlinearity is a better model for threshold effects of auditory fire rate responses. According to the observation in [76], the auditory nerve firing rate is constant when the input sound pressure level is below -10 dB, while the output of the logarithm would be dominated by noise when the intensity of the input signal is low. An example of Mel spectrum and power normalized spectrum is shown in Fig. 3.3 illustrating greater insensitivity to noise for the power normalized spectrogram.

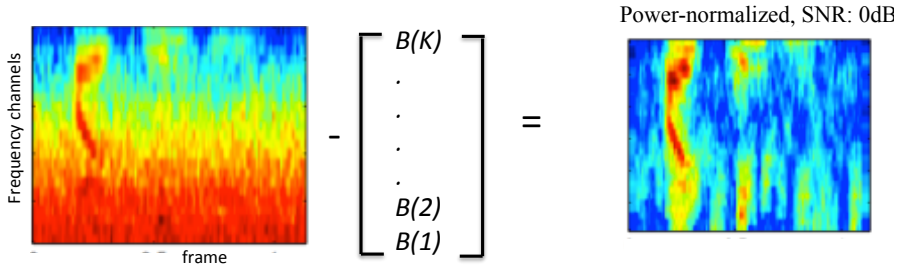

Figure 3.3 (a), Meduim-duration power subtraction of PN spectrum

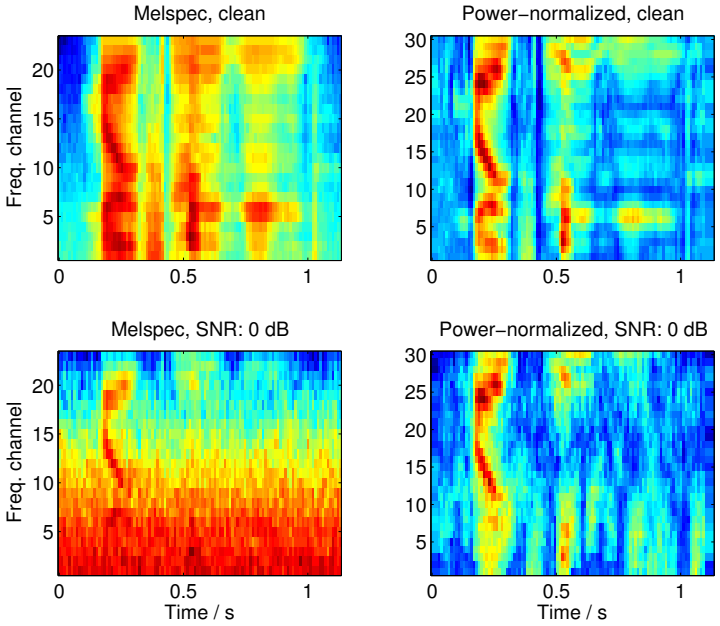

Figure 3.3 (b), Clean and noisy Mel spectrogram (left) and power normalized spectrogram (right).

3.2. Spectral-Temporal Modulation Gabor Filter

After the PN spectum is generated, we further convolve it using two-dimensional spectro-temporal modulation Gabor features. In the remainding section, we discuss the concept of spectro-temporal receptive fields (STRFs), Gabor filter design and sparse selection of informative features from the filter output.

3.2.1. Broader Motivation of Modulation Feature

Over the last decade, a number of physiological experiments on different mammalian species have revealed that the neurons in the primary auditory cortex are sensitive to particular spectro-temporal patterns referred to as spectro-temporal receptive fields [45],

which is a functional descriptor of the linear processing of time-varying acoustic spectra by the auditory system. Based on this evidence, spectro-temporal features, which serve as a model for STRFs, have been applied to ASR. Several studies have successfully incorporated Gabor function approximations into ASR [35, 39, 64]. In general, these approaches define a series of spectral, temporal, and spectral-temporal modulation filters which can be seen as roughly modeling neuron firing patterns for particular spectro-temporal signal components. Purely temporal features such as TRAPS [27] and HATS [11] can be regarded as special cases of spectro-temporal features. Gabor filters have also been used for speech and nonspeech discrimination [46, 72].

3.2.2. Computational Gabor Model for Auditory Receptive Fields

There are two ways to characterize the spectral-temporal response field. From the studies of the ferret primary auditory cortex, simpler spectra consisting of single moving ripples, which are basically sinusoidally modulated spectral profiles with a constant frequency along time and the "logarithmic" frequency axis, could be used effectively to characterize the response fields and transfer functions of auditory cortex cells. The response, thus, is formulated as

$$O(n,k) = \Delta A \sin(2\pi\omega_n n + 2\pi\omega_k k + \Phi) \tag{3.1}$$

where spikes were measured with different temporal (denoted as ω_n) and spectral modulation (denoted as ω_k) at different amplitude levels (denoted as ΔA).

Alternatively, STRF processing can be formulated as filtering:

$$O(n,k) = STRF(n,k) * x(n,k) \tag{3.2}$$

At any particular time *n* and frequency *k*, a neuron's response $O(n,k)$ is given by reverse correlation/convolution of the STRF and the dynamic spectrum of the stimulus around that instant and frequency $x(n,k)$. Thus, the STRF acts as a filter, firing for the strongest responses to spectro-temporal features that most resemble its own structure. Many studies provide different ways to estimate STRFs.

For speech processing, a commonly used approximation of STRFs is a 2D Gabor function. 2D Gabor filters closely resemble the spectro-temporal response fields of neurons in the primary auditory cortex, and in particular are used to extract features that simultaneously capture spectral and temporal modulation frequencies for automatic speech applications, as they are used to extract spatial-temporal modulation frequencies for image processing applications [41]. The overall sensitivity pattern for human hearing has also been observed via perceptual experiments, e.g., Chi et al [13]. It was observed that humans are most sensitive to temporal modulation frequencies up to 16 Hz and spectral modulation frequencies up to 2 cycles per octave.

To generate Gabor filters serving as model for STRFs, we multiply a complex sinusoid with a Hanning envelope.

$$STRF(n,k) \cong Gb(n,k) = s(n,k) \cdot h(n,k) \tag{3.3}$$

The complex sinusoid (with time modulation frequency ω_n and spectral modulation frequency ω_k) is represented as:

$$s(n,k) = \exp\left[i\omega_n n + i\omega_k n\right] \tag{3.4}$$

while Hanning envelope is given:

$$h(n,k) = [\frac{1}{2}(1-\cos(\frac{2\pi n}{W_n+1})][\frac{1}{2}(1-\cos(\frac{2\pi k}{W_k+1}))] \tag{3.5}$$

where the time support W_n and frequency support W_k is defined as 1.75 cycles of the corresponding modulation frequency:

$$W_n = 1.75 \cdot \frac{2\pi}{|\omega_n|}, W_k = 1.75 \cdot \frac{2\pi}{|\omega_k|} \tag{3.6}$$

For purely temporal or spectral filters, this definition results in an infinite support function; in these cases, the support is limited to 40 frequency channels or 99 time frames, which corresponds to the maximum size of the other filters in the respective dimension.

By tuning parameters of spectral and temporal modulation frequency, Gabor functions have different extent and orientation for a given number of oscillations under the envelope. The Gabor filter bank used here has been adapted from [48]. The 59 Gabor filters focus on different temporal and spectral modulation frequencies as shown in Table 3.1 and Fig. 3.4.

Temporal modulation [Hz]	0, 1.9, 3.9, 6.2, 9.9, 15.7, 25
Spectral modulation frequency [cycle/oct]	-0.25, -0.1224, -0.06, -0.0293 0, 0.0293, 0.06, 0.1224, 0.25

Table 3.1, Temporal and spectral modulation frequencies for Gabor function

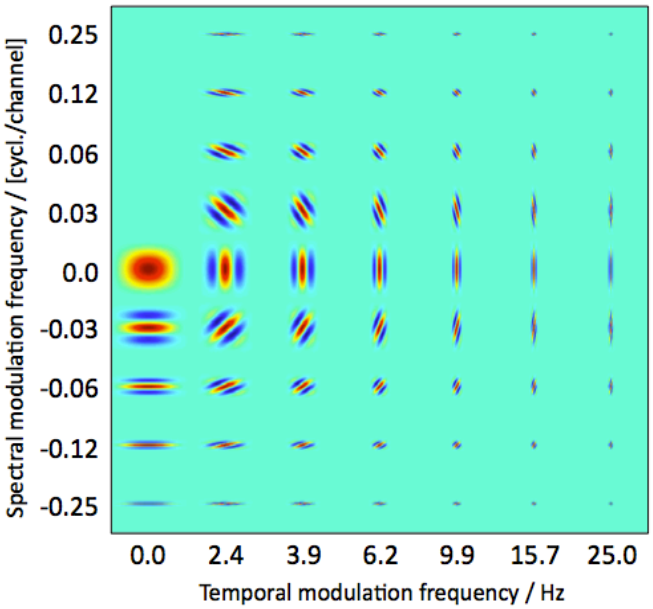

Figure 3.4, Frequency response of Gabor filters

The 59 Gabor filters are then applied to filter the PN spectrum. For the high modulation filters, the narrow filters capture the fast time-varying part of the spectrum, as shown in Fig. 3.5. For the low modulation filters, the wide filters capture the coarse representation of speech dynamics. A variety of tall and short filters corresponding to different spectral modulation frequencies generate features capturing different representations of spectral dynamics. In this regard, the standard cepstral coefficients (MFCC or PNCC) can be considered a special case of spectro-temporal features where the 2D filters measure spectral modulation frequencies across the entire spectrum within a single time frame i.e. cepstral coefficients represent the 2D filters which are very tall and narrow.

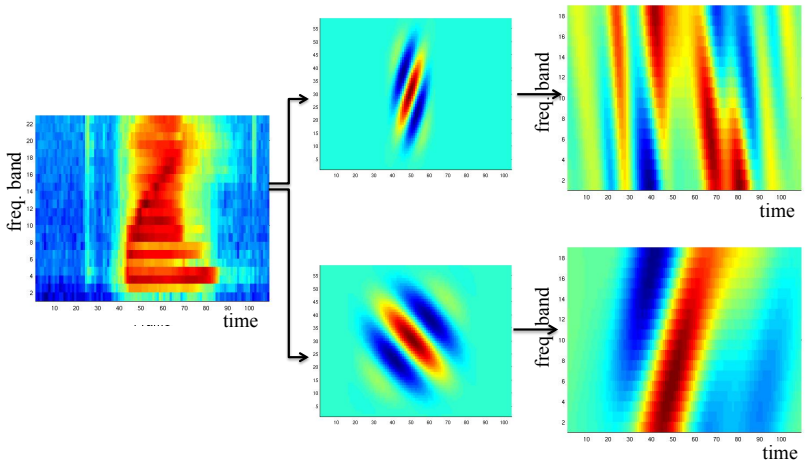

Figure 3.5, Above: a high temporal modulation filter (temporal modulation frequency: 6.2 Hz, spectral modulation frequency: -0.03 cycle/channel) and corresponding filter output; below: a low temporal modulation filter (temporal modulation frequency: 2.4 Hz, spectral modulation frequency: 0.03 cycle/channel) and corresponding filter output

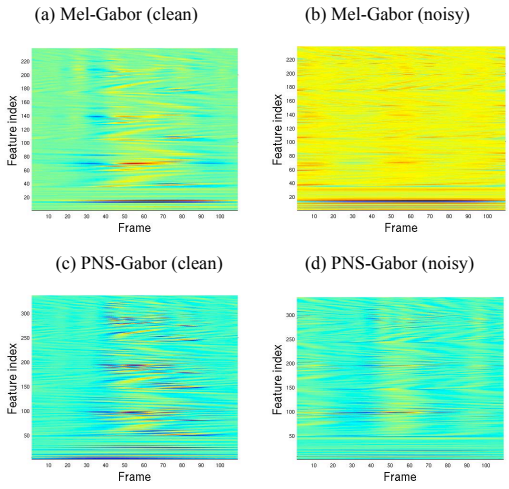

Figure 3.6, Clean and noisy Mel spectrum based (top) and power normalized spectrogram based Gabor features(bottom).

3.3. Sparse Feature Selection

While our experiments have shown the utility of using spectro-temporal Gabor filters for ASR, some features generated by Gabor filters may not be informative. Here we further propose an approach to select informative Gabor features using sparse principle component analysis. Based on the leading sparse principal vectors, we discard Gabor features that might prove to be less useful.

Classical PCA is a widely used tool for dimensionality reduction, providing a linear combination of "all" features that maximizes data variance. However, in general, the principal vectors are dense (i.e., the entries are non-zero), which makes the results

difficult to interpret. Sparse PCA on the other hand, solves the same problem using "sparse" principal vectors. The objective function of sparse PCA is modified from classical PCA with an ℓ_1-norm penalty, so that sparse principal vectors are favored. The objective function is represented as:

$$\max v^T \Sigma_x v - \rho(\|v\|_1^2) \text{ subject to } \|v\|_2 = 1 \tag{3.7}$$

where ρ is a non-negative parameter controlling the sparsity of principal vector, Σx is the empirical covariance matrix calculated from observations $\{x_i\}$ and v is the leading sparse principal vector. The first term in (3.7) is the objective function of classical PCA.
Formula (3.7) can be rewritten as:

$$\max Tr(\Sigma_x v v^T) - \rho(1^T |v v^T| 1) \text{ subject to } \|v\|_2 = 1 \tag{3.8}$$

To get a robust interpretation, we reformulate the ℓ_1-norm penalty:

$$-\rho(1^T |v v^T| 1) = \min_U Tr(U v v^T), -\rho \le U_{ij} \le \rho \tag{3.9}$$

We obtain formula (3.10) by applying (3.9) to (3.8):

$$\max\min Tr((\Sigma_x + U) v v^T) : \|v\|_2 = 1, -\rho \le U_{ij} \le \rho \tag{3.10}$$

By switching the trace of the product in (3.10), it is equivalent to:

$$\max\min v^T (\Sigma_x + U) v : \|v\|_2 = 1, -\rho \le U_{ij} \le \rho \tag{3.11}$$

The dual problem then can be shown as:

$$\max\min v^T (\Sigma_x + U) v : \|v\|_2 = 1, -\rho \le U_{ij} \le \rho \tag{3.12}$$

$$\text{or} \quad \min_U \lambda_{\max}(\Sigma_x + U) \ : -\rho \le U_{ij} \le \rho \tag{3.13}$$

where $\lambda_{max}(\Sigma_x + U)$ means maximum eigenvalue of $(\Sigma_x + U)$

Based on (3.13), we can solve the original problem by searching for the minimum of the largest eigenvalue of the covariance matrix with a noise matrix. Sparsity is accomplished by eliminating small values imposed on the empirical covariance matrix by component-wise noise bounded by ρ. To speed the search for U, we apply the Augmented Lagrangian Method (ALM) algorithm as derived in [56].

We compute only the leading sparse eigenvector, which is sufficient for selecting informative features in our experiment. The feature variables corresponding to zero

entries in the sparse principal vector are less useful to account for data variance. Hence, only informative features corresponding to non-zero entries are selected. The dimensionality of the selected features is much smaller than the original.

The sparse Gabor features are then used as input into a deep neural network. In typical tandem processing, a deep neural network is employed to learn discriminative transformations from front-end features; in this case Gabor filtered PN spectra, to probabilistic features, which are referred as sparse PNS-Gabor DNN features. We used a deep neural network with bottleneck structure where a bottleneck feature vector is generated by a narrow-dimensional layer in the middle of the network as shown in Fig. 3.7.

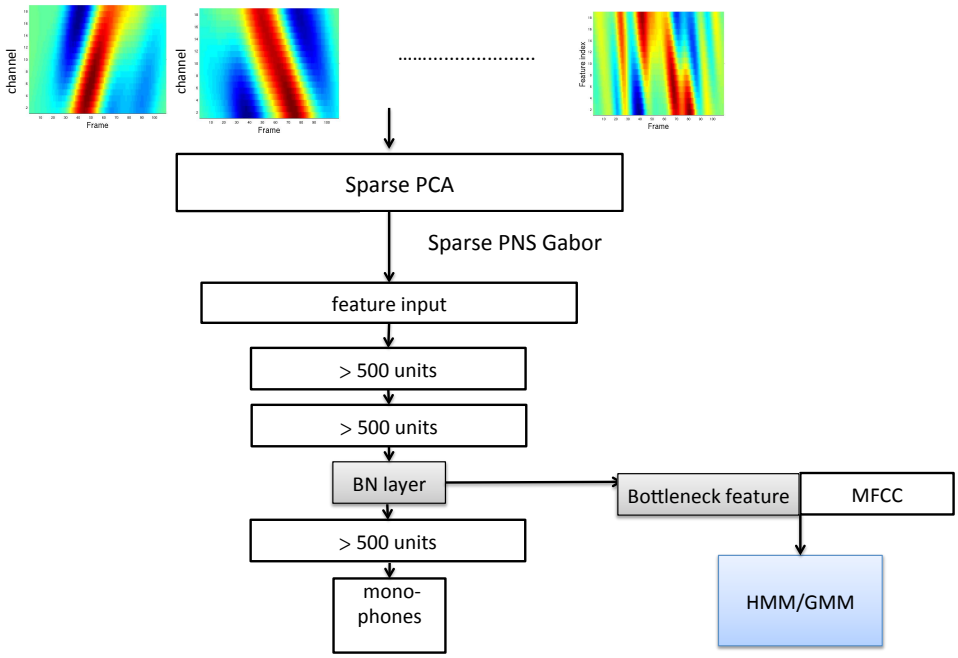

Figure 3.7, Bottleneck deep neural network tandem system using sparse Gabor features

3.4. Gabor Convolutional Neural Network

In additional to incorporating Gabor filters into the input layer of fully connected DNNs, here we propose another way to integrate pre-defined Gabor filters into convolutional neural networks. The proposed neural network architecture, called the Gabor Convolutional Neural Network (GCNN), incorporates Gabor functions into convolutional filter kernels. As described in Chapter 2, typical CNN architectures use shared weights to filter a receptive field, modeling the local characteristics of a spectrum. This filtering process permits us to integrate 2D Gabor filters into the CNN topology. We modified the receptive fields of the CNN, with several time and frequency supports conforming to Gabor filter characteristics. The modified CNN includes Gabor filter coefficients as the

initial filters at the lowest layer, and performs fine-tuning to optimize the coefficients by back propagation training. The GCNN feature is better than both Gabor-DNN and CNN features, where the former kept Gabor coefficients untrained while the latter used trained filters without Gabor modeling. Also, pooling reduced word error rate effectively for recognition of noisy speech.

The typical convolutional neural network using PN-spectrum is shown in Fig. 3.8, which consists of a convolutional layer, a subsampling layer, and fully connected layers as described in section 2.3.

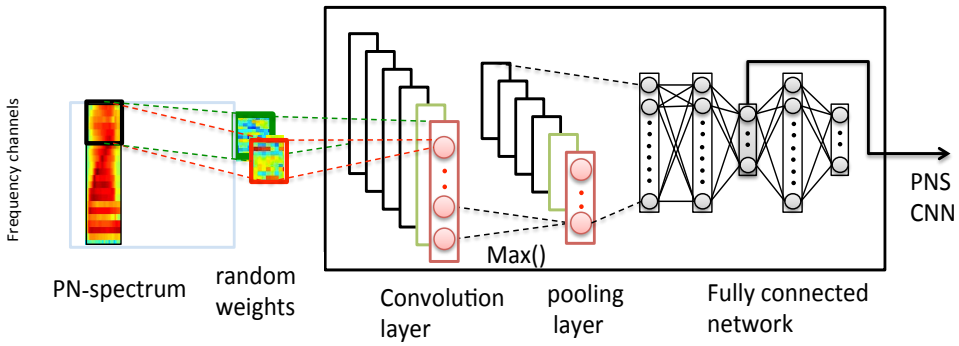

Figure 3.8: Basic convolutional neural network topology with PN-spectrum

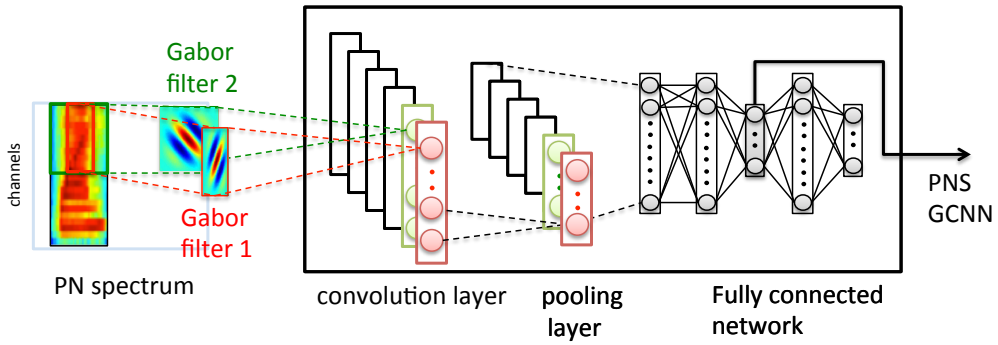

Figure 3.9: Gabor convolutional neural network topology

To model Gabor filtering in CNNs, we made two modifications in the basic processing for the convolution layer. First of all, we used linear instead of sigmoid activations. The bias term was enforced to be zero. By these modifications, the neurons of convolutional layer are just the filter outputs of the receptive field. Second, Gabor features consist of filters with multiple time and frequency band supports. To meet the filter design, we modified the receptive field size to give the same supports as the Gabor filters, instead of using a fixed receptive field size i.e. $2K_m+1$ (bands) by $2N_m+1$ (frames) matrix depending on filter m. The Gabor filter coefficients are then flipped and incorporated into the initial weights W_m. The convolutional activations can be formulated as Eq. 3.14.

$$\begin{aligned} h_m(n,k) &= lin(\sum_{i=-N_m}^{N_m}\sum_{j=-K_m}^{K_m} Gb_m(-i,-j)\cdot x(n+i,k+j)) \\ &= Gb_m(n,k) * x(n,k) \end{aligned} \tag{3.14}$$

where $h_m(n,k)$ represents the neuron of m^{th} feature vector based on a particular Gabor filter, whose receptive field is $2K_m+1$ (bands) by $2N_m+1$ (frames) matrix centered at the current band of the frame, $x(n,k)$. The connection weights W_m perform filtering on the receptive field where the indices of the filter coefficients are flipped from the weight indices, both vertically and horizontally as shown in Eq. 3.14.

As shown in Eq. (3.14), the initial feature vectors were Gabor feature vectors. The modified topology is depicted in Fig. 3.9. These filter coefficients are then trained with typical error back-propagation training. Thus, unlike Gabor-DNN features, filter coefficients were no longer untrained.

Compared to a typical CNN, GCNN used Gabor filter characteristics to define linear activated feature vectors with multiple time and frequency resolutions to initialize back-propagation training, and potentially avoid overfitting to training data. A max-pooling layer follows the convolutional layer to down-sample and smooth the Gabor features. A comparison between Gabor-DNN, CNN and GCNN is summarized in Table 3.2.

Topology	Filter coefficient	Filter support	Pooling
Gabor-DNN	Gabor coefficient	Gabor support	No
CNN	Random initial and trained	Fixed time-frequency support	Yes
GCNN	Gabor initial and trained	Gabor support	Yes

Table 3.2: Comparison between Gabor-DNN, CNN and GCNN

3.5. Experimental Results

3.5.1. Noisy Speech Corpus

The approach proposed here is evaluated with the Aurora 2 [30] testing environment covering the recognition of noisy digits and with two noisy versions of WSJ: (1) Aurora 4 [58] and (2) RATS "re-noised" Wall Street Journal (WSJ) speech.

For Aurora 2, we use the clean connected digits for training. Three testing sets (set A, B and C) are used with clean and noisy data. The testing data set A covers four different noise types (subway, babble, exhibition and car), while the testing data B covers four different noise types (restaurant, street, airport and train station). The testing set C covers two noise types respectively from set A and set B (subway and street), plus additional convolution noise. Different SNR values ranging from 0 dB to 20 dB were tested in each case. The average word error rates (WERs) of this task are obtained by averaging over WERs of the test sets.

The Aurora 4 dataset provides both a clean training set and a multi-condition training set. The clean training set is taken from 7138 utterances of the WSJ0 SI-84 dataset (83 speakers) where the data was recorded using a Sennheiser microphone. The multi-condition training set contains the same number of utterances as the clean training set, while half of the utterances were recorded by a secondary microphone. Six noise types (car, babble, restaurant, street, airport and train) at SNRs between 10dB and 20 dB were randomly added to three-fourths of utterances from both microphone types. The evaluation set is based on 166 utterances of the Nov'92 5k evaluation set (8 speakers), and is composed of 14 subsets: clean and 6 noise corrupted sets for data recorded by both microphone types. The noise types are the same as those used for the multi-condition training set, but were chosen with an SNR between 5 and 15 dB. The 14 subsets are grouped into 4 sets: clean, noisy, clean with microphone distortion and noisy with microphone distortion, which are referred as A, B, C and D respectively.

For RATS re-noised WSJ, we started out with data taken from the WSJ1 dataset (284 speakers) for training and the WSJ-eval94 dataset (20 speakers) for testing. Estimated additive and channel noise from degraded recordings was applied to both training and testing datasets using the "renoiser" tool [21]. Designed for use in the DARPA RATS project, the system analyzes data from RATS rebroadcast example signals (in this case, LDC2011E20) to estimate the noise characteristic including SNRs and frequency-shifts; the original data is described in [2] and consists of a variety of continuous speech sources that have been transmitted and received over 8 different radio channels, resulting in significant signal degradations. The 8 radio channel characteristics are specified in Table 3.3. We applied the same noise characteristics to WSJ data to generate the RATS re-noised WSJ. In this case, the training data was obtained from 51.2 hours of the WSJ1 dataset with the clean channel and channel G (the channel with highest SNR). Testing data was 0.8 hours of WSJ-eval94 for each channel. The results reported here are WERs averaging over clean and 8 noisy channels.

	Microphone	SNR	Frequency shift
Channel A	Motorola HT1250	15.6	0
Channel B	Midland GXT1050	6.2	0
Channel C	Midland GXT1050	6.0	0
Channel D	Galaxy DX2547	3.5	180.9 Hz
Channel E	Icom IC-F70D	0.9	0
Channel F	Trisquare TSX300	3.0	0
Channel G	Vostek LX-3000	18.7	0
Channel H	Magnum 1012 HT	3.0	120.7 Hz

Table 3.3, 8 noisy channels added to WSJ

3.5.2. PN Spectrum and Gabor Processing

First, we perform an analysis of the importance of individual signal processing steps differentiating Mel spectra and power normalized spectra that result in the increased robustness when incorporating Gabor filters. These features are evaluated from Aurora 2. For the small vocabulary experiments in Aurora 2, the HMMs were configured as: whole-word HMMs with 16 states and with 3-Gaussian mixtures with diagonal covariance per state. Baseline results are obtained with the standard Aurora 2 MFCC frontend, which converted each signal frame into 13 cepstral coefficients, with subsequent addition of first and second derivative and utterance-wise mean and variance normalization. The average word error rates (WERs) of this task are obtained by averaging over WERs of the test sets.

In Aurora 2 experiments, the neural networks in the tandem processing are single hidden layer MLPs. The MLPs were trained with a temporal context window of 9 successive frames. We used 160 hidden nodes for Aurora 2 while the output layer consisted of 56 context-independent phonetic targets. MFCCs were then concatenated with Gabor features. The dimension of Gabor features is then reduced via PCA to 32, resulting in a 71-dimension feature vector.

In Table 3.4, we compare several different configurations of PNS-Gabor features and Mel-Gabor features after concatenating MFCC in Aurora 2. The result for Mel-Gabor features plus MFCC is presented in row (2), which is 15% relative better than the MFCC baseline. From row (3) to row (6), the results were obtained from deconstructing PNS into four different configurations. In row (7), instead of being filtered by Gabor filters, PNCC was used as input for MLP, from which we could investigate the benefit of combining MFCC and PNCC without Gabor filtering. As shown in row (3), we only switched from Mel filter banks to gammatone filter banks. We referred to it as GT(l)-Gabor. In row (4), gammatone filter banks were further processed by power nonlinearity compression p instead of logarithm compression l, which was referred as GT(p)-Gabor. PNS(l)-Gabor features were obtained by performing a power bias subtraction followed by logarithmic compression. The result of regular PNS-Gabor features is presented in row

(6). GT(*l*)-Gabor didn't perform as well as conventional Mel-Gabor features, while GT(*p*)-Gabor gave a slight improvement. This implies that the power nonlinearity is helpful to inhibit the effects of noise, as expected. However, as shown in Table 3.4, the most effective step is power bias subtraction, from which we got 16.7% relative improvement by comparing GT(*l*)-Gabor and PNS(*l*)-Gabor features. The best result came from the PNS-Gabor feature, which is 20% relatively better than the Mel-Gabor feature. Even after power bias subtraction, power nonlinear compression can help. In row (7), we showed the WER from combination of PNCC and MFCC using MLP, which is significantly worse than the proposed Gabor-filtered PNCC augmentation of MFCCs; the latter is 15.7% better.

To conclude, we investigated the key parts of the PNCC algorithm, augmented by Gabor filtering. It appears that power bias subtraction and Gabor filtering are the key steps for decreasing the WER (from 18.14% to 14.06%).

	Filter bank	Pow Sub	Com.	Gb.	WER
(1) MFCC	-	-	-	-	18.14
(2) Mel-Gb + MFCC	Mel	no	log	yes	15.41
(3) GT(*l*)-Gb + MFCC	GT	no	log	yes	15.75
(4) GT(*p*)-Gb + MFCC	GT	no	pow	yes	14.86
(5) PNS(*l*)-Gb + MFCC	GT	yes	log	yes	13.12
(6) PNS-Gb + MFCC	GT	yes	pow	yes	**12.30**
(7) PNCC-MLP + MFCC	GT	yes	pow	no	14.06

Table 3.4, Aurora 2 WER of Gabor features with different spectro-temporal representations. The baseline is a 39-dimensional MFCC plus the first 2 derivatives with mean and variance normalization.

3.5.3. Sparse Gabor Features

Next, we investigated the improvement by introducing sparse feature selection to Gabor features. For the experiments, we evaluate these approaches in large vocabulary continuous speech recognition tasks from RATS re-noised WSJ and Aurora 4.

For both Aurora 4 and RATS re-noised WSJ, the acoustic models used cross-word triphones estimated with maximum likelihood. The resulting triphone states were clustered to 2500 tied states, each of which was modeled by 16 components of a Gaussian mixture model. We used version 0.6 of the CMU pronunciation dictionary and the standard 5k bigram language model created at Lincoln Labs for the 1992 evaluation. Unless otherwise specified, mean normalization was performed for the features, while vocal tract length normalization (VTLN) and adaption techniques such as maximum likelihood linear regression (MLLR) were not employed for these tests.

The fully connected deep neural networks were trained with a 4 hidden layer bottleneck structure with a bottleneck (25 units) in the third hidden layer. The output layer consisted of 41 context-independent phonetic targets. The features with 9

successive frames were used as input for a fully connected deep neural network. For fair comparison, the number of free parameters of the neural networks were constrained to roughly 3.5M by controlling the hidden layer size.

Restricted Boltzman machine (RBM) pre-training was employed to initialize the parameters of the neural network. For back propagation following the pre-training, we began with a learning rate of .008 and reduced the learning rate by factors of two once cross-validation indicated limited progresses with each learning rate, and continued until cross-validation showed essentially no further progress.

After DNN features were computed, the MFCCs were concatenated with the deep neural network trained features, resulting in a 64-dimensional feature vector. Also, means and variances were normalized per utterance before HMM training and testing for all the features described here.

We first present a series of baseline WER results for RATS re-noised WSJ and Aurora 4 results using the clean training set. In Table 3.5, PNCC was better than other feature baselines for both cases (on average). As a result we used PNCC or PN spectrum based features for the experiments that followed.

Feature	RATS	Aurora 4				
	WSJ	A	B	C	D	Avg
MFCC	62.4	8.2	36.7	25.5	52.1	40.5
MFCC(CMVN)	61.8	7.9	30.3	22.8	48.2	35.8
ETSI-AFE	59.4	8.8	24.5	24.7	38.9	29.5
PNCC	58.8	9.3	22.1	23.3	37.5	27.9

Table 3.5: MFCC, AFE and PNCC baseline

In Table 3.6, we compare the DNN features based on PNCC, PNS-Gabor and the dimensionally reduced PNS-Gabor feature selected via sparse PCA and classical PCA. In this experiment, the sparse PCA parameter, ρ was set to .008, resulting in 271 selected features from 819 features. The size of hidden layers was increased so that total number of parameters was comparable. As shown in Table 3.6, the sparse-PCA selection was better than using the other features. Aside from this improvement, the sparse PCA also made feature generation more efficient. The sparse PCA approach was furthered compared with classical PCA. Results also suggest that using the reduced "informative" features via sparse PCA is better than using the linear combination of all features obtained by classical PCA.

Feature	RATS WSJ	Aurora 4				
		A	B	C	D	Avg
MFCC-DNN	59.2	6.2	25.2	24.8	45.1	32.3
PNCC-DNN	54.8	6.63	20.4	18.7	36.6	26.2
Gabor-DNN	53.9	6.85	18.9	17.8	35.4	25
Sparse Gabor-DNN	51.7	6.72	18.2	16.9	34.1	24.1
Gabor-DNN (classical PCA)	52.9	6.8	18.5	17.4	34.8	24.6

Table 3.6, WERs for DNN features based on PNCC, PNS-Gabor and dimensional reduced PNS-Gabor using sparse PCA or classical PCA

Using the sparse principal vector, we observed the suggested importance of Gabor filters in Fig. 3.10. The darker areas represent the degree of importance (at least in terms of variance) based on how many channels generated by the filter were selected. As shown in Fig 3.10, filters with low temporal modulation frequency appear to be useful for maximizing data variance. The features focus on temporal modulation frequency of 0, 2,4 and 3.9 Hz account for 94.1% of the weights of the leading sparse principal vector. The corresponding filter lengths are roughly 1, 0.7 and 0.5 sec. It suggests that Gabor filters with longer time scales are particularly informative.

We also note that in all cases except for A (high SNR, no frequency shift) the sparse Gabor outperforms all the other approaches.

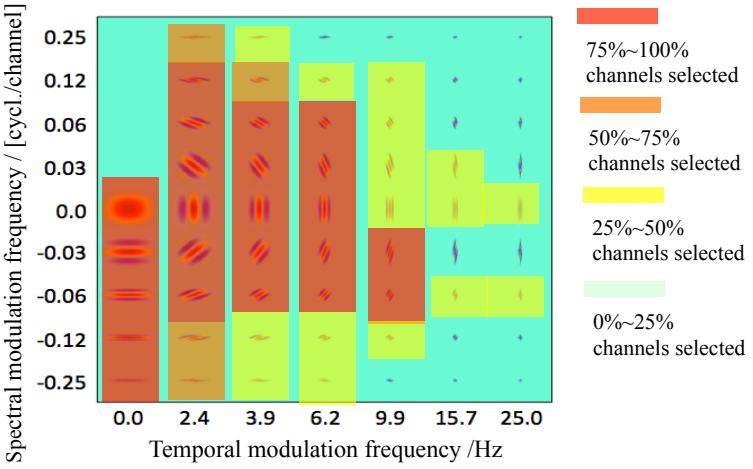

Figure3.10: Importance of Gabor filters based on sparse PCA

3.5.4. Fully Connected DNN, CNN and GCNN

Here, we compare the WERs of fully connected DNN, convolutional neural network and Gabor convolutional neural network. For the CNN topology, we used 120 filters for the convolutional layer. The filter size was 9 frequency bands with 15 successive frames. We used a pooling size of 6 convolutional bands with stride 2 (overlap by 4), which reduced dimensionality by a factor of 2. This layer was fed to a 5-layer fully connected bottleneck structure. In the GCNN architecture, the time support for each filter kernel ranges from 7 to 99 frames, and frequency support ranges from 7 to 40 bands. 59 of the filters were initialized as Gabor filter coefficients, and the other 61 filters were randomly initialized. The rest of network set up is the same as for the CNN. For both CNN and GCNN architecture, 40-d power normalized spectrum was used as input. We didn't use delta and acceleration coefficients to be consistent with Gabor filter input. Back propagation strategy was the same as used for the DNN, while no pre-training was performed. Again, MFCCs were concatenated with the convolutional neural network trained features, resulting in a 64-dimensional feature vector.

In Table 3.7, we compare a series of trained features using a fully connected neural network, a convolutional neural network and a Gabor convolutional neural network. First, the trained features of Table 3.7 are better than untrained features of Table 3.5. In Table 3.7, Gabor-DNN was better than PNCC-DNN except for the clean set (A). Next, we compare Gabor-DNN with PNS-CNN and PNS-GCNN without a pooling layer. Without pooling, the inputs of the fully connected network are feature maps of the convolutional layer. Therefore, these rows function as a comparison between different sets of spectro-temporal filters. Gabor-DNN used filters with variable size, but totally handcrafted. PNS-CNN, on the other hand, learned filters with fixed size and trained on limited data. PNS-GCNN has filters with variable size. Also, the trained filters were initialized with handcrafted filters. In Table 3.7, PNS-GCNN was better than the other two features, although the differences are small. The larger effects visible in the table show the effects of pooling, and the cumulative effects of pooling and using GCNN instead of CNN. In particular, max pooling provides a significant improvement for both RATS WSJ and Aurora 4 (especially for noisy set (B) and noisy set with channel distortion (D)), and particularly with pooling, using Gabor filters to help design the CNN has a good effect.

Feature	RATS WSJ	Aurora 4				
		A	B	C	D	Avg
MFCC-DNN	59.2	6.2	25.2	24.8	45.1	32.3
PNCC-DNN	54.8	6.63	20.4	18.7	36.6	26.2
Gabor-DNN	53.9	6.85	18.9	17.8	35.4	25
PNS-CNN no pooling	54.1	6.34	19.8	17.9	35.8	25.5
PNS-GCNN no pooling	51.6	6.52	18.3	18.5	34.6	24.4
PNS-CNN with pooling	51.5	6.2	18.8	15.6	33.8	24.1
PNS-GCNN with pooling	49.8	6.3	17.8	15.7	32.1	22.9

Table 3.7: WER for neural network features, clean training, noisy test.

In addition to the experiments with mismatched training and testing, we also used the multi-condition training set for Aurora 4. We chose the distinguished features, Gabor-DNN, Sparse Gabor-DNN, PNS-CNN and PNS-GCNN comparing them with two baselines: ETSI-AFE and PNCC. The results are shown in Table 3.8, where the proposed PNS-GCNN could achieve 16.6% WER. This was achieved without VTLN, MLLR, or other modeling enhancements. Table 3.7 and 3.8 concludes that GCNN based features achieve the best performance over all the other features for both Aurora 4 and RATS re-noised WSJ.

Feature	Aurora 4				
	A	B	C	D	Avg
ETSI-AFE	10.6	18.6	19.7	30.9	23.4
PNCC	10.5	17.4	19.1	30	22.5
Gabor-DNN	8.4	14.2	14.3	25.8	18.8
Sparse Gabor-DNN	7.8	13.8	12.5	24.1	17.7
PNS-CNN with pooling	7.4	13.4	12.8	24.7	17.8
PNS-GCNN with pooling	7.3	12.8	12.1	22.7	16.6

Table 3.8: WER for multi-condition training set.

3.5.5. CNN Trained Filter versus Gabor Filter

In the previous experiments, we reported ASR results of GCNN that integrated pre-defined Gabor filters and trained convolutional neural networks to generate a more robust feature. From the experiments, the GCNN features performed better than both Gabor-DNN and CNN features. We further investigated the filters trained from CNN and GCNN. The trained CNN filters are composed of several vertical (spectral) and horizontal (temporal) filters, as the examples in Fig. 3.11 show. However, the filters have very low correlation to the diagonal Gabor filter, such as the diagonal filter in Fig. 3.12 (left) while the diagonal filter would be kept and tuned in GCNN topology and the final filter is shown in Fig. 3.12 (right). Thus, diagonal filtering is another factor distinguishing trained filters with Gabor initialization and those with random initialization.

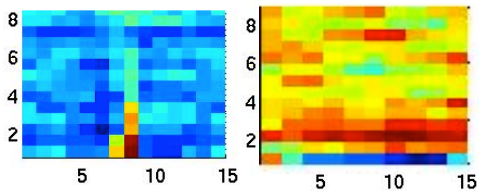

Figure 3.11, a vertical and a horizontal filter example

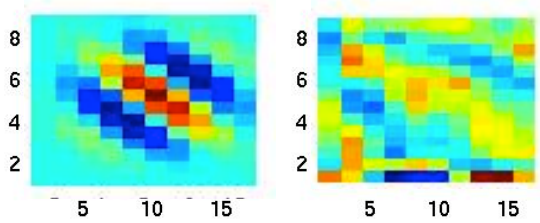

Figure 3.12, (left): diagonal Gabor filter, (right): GCNN tuned diagonal Gabor filter

3.6. Summary

Here we incorporated Gabor features into a deep neural network and a convolutional neural network. First of all, we employed a more robust spectro-temporal representation incorporating key parts of the PNCC algorithm, augmented by Gabor filtering. From the analysis experiments, the power bias subtraction and Gabor filtering are the key steps for decreasing the WER. Second, we reported our use of sparse PCA on PNS-Gabor, which is used as input for bottleneck deep neural networks using two large hidden layers following the input layer. The key factor in this step is discarding uninformative features. Third, we proposed a robust CNN architecture integrating Gabor filter design. The proposed GCNN architecture learned local features with multiple temporal and spectral resolutions with Gabor filter initialization, both for structure and initial weights. The filter coefficients were further optimized by back propagation training. A maximum pooling layer also gave significant improvement in our experiments. Our results indicated that the proposed GCNN feature achieved the best results among other noise robust feature and neural network feature for the two noisy WSJ corpora. It appears that, at least for these tasks, it is useful to design the architecture and the input features for greater robustness rather than just relying on the CNN to learn everything.

Chapter 4

Quantifying Neural Network Feature Errors and Model Errors

In this chapter, we explore the questions surrounding how the application of deep neural networks improves speech recognition accuracy, and why it fails for particular train-test conditions rather than focusing on how to actually improve speech recognition accuracy as in Chapter 3.

The sources of errors in ASR are from two primary factors, model errors and observation mismatch. For the model error, the major factor comes from the incorrect assumptions of the standard acoustic model, the hidden Markov model, which fails to accurately model speech data. Observation mismatch comes from uncompensated training-testing data difference including noise, reverberation, speaking style (accent, speaking rate) and etc. While these factors have long been observed, most research aimed at improving speech recognition accuracy in either acoustic feature or model aspects has largely ignored questions about quantifying the underlying causes of recognition errors. These improvements from models or features without standard statistical data analysis have long been argued as trial and error processes.

In this chapter, we used the statistical tool for analysis experiments introduced by Wegmann, et al. [22, 59], which used simulation and a novel sampling process to quantify the effects that major HMM assumptions have on recognition accuracy. We first discovered the basic mechanisms that neural network-based features use to substantially improve Gaussian mixture based HMM speech recognition systems for matched near-field or far-field experiments. Second, we investigated the failings of standard MFCC based DNN for the mismatched train-test condition. Third, we explored the contribution of robust signal processing techniques prior to neural network training. To accomplish robust processing, we employed the representation of PNS-Gabor as described in Chapter 3 that incorporated Gabor filtering and power normalized spectrum prior to neural network training. The analyses of the improvement from DNN features based on this robust representation allowed us to investigate the contribution of robust feature generation within the DNN framework. We conducted the experiments on the recognition performance of the ICSI meeting corpus where near-field and far-field conditions are recorded simultaneously.

4.1. Model Errors and Observation Mismatches

It has now been decades since hidden Markov models were first applied to the problem of speech recognition ([5, 33]). Moreover, it has been over 20 years since the speech recognition community has used HMM as the dominant paradigm for most acoustic modeling problems.

While other alternatives exist, including conditional random field [25] and more recently long short-term memory (LSTM) with connectionist temporal classification (CTC) implementation [23], HMMs still dominate most speech recognition tasks. When applying HMMs to the problem of ASR, there are two main assumptions that we make. The first assumption is the parametric models that we use for the HMM's output distributions. Often, the emission probability is assumed to be multivariate Gaussian with diagonal covariance (though the hybrid neural network/HMM approach didn't model Gaussian distribution). The second assumption is the statistical independence of frames where we assume that successive frames generated by a certain state are independent, moreover, that frames generated in one state are independent of those generated by a different state. More precisely, given state sequence $s_{1,2...T}$ and the model parameter λ, the likelihood of observations is obtained from the multiplication of likelihoods of each frame:

$$P(o_1, o_2 ...., o_T \mid s_1, s_2 ...., s_T, \lambda) = \prod_{i=1}^{T} P(o_i \mid s_i, \lambda) \tag{4.1}$$

The independence assumption is incorrect partly because of the mechanics of speech production and partly because of the feature extraction process; for example, cepstral coefficients have 15 ms overlapping while computed from 25 ms analytic window and appended with first and second difference computed from adjacent frames; and Gabor features processed time support up to 99 frames. While both of these assumptions are understood to be incorrect for speech data, we can investigate the impact each of these two assumptions has on recognition accuracy and in particular if one assumption dominates recognition errors. This type of diagnostic information is a critical step towards improving or replacing the HMM for speech recognition.

Another source of errors comes from mismatched training/testing data. As described earlier, speech recognition is operated in the environment where noise and reverberation may be present. It is not feasible to collect and transcribe the huge amount of training data that include all noise, reverberation, and other conditions. The testing data with unseen acoustic condition that deviate from training data degrades the performance rapidly leading to another brittleness of ASR. To analyze the problem, we used the paralleled recordings from near-field and far-field microphones in the ICSI meeting corpus [32] to construct three sets of related recognition tasks: (a) matched near-field acoustic model training and recognition test data; (b) matched far-field acoustic model training and recognition test data; (c) mismatched near-field acoustic model training data and far-field recognition test data. The experiments in the matched cases (a) and (b) focused on model errors: (1) long-term statistical dependence that is present in speech

data and violates the HMM's conditional independence assumption and (2) deviation of feature distribution to Gaussian assumptions. For the mismatched case in (c), we quantify the recognition errors from the lacking of robustness to the transformation between the near and far-field acoustics.

4.2. Simulation and Bootstrap Sampling

To quantify these factors, we evaluated the recognition performance on pseudo data that is generated with the controlled statistical property. We used simulation and sampling process to fabricate pseudo test data that deviated from the HMM in different levels. At one extreme, it agrees with all of the model's assumptions, and at the other extreme, deviates from the model in the way real data do. In between, we can precisely control the degree of data/model mismatch. By measuring recognition performance on this pseudo test data, we are able to quantify the effect of this controlled data/model residual on recognition accuracy.

4.2.1. Simulation Pseudo Observations

First, we followed the full generative process assumed by HMMs to simulate the ideal data, which respects all the assumptions of the model. As described above, we used a Gaussian mixture model for state emission distributions. Therefore, the generative process includes sampling steps from two parametric probability distributions where Gaussian distributions determine the feature values while transition probabilities controls the state duration. Thus, the fabricated observations obtained by the simulation process are indeed independent conditioned on the states and follow the Gaussian form of the given hidden states.

To generate the test data by simulation, we started with the ground truth of test utterance, and unpacked the word transcription into phone transcriptions by looking up pronunciation dictionary. Next, we walked through the states and generated the output distribution associated with the states belonging to the triphones to generate the data. Fig. 4.1 shows the full generative process and it is summarized as the following steps:

Simulation process

1. *Convert test transcription from word to phone*
2. *Sample from transition distribution to walk through states*
3. *Sample from Gaussian distribution to generate pseudo data*
4. *Decode the pseudo data*

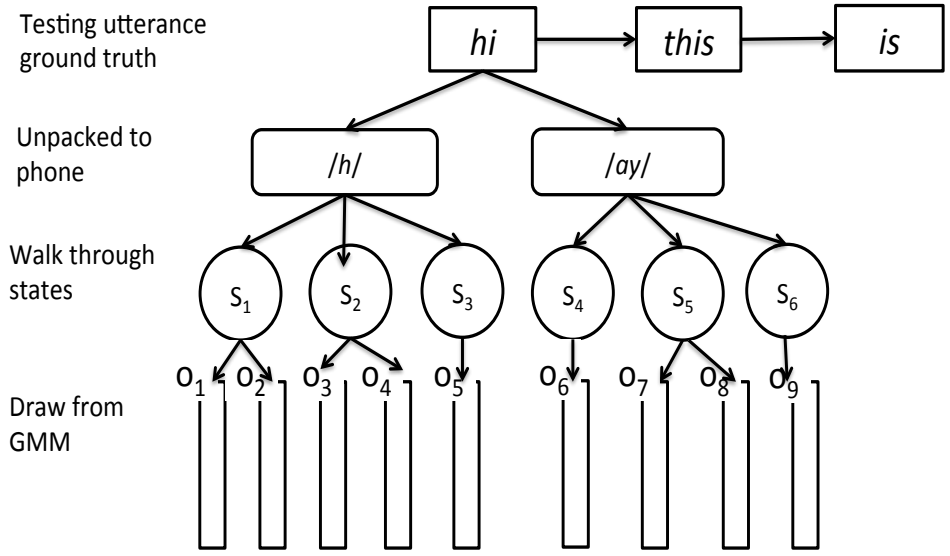

Figure 4.1: Simulation process

In the process mentioned above, although MFCC has delta coefficients and acceleration coefficients computed from adjacent frames appended to the static cepstral features, the GMMs never learn about the temporal consistency between successive feature vectors and their corresponding delta and acceleration features. Therefore, while the pseudo data that we generated still has features corresponding to the static cepstral features, delta and acceleration, they lost the temporal dependency. To simplify the generative process, we used only a single Gaussian distribution for each state instead of typical 32-128 Gaussian mixtures.

4.2.1. Frame Resampling

In this section, we modified the generative process by bootstrap sampling from real data rather than drawing samples from Gaussian distribution. In this fashion, we created data that respects the independence assumptions while follow empirical distribution instead of Gaussian distribution that is controlled by means and variances. To perform bootstrap sampling, we first used the original model to perform forced alignment on the training data, so that each speech frame is aligned with its most likely generating state. Next, we walked through this alignment, filling an urn for each state with its representative frames; at the end of this process, each urn was populated with frames representing its empirical distribution. To generate resampled data, we used the model to create a forced alignment of the test data, and then sampled a frame (at random, with replacement) from the appropriate urn for each frame position; these resampled frames were concatenated as shown in Fig. 4.2. With this resampling, the pseudo test data had exactly the same length as the original, and had the same underlying alignment, but the frames were then conditionally independent (given the state). The resampling process is summarized as:

Frame Resampling

1. *Perform force aligning to obtain frame-level alignment for both training and testing data*
2. *Fill the urns of each state with frames from training data*
3. *Generate test data by sampling (with replacement) from the corresponding urns*
4. *Viterbi decoding the resampled data*

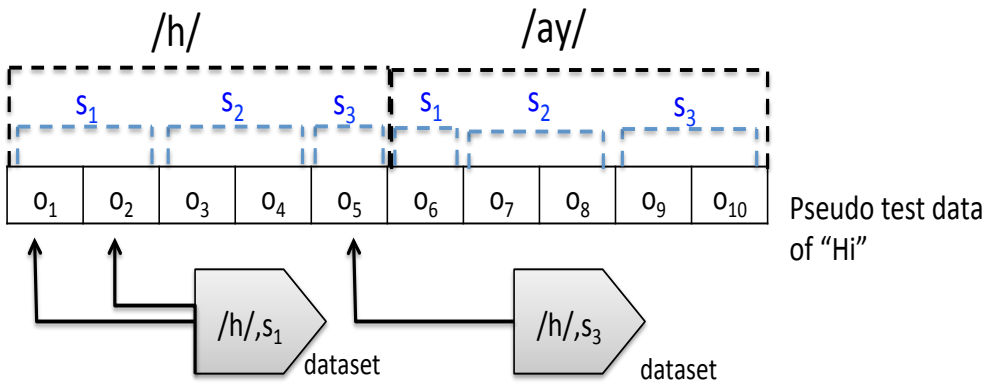

Figure 4.2: Pseudo test data ("Hi") generated using frame resampling

4.2.2. Segment Resampling

From frame resampling, we generated independent observations from empirical distribution. We could further extend the sampling process from frames to segments and relax the independence assumptions to cross-state/phone level. By placing entire state/phone sequences of frames in the urns, and then resampling (again, concatenating samples), we ended up with pseudo test data with dependence among frames within state/phone regions, but independence across state/phone boundaries. As the resampling units were larger than single frames, pseudo test data had different lengths from the original. Fig. 4.3 and 4.4 show the example generating pseudo test data using state/phone resampling.

Phone resampling

1. *Perform force aligning to obtain frame-level alignment for training*
2. *Fill the urns of each "phone" with frames from training data*
3. *Generate test data by sampling (with replacement) from the corresponding urn*
4. *Viterbi decoding the resampled data*

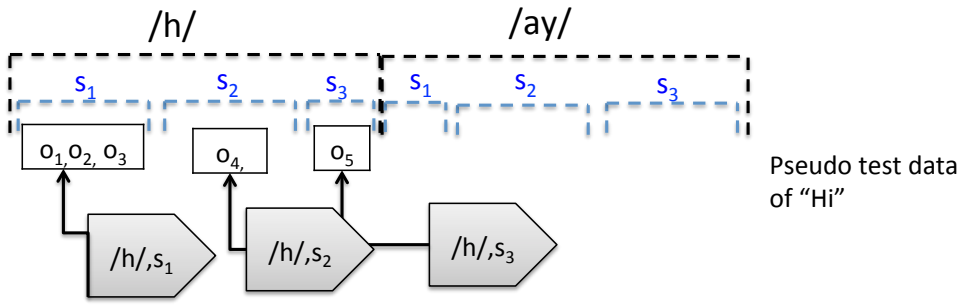

Figure 4.3: Pseudo test data ("Hi") generated using state resampling

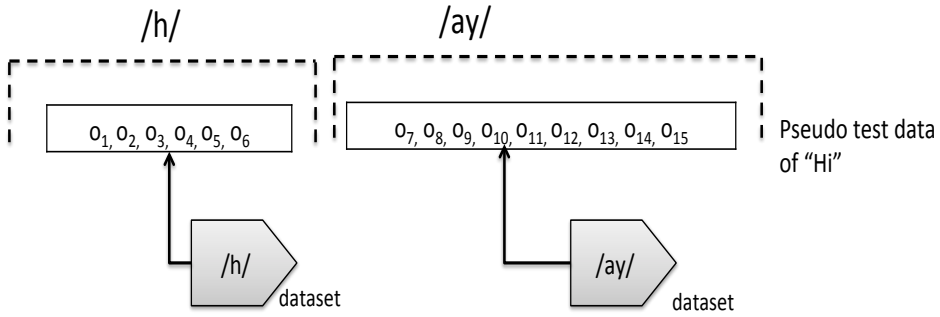

Figure 4.4: Pseudo test data ("Hi") generated using phone resampling

4.3. Data Preparation

For the statistical ASR analysis in this chapter, we used a dataset of spontaneous meeting speech recorded at ICSI [32] where each spoken utterance was captured using near-field and far-field microphones. To avoid the results smeared by a synchronized time skew, we performed time aligning procedures, so that the datasets and the models in the near-field and the far-field cases were completely parallel.

Our training set is based on the meeting data used for adaptation in the SRI-ICSI meeting recognition system [71]. For the test set we used the ICSI meetings drawn from the NIST RT eval sets; this was done to control the variability in the data for the resampling experiments. The remainder of this section discusses the creation of the parallel near-field and far-field corpora. First we describe how we estimated and removed a variable length time delay that existed between the corresponding near-field and far-field utterances, so that each training and test utterance had two parallel versions: near-field and far-field that lined up at the MFCC frame level. Next we discuss how we partitioned these parallel near-field and far-field corpora data into training and test sets.

4.3.1. Time-Aligning the Corpora

In order to synchronize the near-field and far-field recordings, we had to deal with a time delay, or skew, that exists between the two recordings. These time delays arise from two factors: (1) different physical distances between the speakers and the microphones, and (2) systematic delays introduced by the recording software. The latter factor appears to dominate the skew between the near-field and far-field recordings. Fixed delays were introduced when the channels were initialized at the start of a recording. Since this systematic delay dominated the skew, the near-field recordings had a time delay relative to the far-field recordings. Fig. 4.5(a) illustrates an utterance captured by the far-field microphone that is advanced in time in comparison to the same utterance captured by the near-field microphone.

A time delay is more pronounced in the cross-correlation between the near-field and far-field signals, as shown in Fig. 4.5(b). The delay could be estimated by searching for a peak in the cross-correlation sequence. In Fig. 4.5(b) the peak is at a lag of 41.88 ms (670 samples at 16 kHz). However, this detection could be difficult because of the recording quality and noise. To guarantee a more precise detection, we divide each utterance into overlapping windows, where the window size is a third of the utterance length and the step size for successive windows is a tenth of the utterance length. For each step, the cross correlation sequence is calculated and a delay is estimated by taking average of the three segments.

When the variation between the estimated delays in the windows for a given utterance is large, then the estimation is regarded as unreliable and the utterance is discarded. Approximately 30% of the utterances were discarded because of these unreliable delay estimates. The delays between near-field and far-field channels for the reliable data ranged from 12.5 ms to 61.25 ms. Fig. 4.6 shows a discarded utterance. This is a recording of "laugh" followed by “breathing”. However, it appears that the recording from far-field microphone missed the laugh segment. By computing cross correlation, the peak value comes from matching the "laugh" segment of near-field recording and the “breathing” segment of the the far-field recording.

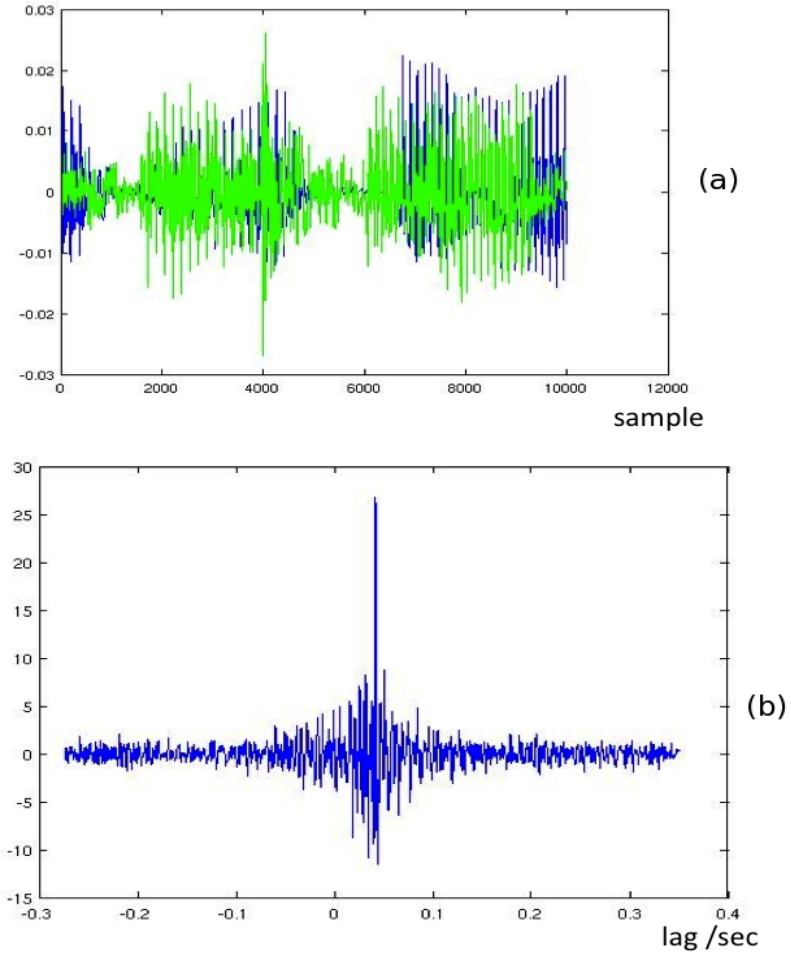

Figure 4.5, Time alignment: (a) near-field (blue) and far-field (green) signals (b) Cross-correlation between the signals.

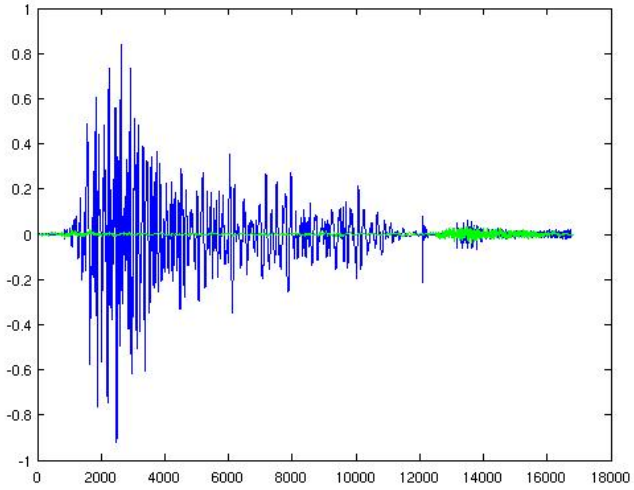

Figure 4.6, Discarded example: near-field (blue) and far-field (green) signals

With time aligning, the datasets and the models are completely parallel in the near- field and the far-field cases so that the errors in the mismatched case can be attributed solely to either the features or the models.

For the alignment of training data, we created alignments using the near-field model on the near- field data, and used this alignment to generate pseudo, far-field test data (for the mismatched case). This avoids the differences in the alignments created by the near-field and the far-field model and lead to the parallel creation of pseudo test sets.

4.3.2. Data Partition

Because of the parallel nature of the near-field and far-field corpora, the data partitions are identical. For simplicity, we describe the near-field partitioning. The training set had a dominant speaker accounting for nearly a quarter; clearly this would skew the data generated by the resampling process. On the other hand, perfect speaker balancing cannot be achieved given that this is a corpus of spontaneous speech. There is, therefore, a trade-off between "the amount of data" and an "egalitarian distribution of speakers". The resulting near-field training and test sets consist of about 20 hours and 1 hour respectively and their statistics are reported in Table 4.1.

ICSI meeting corpus	Training	Test
Speakers	26	18
Utterances	23729	1063
Duration	20.4 (hours)	57.9 (min)

Table 4.1, Training and test statistics for both near-field and far-field set.

4.4. Acoustic Models and Features

The near-field acoustic models use cross-word triphones and are estimated using maximum likelihood. Except for silence, each triphone is modeled using a three-state HMM with a discrete linear transition structure that prevents skipping. The output distribution for each HMM state is a single, multivariate Gaussian with diagonal covariance. While signicantly better performance can be achieved with mixtures of more components, the simplicity of a single component is preferable for our analysis; it also highlights the performance differences between our experiments. Maximum likelihood training roughly follows the HTK tutorial: monophone models are estimated from a "flat start", duplicated to form triphone models, clustered to 2500 states and re-estimated.

Instead of building the far-field acoustic models from a flat start, we exploited the parallel nature of the near-field and far-field training sets to build the far-field models

using *single-pass retraining* from the final near-field models and the far-field data. Single-pass retraining is a form of EM where, in our case, the E-step was performed using the near-field models and data, while the M-step and model updates used the far-field data. We only updated the means and variances of the far-field models, so the result was a parallel set of near-field and far-field acoustic models that share the same state-tying. But the (unknown) transformation between the near-field and far-field means and variances is determined by the frame-level transformation between the parallel near-field and far-field acoustic data.

Since we were using relatively simple acoustic models– single mixture component per state and 2500 tied states–and that the recognition task is much more complex compared to [22], we used a powerful language model (LM) to keep the error rate manageable. In fact, our initial experiments using a weaker LM derived from the training set resulted in WERs as high as 64% in the matched near-field condition.

We used a LM [71] that was trained at SRI by interpolating a number of source LMs; these consisted of webtext and the transcripts of the following corpora: Switchboard, meetings (CMU, ICSI, and NIST), Fisher, Hub4-LM96, and TDT4. We then removed words not in the training dictionary from the trigram LM, and renormalized it. The perplexity of this meeting room LM is around 70 on our test set. To avoid out-of-vocabulary issues, all test utterances containing a word not present in the LM are removed. We used the SRI pronunciation dictionary; it uses two extra phones in comparison with the CMU phone set–"puh" and "pum"– for hesitations.

The features explored here are (1) MFCC (2) DNN features based on MFCC input and (3) DNN features based on robust representation (means were normalized per utterance before HMM training and testing for all the features). We are primarily focusing on the actual improvement of the two DNN features comparing to 39-d MFCC baseline.

For MFCC based DNN features, again we exploited a 4-hidden layer neural network structure with a bottleneck layer in the 3rd hidden layer. The bottleneck size was set to 25 while other hidden layers each consisting of 1600 neurons so that the total number of parameters is about 3M. The network input is 9 successive frames of MFCC. The output layer consisted of 43 context-independent phonetic targets. Restricted Boltzman machine (RBM) pre-training is used to initialize the parameters of the neural network. For back-propagation following the pre-training, we began with a learning rate of .008 and reduced the learning rate by factors of two once cross-validation indicated limited progress with each learning rate, and continued until cross-validation showed essentially no further progress. The final feature was taken from the 25-d bottleneck feature augmented with 39-d MFCCs, which is called MFCC-DNN as used in Chapter 3.

4.5. Results and Discussions

In addition to the original test data, we created near-field and far-field test data by simulation, resampling frames and phones. The corresponding recognition models were used for decoding. All simulation/resampling results reported the average results of 5 repeated experiments. The results of matched near-field and far-field experiments are

discussed in section 4.5.1 and 4.5.2 respectively. The experiments of near-field training and far-field testing experiments are reported in section 4.5.3.

4.5.1. Analysis of Matched Near-Field Results

Previous work on MFCC [22] with matched training/test has shown that recognition errors are dominated by incorrect independent assumptions. The observation still holds for deep neural network features as shown in Table 4.1. In particular, WERs are extremely low for simulated and frame-resampled data where the independent assumption is satisfied by data. By comparing MFCC to DNN features, we observe that deep neural network trained features (both MFCC-DNN and PNS-Gabor DNN) consistently outperform MFCC. For simulated data, transforming MFCC with deep neural network reduces recognition errors by 73%. The improvement decreases as we introduce dependency (at phone-level). Thus, dependency in real data degrades improvement of deep neural network feature in HMM framework.

resampling	(1) MFCC	(2) MFCC-DNN		(3) PNS-Gabor DNN	
	WER	WER	Rel to (1)	WER	Rel to (1)
sim	1.5	0.4	73%	0.5	67%
frame	2.4	0.7	71%	0.8	67%
phone	28.6	12.7	56%	15.3	47%
original	44.7	33.9	24%	36.5	18%

Table 4.2, Results for matched near-field data

4.5.2. Analysis of Matched Far-Field Results

For matched far-field data, deep neural network trained features keep providing significant improvement as shown in Table 4.3. Thus, acoustic feature can be learned using deep neural networks even when training data is noisy. Also, since the difference of PNSGB-DNN and MFCC-DNN is negligible, it suggests that signal processing techniques prior to neural network training didn't contribute to extra improvement for matched far-field data.

resampling	(1) MFCC	(2) MFCC-DNN		(3) PNS-Gabor DNN	
	WER	WER	Rel to (1)	WER	Rel to (1)
sim	1.8	0.5	72%	0.5	72%
frame	3.4	1.2	65%	1	71%
phone	45.5	33.5	26%	33.1	27%
original	71.4	62.8	12%	62.9	12%

Table 4.3, Results for matched far-field data

4.5.3. Analysis of Mismatched Case

For the mismatched case, we observe that MFCC-DNN and MFCC have nearly identical and poor performance. As the diagnostic experiments reported in Table 4.4, recognition errors from observation mismatch of MFCC-DNN is more pronounced for simulated and frame-resampled data where the WERs are 70.9% and 76.9% respectively. For simulated data, applying DNN transformation trained from near-field data to far-field data increase 65% WER relative to MFCC. The result indicates that observation mismatch is a serious issue for MFCC-DNN. While MFCC-DNN is corrupted in the presence of serious mismatch, DNN based on PNS-Gabor performs significantly better than MFCC or MFCC-DNN for all the experiments in Table 4. For simulation, PNS-Gabor DNN reduced 43% WER relative to MFCC and 65% relative to MFCC-DNN. The results suggest that robust signal processing prior to DNN training is the key step for decreasing WER by avoiding specialization and generating more invariant features.

resampling	(1) MFCC	(2) MFCC-DNN		(3) PNS-Gabor DNN	
	WER	WER	Rel to (1)	WER	Rel to (1)
sim	43.0	70.9	-65%	24.5	43%
frame	59.9	76.9	-28%	43.4	28%
phone	80.6	80.8	0%	55.9	31%
original	84.7	83.6	1%	70.1	17%

Table 4.4, Results for mismatched scenario

From our experiments, DNN features can effectively reduce recognition errors when training and test sets are matched whether they are both clean or both noisy; however, the transformation is not generalized enough to apply to realistic data with serious mismatch. Thus, robust signal processing is important for deep neural network feature extraction.

4.5.4. Analysis of Neural Network Weights

Given the robustness of PNS-Gabor DNN, we were curious about whether the weights based on PNS-Gabor feature space are indeed more invariant to different data. To accomplish this, we analyzed the weights between networks trained from near-field and far-field data using both MFCC and PNS-Gabor input. It is infeasible to compare the parameters of two nets, so we trained another net where we initialized it using a net trained from near-field data and then trained the net further using far-field data. Thus, the net starts at near-field trained weights and then eventually adapted to far-field data. The experiments allow us to compare the weights moving from one to another data. We conduct the experiments for both MFCC and PNS-Gabor input. Applying the adapted nets in the mismatched case, WER of MFCC-DNN was significantly decreased from 83.6% to 65.8% and it was decreased from 70.1% to 65.3% for PNS-Gabor DNN. Figure 4.7 shows the weights for the first hidden layer using MFCC (4.7a) and PNS-Gabor (4.7b) input where each data point consists of two variables: initial weight (*X-axis*) and the final adapted weight (*Y-axis*). We observe that weight adaption is more evident for MFCC input and the deviation for PNS-Gabor DNN is relative small. The root mean square deviation between initial weights and final weights for MFCC-DNN is 0.25 and 0.12 for PNS-Gabor DNN. Thus, the net based on PNS-Gabor is indeed more invariant for different data. In particular, Figure 4.8 shows the initial and final weights learned by two hidden nodes of the first hidden layer for 9 frames of a MFCC (C1) and a PNS-Gabor input. The examples illustrate greater insensitivity to different data for the DNN transformation using PNS-Gabor input.

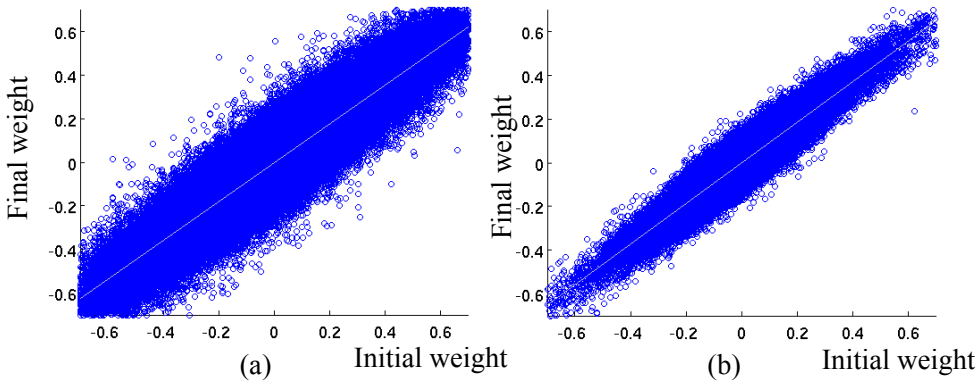

Figure 4.7, Initial weights and final weights of first hidden layer for MFCC input (a) and PNS-Gabor input (b)

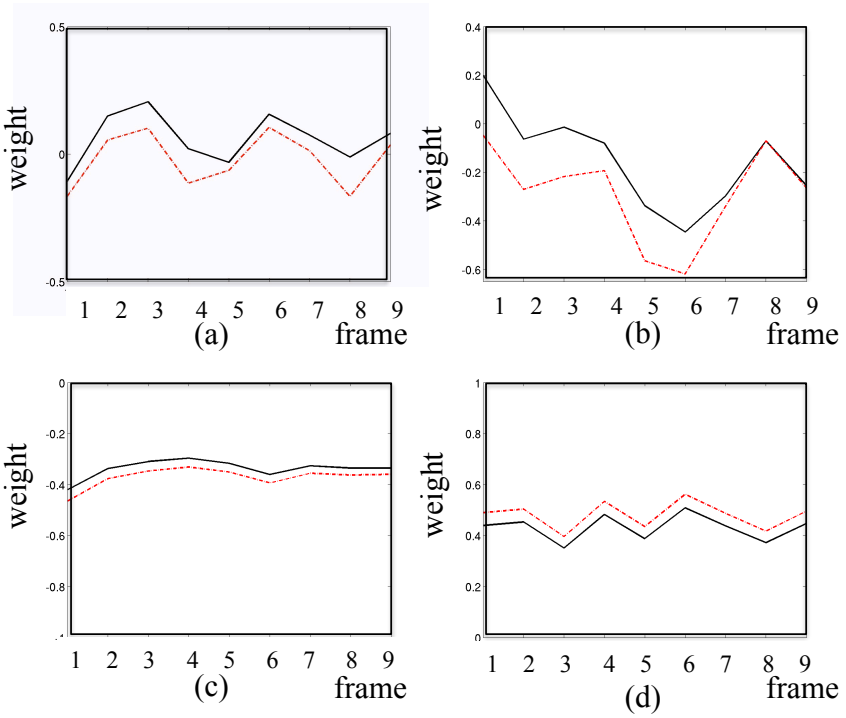

Figure 4.8. Initial weight (black line) and final weight (red dashed line) learned from 2 hidden nodes for 9 frames of a MFCC (C1) (above: (a) and (b)) and a PNS-Gabor input (below: (c) and (d))

Similar characteristic can be observed for the following layers in our experiments. As both PNS-Gabor input and the following neural network transformation is more invariant, PSN-Gabor DNN is better than MFCC-DNN in a mismatched scenario.

4.6. Summary

In this chapter, we exploited the method of simulation and resampling to investigate the success and failings of deep neural network features in different train/test scenarios. Diagnosis shows that DNN-based feature representation is indeed superior to the corresponding MFCC in both two matched scenarios where the incorrect independent assumption of HMM dominates recognition errors. However, evidence from simulation and resampling experiments reveals that the MFCC-DNN feature is easily specialized to training data and the observation mismatch dominates the source of recognition errors in mismatched scenario. On the other hand, DNN-based features that use PNS-Gabor, perform nearly identically to MFCC-DNN features in the matched scenarios and much better than MFCCs and MFCC-DNNs in mismatched scenario. Thus, modeling and robustness are the key steps to improve ASR performance using deep neural network.

Chapter 5

Feature Designs for Speaking Rate Variability

In Chapter 3 and 4, we describe the robustness of word recognition in the presence of noise with the use of a Gabor model. Here, we focus on modeling human perception that is robust to variability of speaking rate. To quantitatively measure speaking rate robustness, we used UCLA CVC (consonant-vowel-consonant) stimuli, which are uttered quickly and more slowly, and conducted perceptual tests for clean and noisy versions. By conducting tests on the data, we investigated if inclusion of Gabor-filtered spectrograms with lower or higher temporal modulations could be used to correlate better with human perception. Our results in this chapter confirmed an improvement in this correlation, while also improving the accuracy.

Unlike our previous large vocabulary continuous speech recognition results, our source material was much simpler (CVC syllables, recorded at UCLA), so that we could observe the distribution of accuracies in order to derive correlations with human perception. Similar to the analytical experiments in Chapter 4, the specific goal of these experiments was not to improve ASR, but rather to observe whether these specific modifications of a standard ASR signal representation would improve the correlations with measures of hearing. Again, we focus on the physiologically inspired representations using a Gabor approach to modulation processing as described in Chapter 3.

5.1. CVC Data Collection

In this section, we describe the collection of the UCLA CVC stimuli[1]. A set of 36 CVC phonetically balanced syllables that incorporated 13 consonants and 3 vowels was selected. Recordings took place at UCLA using an AKG C-410 head mounted microphone in a soundproof room, with two speakers (one male and one female). Each CVC was repeated twice by each speaker. Babble noise from the Noisex database [73] was added to the CVCs to prepare noisy stimuli. The SNR was calculated by using the average SNR level over the speech-only segment, which was then used to determine the

[1] The data collection is done by Anirudh Raju, Abeer Alwan and Jody Kreiman at UCLA Speech Processing and Auditory Perception laboratory

noise power to be added. Each stimulus was prefixed with 100 ms of pure noise (at the noisy power calculated in the previous step) in order to enable listeners to adapt to the noise environment. Stimuli were generated corresponding to 6 conditions: 3 SNRs (quiet, 5 dB, 0 dB) x 2 speaking rates (slow, fast). Given the two speakers and the two repetitions, this yielded (36x2x2x2x3) = 864 utterances for both perceptual and ASR experiments.

Listening experiments were conducted with 52 subjects, in a soundproof booth at UCLA using the stimuli described above. The subjects would hear the set of 864 CVC stimuli (36 syllables x 2 speaker x 2 speaking rates x 3 noise levels x 2 repetitions) over two sessions of one hour each, corresponding to 432 stimuli per session. The stimuli were played back to back, and the subjects were given a 3 second window between the stimuli in order to respond. They were asked to repeat back the stimulus that they heard. A short break of 10 seconds was given after every 20 stimuli. Two phonetically trained linguists transcribed these manually. The consonant accuracies from the subjects are reported in Table 5.1.

For analysis of the perceptual data, effects were observed for each of the generation factors, but for the purpose of this report, the focus will be on the speaking rate characteristics over the range of consonants.

Human	clean	SNR5	SNR0
overall	95.0	71.4	56.5
slow	95.5	74.7	60.0
fast	94.4	68.1	53.0

Table 5.1, Consonant accuracy from the subjects for clean and noisy condition

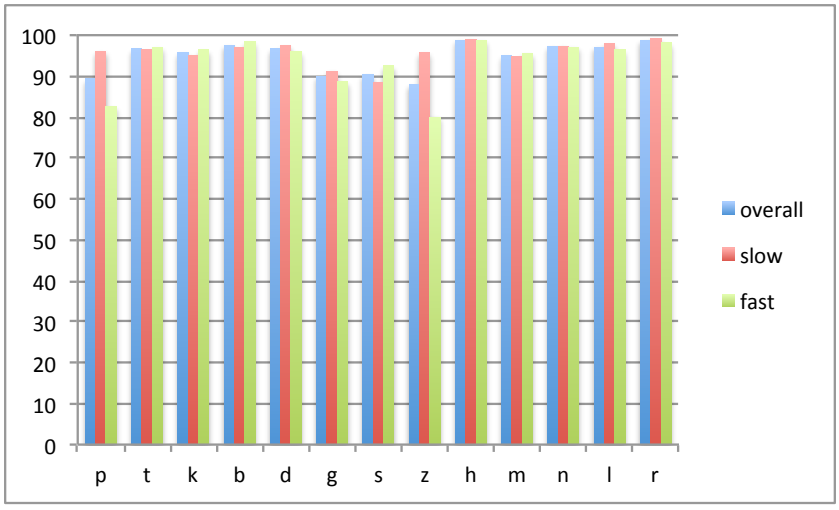

Figure 5.1, Accuracy for each consonant in clean condition

5.2. CVC Recognizer

For the ASR experiments, we used a neural network to generate features, and HMM/GMM tandem system for the modeling of the CVCs. Given the limited amount of CVC data, we primarily used 51 hours of read Wall Street Journal speech (22,092 utterances) for training of both the neural network feature generator and the GMM/HMM acoustic models. Each triphone is modeled using a 3-state HMM. The resulting triphone states are clustered using a decision tree to 5000 tied states where the output distribution for each tied state is modeled with 32 mixtures of multivariate Gaussian with diagonal covariance. Given the properties of the artificial data, we built the constraint word net as shown in Fig. 5.2 to reduce the searching space while decoding.

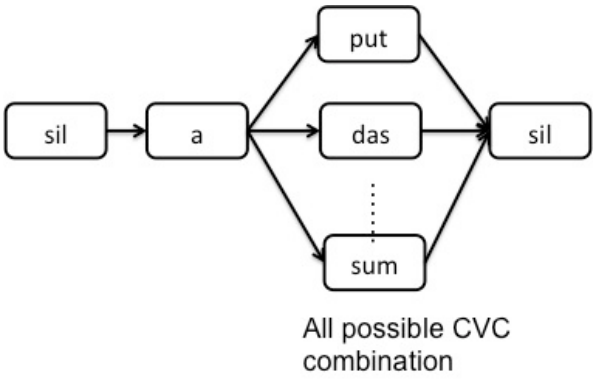

Figure 5.2, Word net for decoding CVC data

Given that the WSJ data is very different in character from CVC syllables, we further adapted the neural networks to 471 of the CVC utterances (unused for other testing). Testing was then done on an independent set of 864 CVC utterances. Results for recognition of the different consonants were then compared to the UCLA perceptual results for the different front-end models that were considered, and for differing noise levels and speaking rates.

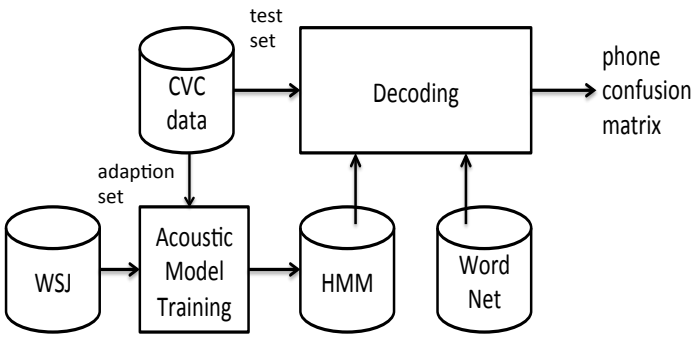

Figure 5.3, Classification system for CVC data

5.3. Multi-Stream Processing for Gabor Features

5.3.1. Temporal Division

Unlike the single Gabor feature stream processing used in the previous chapters, we split the filter outputs into multiple feature streams based on the range of temporal modulation. The temporal division allowed us to extract feature streams of slow or fast-modulated spectrum for the purpose of ASR for the given speaking rates. For our design, 7 ranges of temporal modulations, including 4 in the range of from 0 to 6.2 Hz (centered at 0, 2.4, 3.9, and 6.2 Hz) and 3 from 9.9 to 25 Hz (centered at 9.9, 15.7, and 25 Hz) were used where the former set is called low modulation streams while the latter set is referred as high modulation streams. For each feature stream, there were 9 spectral modulations ranging from -0.25 to 0.25 cycles per channel leading to a total of 125 inputs (as shown in Table 1 below). The temporal windows of high modulation filters range from 17 to 26 frames. The temporal windows for low modulation filters range from 28 to 99 frames.

For the high modulation stream, "skinny" filters capture the fast time-varying part of the spectrum. For the low modulation stream, "fat" filters capture the coarse representation of speech dynamic. For each of the two streams, a variety of "tall" and "short" filters corresponding to different spectral modulation frequencies generate features capturing different representations of spectral dynamics.

Low modulation frequency (Hz)	0, 2.4, 3.9, 6.2
High modulation frequency (Hz)	9.9, 15.7, 25

Table 5.2: Temporal division of Gabor feature streams

Spectral modulation frequency (cycle/channel)	Number of outputs
0.25, -0.25	40
0.12, -0.12	13
0.06, -0.06	5
0.03, -0.03	3
0	3

Table 5.3: Number of features for each spectral modulation frequency

5.3.2. Combination of Temporal Modulation Streams

The output of each of these temporal modulation streams is processed by a neural network that has been trained to discriminate between phonetic classes. The neural network outputs can then be interpreted as posterior probabilities for these classes and combined in a controlled fashion.

Previously, several neural network fusions approaches were studied including the arithmetic, geometric, and harmonic means of the posterior outputs. Beyond these static methods, we applied the dynamic weighting method where the weights that are computed dynamically at each frame in order to take advantage of the properties of the different feature streams for each new acoustic situation. Here we used an inverse entropy combination [77]. For each stream s, an entropy of the output posteriors at frame i, $entropy_{s,i}$ can be calculated as:

$$entropy_{s,i} = \sum_j p_s(c_j \mid o_i) \log p(c_j \mid o_i) \tag{5.1}$$

When a posterior provide an accurate probability estimation of a given phone, the entropy is small. Ideally, target distribution consists of a 1.0 for a phone and 0.0 for the others. Thus, inverse entropy can be used as an indicator to approximate the reliability of the estimation. We use inverse entropy weighting to highlight the informative streams and deny the others. The weight for stream s at frame i, $w_{s,i}$, is calculated as

$$w_{s,i} = \frac{1/entropy_{s,i}}{\sum_{s'} 1/entropy_{s',i}} \tag{5.2}$$

As is typical in tandem processing, the weighed posterior phone probabilities are further processed by logarithm and principle component analysis (PCA) where the logarithm transformation makes the distribution more Gaussian like while PCA performs de-correlation and dimension reduction for diagonal Gaussian distribution. In the following experiments, we combined low, high or all temporal modulation streams to test CVC syllables uttered in slow and high speaking rates as shown in Fig. 5.4. The results confirmed the intuition in our feature design where high/low modulated features prefer a rapid/slow speaking rate respectively.

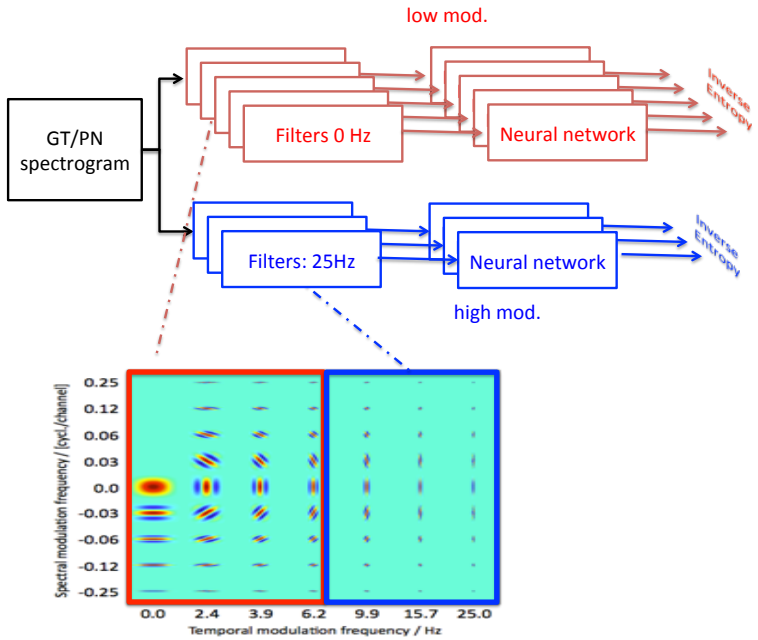

Figure 5.4, Combination of neural network outputs using inverse entropy. Low temporal modulation streams (red) and high temporal modulation streams are split.

5.4. Results and Discussions

In the tandem processing, neural networks are trained to generate features for a GMM/HMM system. However, we used only a single-hidden-layer MLP rather than deeper network to avoid overfitting given the difference between the large WSJ set and the small CVC syllable set. For the MLP inputs, we compare three different features. For the baseline system, MFCCs, which model some basic properties of hearing is used. For the Gabor features, we applied both the gammatone and the power normalized spectrum processed with Gabor filters. MFCCs are appended to MLP outputs as conventional fashion.

Here we focus on the classification accuracies in clean conditions. The results of noisy conditions show a similar trend but much worse accuracies because of the limited amount of training data. Table 5.4, 5.5 and 5.6 demonstrate the results of consonant accuracies and the correlations with human perception for overall (288 utterances), slow (144 utterances) and rapid (144 utterances) speech. The results show that the PN-Gabor MLP performs better than MFCC-MLP for both consonant accuracy and correlation with perception in all conditions.

Front end method	Machine consonant accuracy in clean condition	Correlation with perception in clean condition
MFCC + MFCC-MLP	69.5%	0.91
MFCC + GT-Gabor-MLP	67.0%	0.89
MFCC + PN-Gabor-MLP	73.4%	0.93

Table 5.4, Front end effects for clean CVCs, all speech

Front end method	Machine consonant accuracy in clean, slow speech	Correlation with perception in clean, slow speech
MFCC + MFCC-MLP	68.4%	0.88
MFCC + GT-Gabor-MLP	68.9%	0.87
MFCC + PN-Gabor-MLP	74.9%	0.92

Table 5.5, Front end effects for clean CVCs, slow speech

Front end method	Machine consonant accuracy in clean, rapid speech	Correlation with perception in clean, rapid speech
MFCC + MFCC-MLP	70.5%	0.91
MFCC + GT-Gabor-MLP	65.1%	0.87
MFCC + PN-Gabor-MLP	71.9%	0.92

Table 5.6, Front end effects for clean CVCs, rapid speech

Testing condition	Machine consonant accuracy, GT-Gabor MLP low modulations only	Machine consonant accuracy, GT-Gabor MLP high modulations only
Clean, rapid speech	51.1%	70.3%
Clean, slow speech	63.3%	67.2%
Clean, all speech	57.2%	68.8%

Table 5.7: Comparison between using low modulation filters and only the high ones for slow and rapid speech, using the GT-Gabor MLP front end, consonant accuracies.

Testing condition	Machine correlation with perception, GT-Gabor MLP low modulations only	Machine correlation with perception, GT-Gabor MLP high modulations only
Clean, rapid speech	0.68	0.88
Clean, slow speech	0.81	0.81
Clean, all speech	0.74	0.87

Table 5.8: Comparison between using low modulation filters and only the high ones for slow and rapid speech, using the GT-Gabor MLP front end, correlation with perception.

To further investigate the effect of modulation design on recognition performance, we split GT-Gabor streams into low (0.0-6.2 Hz) and high (9.9-25 Hz) temporal modulation features and conducted the same experiments for both cases as well. Here, we selected GT-Gabor for the analyses since the medium-duration spectrum subtraction step in the PN spectrum could smear the results. For the cases of low or high modulations, the total number of trained parameters was significantly lower, since only some of the MLPs were used and the number of hidden units was kept constant.

In general, the high modulations are better in all conditions, as highlighted by Tables 5.7 and 5.8. The behavior could be due to the fact that MFCC features to which the modulation features are appended are essentially broad in modulation range, and so emphasizing the higher modulations may help to boost an important region. While high modulations are consistently better than low modulations, it appears that Gabor filters corresponding to low temporal modulation frequencies gave both higher consonantal phoneme accuracy and better correlations with human consonantal phoneme accuracies for slower speech than they did for rapid speech. Similarly, higher temporal modulation frequencies corresponded to better phoneme recognition for rapid speech than for slow speech, and also had a better correlation with the perceptual phoneme accuracies. Also, we observe that high modulations alone (70.3%) lead to better accuracies than the entire feature streams (65.1%) for rapid speech. Therefore, neural networks cannot handle the reduction of low modulations automatically without some design effect to detect speaking rate.

5.5. Summary

For most of the tests, any strategy that improved consonant accuracy also yielded a better correlation with human performance. However, all of the measured ASR accuracies were far lower than the human perceptual case. As we had expected, the experiments show that low (high) temporal modulations were indeed more effective in recognizing slow (rapid) speech. Also, reduction of low modulations manually leads to better results for rapid speech that were not achieved by a neural network automatically without some design effect to detect speaking rate.

Chapter 6

Phone Recognition for Mixed Speech Signal Using Human Cortical Signal

In addition to the robustness to noise and speaking rate variability, another unique and defining property of human speech perception is the ability to robustly process speech sounds in the context of interference-filled acoustic conditions. A common condition is the multi-speaker environment, where selective listening is required. Humans are very successful at understanding speech in this condition even when spatial information is limited or missing, which is called the "cocktail party" effect. For example, videos often include background music mixed with spoken language, and yet the speech is understandable by the viewer.

In this Chapter, we study cortical signals involved in auditory source separation for mixed single-channel speech signals, using neuroelectric responses directly measured from the surface of the human cortex using a 256-electrode array, giving a view of cortical sound processing of unprecedented detail and flexibility. While human beings can attend to a single sound source within a mixed signal from multiple sources, the automatic speech recognition without the benefit of effective blind source separation is quite poor at this task. Here we report on the analysis of human cortical signals to demonstrate the relative robustness of these signals to the mixed signal phenomenon, which is contrasted to a deep neural network-based ASR system.

6.1. Neural Data

The acoustic and neural data used in this study were previously generated in sessions with surgery patients as described in [44]. The patients had customized high-density electrode arrays implanted subdurally that are sometimes necessary for the surgical management of patients with epilepsy refractory to medications [10] (see Figure 6.1). They participated in behavioral testing using stimuli from the Coordinate Response Measure (CRM) corpus [7]. The corpus consists of phrases of the form "Ready (call sign) go to (color) (number) now" spoken with different combinations of call signs ("Tiger" or "Ringo"), 3 colors ("Blue", "Green", "Red"), 3 numbers ("Two", "Five", Seven") and two speakers. The patients were instructed to report the color and number associated with a call sign (e.g. "Tiger"); however, they did not know *a priori* which speaker will be the target in each trial. Consequently, subjects are required to attend to both speakers at the

beginning of the mixture until they heard the target call sign and then attended to the corresponding speaker. At the end of each experiment block, the patients had responded to the same 28 sound mixtures while attending to both speakers. This experimental design allowed us to determine the effect of attention on neural responses, while controlling for identical acoustic stimulus conditions (i.e., hearing the same speaker mixture, while attending to only one or the other voice). For the purposes of this study, we have been working with data collected from three subjects, but since neural responses are highly individual, all the results reported here are from a single individual.

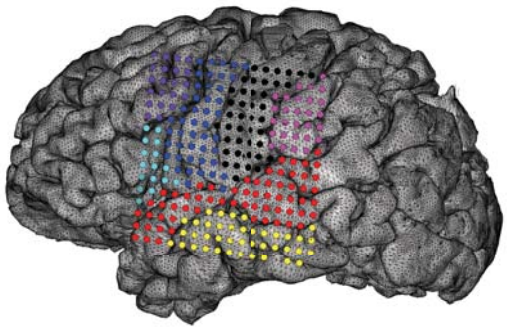

Figure 6.1: MRI reconstruction of subject's cortex, with electrocorticogram (ECoG) electrodes (16x16 grid, 4 mm spacing) indicated by dots. The red and yellow regions (temporal lobe) are expected to contribute most to speech processing.

Given the extremely modest amount of data (e.g., 3115 phones) from the CRM experiments, we also used a subset of TIMIT to augment the acoustic training set. As described below, we also used a jackknifing technique [19] to make better use of the limited amount of data.

6.2. Neural Feature Extraction

In this section, we describe the methods to exploit a good representation from raw human cortical signals. While the usual signal representations for phone or speech recognition are acoustic features, we also attempt to decode the phone sequence directly from *neural features*[2] that serves as a suitably preprocessed input to the recognizer. Fig. 6.2 demonstrates an example of raw cortical signal from one electrode.

[2] The neural feature extraction and dimension reduction described in 6.2 and 6.3 are done by Erik Edwards from UCSF

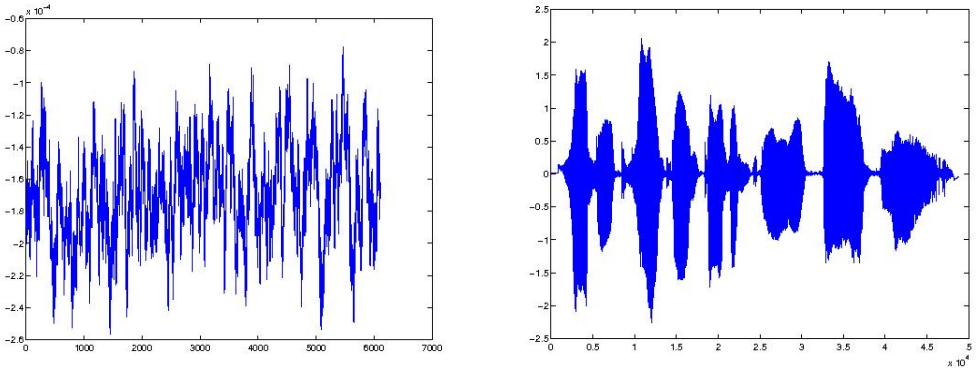

Figure 6.2, Example of raw cortical signal (left) from one electrode and the corresponding audio signal (right)

The feature generation process can be divided into preprocessing and dimension reduction. We started with raw cortical signal collected from 256 electrodes. Next, 256-d raw neural features focusing on a particular frequency band are generated. The dimension reduced feature was then used for hybrid HMM/DNN phone recognition system.

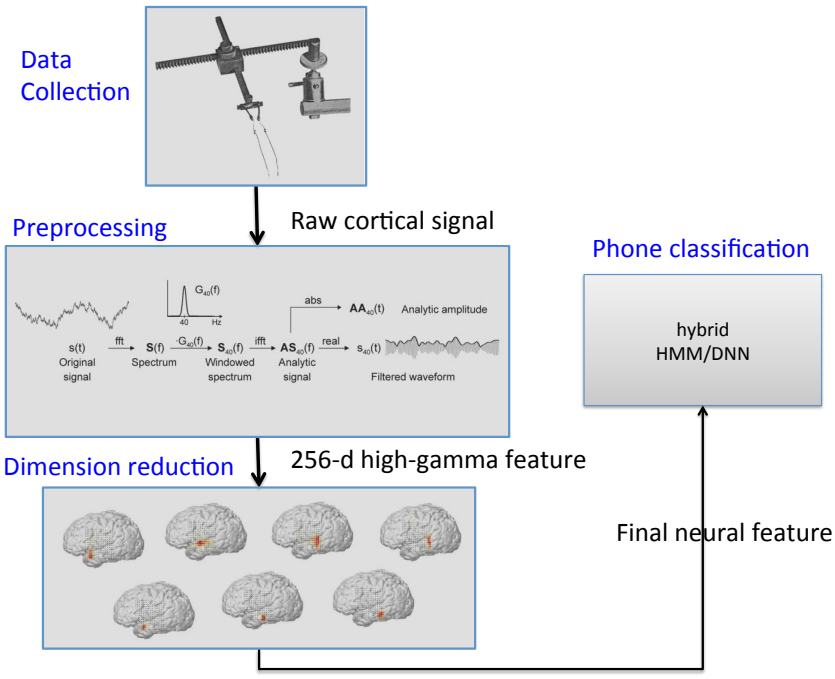

Figure 6.3, Neural feature generation process

6.2.1. Neural Features from High-Gamma Band

Despite advanced in neural research, the question to suitably process the raw, often noisy, broadband ECoG signal for optimal performance in the recognizer is still poorly understood. First, some degree of noise is unavoidable in the present clinical context, and some epochs and electrodes are rejected after human examination [as in 44]. Second, although human brain waves have been known since 1929, it was not until relatively recently [14, 15] that it became apparent that the information available in the so-called "high-gamma" band (~70-170 Hz) carries information of far greater spatial and temporal specificity compared to the more traditionally studied frequency ranges below 60 Hz [17]. Although we have not ruled out the use of lower frequencies as auxiliary information [e.g., 80], the present study uses only the ECoG data in the high-gamma range. Specifically, we sum over Hilbert envelopes [4] for center frequencies 70-170 Hz [as in 17, 18], and down-sample to 100 Hz. Analytic amplitudes are Rayleigh distribution and heteroscedastic [3, 6], so we further take the natural log to render Normal distribution and to stabilize variance [60, 63]. The end result of preprocessing is a set of 256 high-gamma time series as shown in Fig. 6.4, one for each retained electrode.

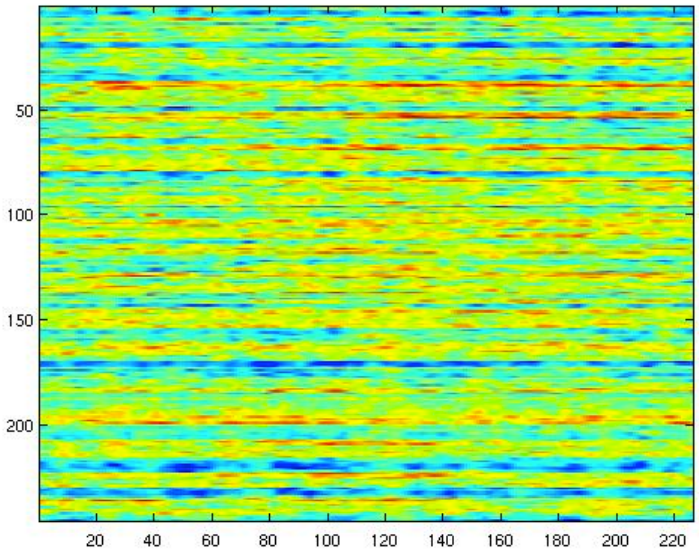

Figure 6.4, 256-d high gamma feature

6.2.2. Dimension Reduction

With small training sets, DNNs may be prone to overfitting, so dimensionality reduction to 48 neural features was deemed necessary. We then explored several approaches to reduce the single-electrode features to 48 neural features.

The most obvious, and nearly the oldest, method of dimensionality reduction is spatial principal component analysis. However, the resulting components lack any physiological significance and also serve poorly in a pattern recognition sense. This problem has been known for nearly as long as spatial PCA itself [34, 43], and the most frequent solution has been to *rotate* the components so as to achieve sparsity, locality, clustering by similarity, or some other objective. After extensive study of available rotation methods, component analyses, hard and soft clustering approaches, and various other machine learning methods surrounding embedding and dimensionality reduction, it was surprising to find that the old method of *varimax rotation* [36, 65] performed as well or better than any of the several modern methods tried. This becomes less surprising when one considers that such rotation is the old L2-norm method to achieve sparsity (called "simple structure" in factor analysis), co-sparsity [20], within-cluster smoothness, across-cluster decorrelation, and greater physical plausibility; all of these known to be virtues for pattern recognition features [e.g., 26]. As an L2 method, it is also extremely fast.

Convex non-negative matrix factorization (NMF) [16] was found to slightly improve the results taking the varimax components as initializing input. Neither ordinary NMF [42], nor convex NMF without a good initialization, performed as well as varimax for our purposes. The requirement of a good initialization is the known drawback of NMF. Intuitively, convex NMF achieves the clustering objective that correlated electrodes should cluster together, whereas the resulting cluster time series should be as decorrelated as possible.

The final neural features used here are thus 48 convex NMF components derived from varimax rotation of spatial PCA analysis of 256 log-high-gamma time series (see Figure 6.5).

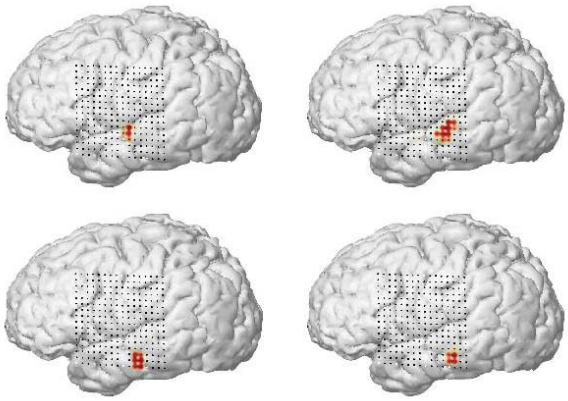

Figure 6.5: Spatial weightings for 4 out of 48 of the convex NMF neural features used. These are examples, chosen for their loadings onto temporal lobe sites important for speech processing.

6.3. Phone Recognition System

For both neural and acoustic observations, we exploited the hybrid HMM/DNN. We used the Kaldi toolkit [61] for both model training and decoding, as well as for the DNN processing. The hybrid HMM/ANN set up was adapted from Kaldi recipe s5.

The inputs of the DNNs were obtained from splicing 39-d MFCCs or 48-d neural features across 17 frames, followed by reducing the dimension to 250 using linear discriminant analysis. Mean and variance normalization were performed for both MFCCs and neural features. The DNNs had 2 hidden layers, each of which consists of 1100 *tanh* units. The output layer consisted of 117 context-independent phonetic states (three states per phoneme), giving 1.6 M parameters in total. Frame-level forced alignment was provided by a simple context-independent HMM/GMM system. The DNNs were trained with stochastic gradient descent, starting with a learning rate of 0.015 and ending at 0.002.

During training we decreased them by a factor of 1.14, except for 5 epochs at the end during which we kept them constant. The network was trained for a total of 20 epochs. A biphone language model was estimated on the training set.

The acoustic recognition scenario gave a 26.9% phone error rate for the standard TIMIT train-test set, where 3696 utterances were included for training and 192 utterances for testing. Given the constraint of ECoG neural data, we set up our train-test sets using only 374 utterances from TIMIT, 155 utterances from single source CRM data and 61 utterances from mixed source CRM data.

Due to the small amount of data, we used the jackknife resampling process [19], splitting the set of CRM data into 5 different train-test cuts. For each of the training sets, we randomly drew samples from the CRM data and augmented them with 374 utterances of the TIMIT set. The rest of the CRM sets were used for testing. This yielded 7,520 instances of phones for single source test sets and 3,115 instances for mixed source test sets. The results reported here are average score over the 5 different train-test sets.

6.4. Results

Table 6.1 shows that the phone error rate for the acoustic features, while far better than that achieved with our neural signals for the single voice case, degrades greatly for the mixed signal. For the neural features, the error rates overall are quite high, but there is very little additional degradation when the subject is motivated to attend to the desired voice.

The primary hypothesis being tested was, given specific neural signals, and given the chosen features extracted from these signals, that the phone recognition error rates would be affected far less for the neural signals than for the acoustic signals. While it is a common experience that humans do better in the cocktail party scenario than our current machine methods, what was being tested here was the utility of the specific brain signals that we were measuring to show this phenomenon. Furthermore, the experimentation with feature extraction methods given the raw data had begun to show us what aspects of the STG neural signals were most effective for phone recognition.

	Acoustic features	Neural features
Single source	48.6%	68.2%
Mixed source	73.2%	70.5%

Table 6.1: Phone error rate for complete CRM utterances

It must also be recognized that we were not yet attempting to build sophisticated systems for this purpose; we were making no use of standard enhancement or blind source separation algorithms. Consequently all error rates are quite high. Nonetheless, the observed effects are quite striking: the neural features yield nearly the same error rates for single and mixed sources, when the acoustic features are much less informative for the mixed source case.

As shown in tables 6.2 and 6.3, we can observe similar trends for the phone error rates in target words (color and number, respectively). Phone error rates for acoustic features rise significantly for the mixed signal case, but are nearly the same using neural features.

	Acoustic features	Neural features
Single source	54.0%	72.0%
Mixed source	75.2%	73.1%

Table 6.2: Phone error rates within the color target word (red, green, or blue)

	Acoustic features	Neural features
Single source	56.1%	73.5%
Mixed source	79.9%	73.9%

Table 6.3: Phone error rates within the number target word (two, five, or seven)

6.5. Summary

The work reported here confirmed that the STG can provide meaningful information for phonetic recognition, at least for the task used, that can be relatively independent of interfering voices that the subject is not paying attention to. This is in contrast to the observed increases in error rate for phone recognition given such interfering signals.

Chapter 7

Conclusion

7.1. Contributions

In this thesis, we applied Gabor filter modeling some properties of "natural" human auditory processing. This is instrumental in improving generalization to unanticipated deviations from what was seen in training. We demonstrated that integration of Gabor models as DNN input or convolutional kernels can improve accuracy on speech recognition for unanticipated types of input including noise, channel distortion, reverberation and speaking rate variation.

In Chapter 3, we propose the features that integrate Gabor filters and a fully connected DNN or CNN. We first employed a more robust PN spectrum incorporating key parts of the PNCC algorithm. The Gabor features based on PN spectrum are then used as input for DNN. Further improvement can be obtained by feature selection via sparse PCA. Finally, we showed that a modified CNN could learn features with multiple temporal and spectral resolutions by making use of Gabor kernels. The proposed feature showed around 30% improvement relative to MFCC-DNN for noise-robust speech recognition.

In Chapter 4, we exploited the bootstrap sampling method to investigate the model residual of DNN features based on Gabor and MFCC in different train/test scenarios. For both matched near-field or far-field recordings, diagnostic analysis showed that the DNN accounts for most of the improvement where the incorrect independent assumption of HMM dominates recognition errors. However, the result revealed that the MFCC-DNN feature is easily specialized to training data and the observation mismatch dominates the source of recognition errors in mismatched scenario. On the other hand, a DNN-based feature that uses Gabor, performs nearly identical to MFCC-DNN features in the matched scenarios and much better than MFCCs and MFCC-DNNs in mismatched scenario. Thus, we conclude that modeling and robust feature designs are the key steps to improve ASR performance using a DNN.

In Chapter 5, we explored fast and slow modulation Gabor features to model human hearing of rapid and slow speech. We divided the temporal modulation range into low (0.0-2.5 Hz) and high (9.9-25 Hz) regions. Experiments revealed that a low temporal modulated feature shows better recognition accuracies and correlation to human hearing on slow speech and much worse on rapid speech. Similarly, a high temporal modulation feature is more correlated to human perception on rapid speech and worse for slow speech. Also, because reduction of low modulation features manually improves the

performance on rapid speech, the neural network didn't handle the variability in speaking rate automatically.

In Chapter 6, we developed the phone recognition system under overlapped speech based on human cortical signals. The work reported confirmed that the STG can provide some information for phonetic recognition, at least for the task used, which can be relatively independent of interfering voices that the subject is not paying attention to. This is in contrast to the huge increases observed in error rate for phone recognition given such interfering signals.

Therefore, we conclude that neural network trained features are not enough for robust speech recognition. The designed feature based on human hearing system is useful for greater robustness than just relying on DNN or CNN with a number of example tasks in the thesis.

7.2. Future work

The work on GCNN in Chapter 3 has shown the basic effectiveness of using Gabor to expand temporal and frequency resolutions of CNNs. Some research (e.g. [23]) has demonstrated that LSTMs are complementary to CNNs as they have better modeling capabilities to temporal information. While both LSTMs and Gabor handle temporal modeling problem, Gabor also provides better frequency resolutions and initial coefficients based on a biological model. It's likely the improvement can be achieved by unified architecture integrating GCNNs and LSTMs. For example, we could apply Gabor convolutional layer and a few other convolutional layers and pass the pooled activations to LSTM layers. Also, the spectral modulation and temporal modulation frequencies used in Gabor filters here are tuned beforehand. These parameters could be implemented as part of neural network architecture and trained by back-propagation as well.

In Chapter 4, we performed bootstrap sampling on only HMM-based acoustic models. A more sophisticated neural network architectures (e.g. LSTM) using Connectionist Temporal Classification (CTC) implementation could be trained with an unsegmented sequence providing a better choice over HMM. It is reasonable to conduct similar statistical analysis on the effect of model residual on recognition accuracy. Furthermore, some ongoing researches work is focused on building an end-to-end neural network based speech recognition system including an acoustic model, a pronunciation model and language model. With the end-to-end system, statistical analysis would not be constrained to a smaller model such as acoustic model here.

The experiments in Chapter 5 and 6 are limited by the amount of training data. These experiments could be repeated with far more training data. Furthermore, a follow-up experiment in Chapter 5 should incorporate adaptation of the GMMs. While this would also be expected to improve the baseline results, our experience with neural networks suggests that they could be even more susceptible to overfitting an insufficient amount of training data. For the experiments on the human cortical signal, our final goal in is to learn more about how the brain is recognizes speech when competing signals (particularly multiple voices) are present. But our other goal is to learn from this exploration how we can improve automatic speech recognition in the presence of mixed

signals, we will have to use partial information (for instance from comparison of error patterns) to modify our artificial systems to act more like the natural one. We are just beginning this process, and the results in Chapter 6 reports the initial effort to simultaneously study the scientific and engineering components.

Bibliography

[1] O. Abdel-Hamid, L. Deng, and D. Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. INTERSPEECH. 2013.
[2] K. Walker and S. Strassel. The rats radio traffic collection system. In Proc. of ISCA Odyssey, 2012.
[3] R. Arens. Complex processes for envelopes of normal noise. IRE Trans Inf Theory 3(3): 204-7, 1957.
[4] E.J. Baghdady. Diversity techniques. In: Baghdady EJ, ed. Lectures on communication system theory. New York: McGraw-Hill: 125-75, 1961.
[5] J.K. Baker. The DRAGON system – an overview. IEEE Transactions on Acous- tics, Speech, and Signal Processing, 23 (1), 24-29, 1975.
[6] J.S. Bendat, A.G. Piersol. Random data: analysis and measurement procedures, 3rd Ed. New York: J. Wiley, 2000.
[7] R.S. Bolia, WT Nelson, MA Ericson, BD Simpson. A speech corpus for multitalker communications research. The Journal of the Acoustical Society of America 107, 1065, 2000.
[8] H. Bourlard and C. Wellekens. Links between Markov models and multilayer perceptrons. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12 (12), 1167–1178, 1990.
[9] H. Bourlard and N. Morgan. Connectionist Speech Recognition: A Hybrid Approach. Kluwer Press, 1993
[10] E. Chang, J. Rieger, K. Johnson, M. Berger, N. Barbaro, R. Knight. Categorical speech representation in human superior temporal gyrus. Nature Neuroscience, 2010.
[11] B.Y. Chen, Q. Zhu, and N. Morgan. A Neural Network for Learning Long-Term Temporal Features for Speech Recognition. Proc. ICASSP 2005, March 2005, pp. 945-948
[12] G. Chen, H, Xu, M. Wu, D. Povey and S. Khudanpur. Pronunciation and silence probability modeling for ASR. In Proceedings of INTERSPEECH, 2015.
[13] T. Chi, Y. Gao, M. Guyton, P. Ru and S. Shamma. Spectro-temporal modulation transfer functions and speech intelligibility. The Journal of the Acoustical Society of America, 106(5), 2719-2732, 1999.
[14] N.E. Crone, D.L. Miglioretti, B. Gordon, R.P. Lesser. Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis. II. Event-related synchronization in the gamma band. Brain 121(Pt 12): 2301-15, 1998.
[15] N.E. Crone, D. Boatman, B. Gordon, L. Hao. Induced electrocorticographic gamma activity during auditory perception. Clin Neurophysiol 112(4): 565-82, 2001.
[16] C. Ding, T. Li, M.I. Jordan. Convex and semi-nonnegative matrix factorizations. IEEE Trans Pattern Anal Mach Intell 32(1): 45- 55, 2010.
[17] E. Edwards. Electrocortical activation and human brain mapping. Dept. of Psychology dissertation. Berkeley: Univ. of California: 147 p, 2007.

[18] E. Edwards, M. Soltani, W. Kim, S.S. Dalal, S.S. Nagarajan, M.S. Berger, R.T. Knight. Comparison of time-frequency responses and the event-related potential to auditory speech stimuli in human cortex. J Neurophysiol 102(1): 377-86, 2009.
[19] B. Efron. The jackknife, the bootstrap, and other resampling plans. Society of Industrial and Applied Mathematics CBMS-NSF Monographs, 38, 1982.
[20] M. Elad. Sparse and redundant representation modeling – what next? IEEE Signal Process Lett 19(12): 922-8, 2012.
[21] D. Ellis. Renoiser web page. http://labrosa.ee.columbia.edu/projects/renoiser/create_wsj.html
[22] D. Gillick, L. Gillick, and S. Wegmann. Dont Multiply Lightly: Quantifying Problems with the Acoustic Model Assumptions in Speech Recognition. In Proceedings of ASRU, pp. 71–76, IEEE, 2011
[23] A. Graves, A.R. Mohamed and G. Hinton. Speech recognition with deep recurrent neural networks. IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 6645-6649), 2013.
[24] F. Grezl and P. Fousek. Optimizing bottle-neck features for LVCSR. Proc. ICASSP 2008, pp. 4729-4732
[25] A. Gunawardana, M. Mahajan, A. Acero and J.C. Platt. Hidden conditional random fields for phone classification. In INTERSPEECH (pp. 1117-1120), 2005.
[26] M.A. Hall. Correlation-based feature selection for machine learning. Dept. of Computer Science. Hamilton, New Zealand: University of Waikato: 178 p., 1999.
[27] H. Hermansky and S. Sharma. Temporal patterns (TRAPs) in ASR of noisy speech," Proc. ICASSP 1999, March 1999, pp. 289-292 vol. 1.
[28] H. Hermansky, D. P. W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional HMM systems. In Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2000, vol. 3, pp. 1635–1638.
[29] G. Hinton, and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313, 504–507, 2006.
[30] H.G. Hirsch, & D. Pearce. The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW), 2000.
[31] Y. Hoshen, R.J. Weiss & K.W. Wilson. Speech acoustic modeling from raw multichannel waveforms. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on (pp. 4624-4628). IEEE.
[32] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. The ICSI meeting corpus. Proceedings of International Conference on Acoustics, Speech, and Signal Processing, 2003.
[33] F. Jelinek. Continuous speech recognition by statistical methods. IEEE Proceed- ings, 64 (4), 532-556, 1967.
[34] I.T. Jolliffe. Principal component analysis, 2nd Ed. New York: Springer, 2002.
[35] F. Joublin X. Domont, M.Heckmann and C. Goerick. Hierarchical spectro-temporal features for robust speech recognition. Proc. ICASSP 2008, March 2008, pp. 4417-4420
[36] H.F. Kaiser. The varimax criterion for analytic rotation in factor analysis. Psychometrika 23(3): 187-200, 1958.

[37] O. Kalinli, M.L. Seltzer, and A. Acero. Noise adaptive training using a vector Taylor series approach for noise robust automatic speech recognition. In Proc. ICASSP, 2009, pp. 3825-3828

[38] C. Kim and R. M. Stern. Feature extraction for robust speech recognition based on maximizing the sharpness of the power distribution and on power flooring. In Proc. ICASSP, pp. 4574–4577, 2010.

[39] M. Kleinschmidt. Localized spectro-temporal features for automatic speech recognition," in Proc. of Eurospeech, 2003, Sep 2003, pp. 2573–2576.

[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324, 1998.

[41] T. Lee. Image representation using 2D Gabor wavelets. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 18(10), 959-971, 1996.

[42] D.D. Lee, H.S. Seung. Learning the parts of objects by non- negative matrix factorization. Nature 401(6755): 788-91, 1999.

[43] E.N. Lorenz. Empirical orthogonal functions and statistical weather prediction. Statistical Forecasting Project. Scientific Report No. 1. Cambridge, MA: MIT, Dept. of Meteorology: 49 p, 1956.

[44] N. Mesgarani N, E. Chang. Selective cortical representation of attended speaker in multi-talker speech perception. Nature 485: 233- 236, 2012.

[45] N. Mesgarani, and S. Shamma. Speech Processing with a Cortical Representation of Audio", Proc. ICASSP 2011, May 2011, pp. 5872-5875.

[46] N. Mesgarani, M. Slaney, and S. A. Shamma. Discrimination of Speech From Nonspeech Based on Multiscale Spectro- Temporal Modulations", IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 3, May 2006.

[47] N. Mesgarani, C. Cheung, K. Johnson, E.F. Chang. Phonetic feature encoding in human superior temporal gyrus. Science 343(6174): 1006-10, 2014.

[48] B. Meyer, C. Spille, B. Kollmeier and N. Morgan. Hooking up spectro-temporal filters with auditory-inspired representations for robust automatic speech recognition", in Proc. Interspeech 2012

[49] T. Mikolov, S. Kombrink, L. Burget, J.H. Černocký, & S. Khudanpur. Extensions of recurrent neural network language model. In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on (pp. 5528-5531). IEEE.

[50] V. Mitra, H. Franco, M. Graciarena, and A. Mandal. Normalized amplitude modulation features for large vocabulary noise-robust speech recognition. Proc. ICASSP 2012, March 2012, pp. 4117-4120

[51] A. Mohamed, D. Yu, and L. Deng. Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition. In Interspeech 2010. pp. 2846-2849.

[52] A. Mohamed, G. Dahl, G. Hinton. Acoustic modeling using deep belief networks. Audio, Speech, and Language Processing, IEEE Transactions on, vol. 20, no. 1, pp. 14 – 22.

[53] A. Mohamed, G. Hinton and G. Penn. Understanding how deep belief networks perform acoustic modelling. In Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on (pp. 4273-4276). IEEE.

[54] N. Morgan, J. Cohen, S. Krishnan, S. Chang and S. Wegmann. Final Report: OUCH Project (Outing Unfortunate Characteristics of HMMs), 2013.

[55] C.S. Myers & L.R. Rabiner. A Comparative Study of Several Dynamic Time‑Warping Algorithms for Connected‑Word Recognition. Bell System Technical Journal, 60(7), 1389-1409, 1981.
[56] N. Naikal, A. Yang, and S. Sastry. Informative feature selection for object recognition via sparse PCA. In Proceedings of the 13th International Conference on Computer Vision (ICCV), 2011, pp. 818-825
[57] V. Nair and Geoffrey Hinton. Rectified linear units improve restricted Boltzmann machines (PDF). ICML, 2010.
[58] N. Parihar, J. Picone, D. Pearce, H.G. Hirsch. Performance analysis of the Aurora large vocabulary baseline system. Proceedings of the European Signal Processing Conference, Vienna, Austria, 2004.
[59] H. Parthasarathi, S.Y. Chang, J. Cohen, N. Morgan and S. Wegmann. The blame game in meeting room ASR: An analysis of feature versus model errors in noisy and mismatched conditions. In Proc, ICASSP pp. 6758-6762, 2013
[60] I. Pitas, A.N. Venetsanopoulos. Nonlinear digital filters: principles and applications. Boston: Kluwer Academic, 1990.
[61] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlícek, P. Qian, P. Schwarz, J. Silovsk, G. Stemmer G, Vesey K (2011). The kaldi speech recognition toolkit. Proc. IEEE 2011 Workshop on ASRU. Dec. 2011, IEEE Signal Processing Society.
[62] Speech Processing, Transmission and Quality Aspects (STQ); Distributed Speech Recognition; Adv. Front-end Feature Extraction Algorithm; Compression Algorithms, ETSI ES 202 050 Ver. 1.1.5, 2007
[63] P.R. Prucnal, E.L. Goldstein. Exact variance-stabilizing transformations for image-signal-dependent Rayleigh and other Weibull noise sources. Appl Opt 26(6): 1038-41, 1987.
[64] S. Ravuri and N. Morgan. Using spectro-temporal features to improve AFE feature extraction for automatic speech recognition. In Proc. ICASSP, 2010, March 2010, pp. 1181– 1184.
[65] M.B. Richman. Rotation of principal components. J Climatol 6(3): 293-335, 1986.
[66] F. Rosenblat, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review 65 (1958), 386-408.
[67] F. Rosenblatt, Principles of Neurodynamics. Spartan Books, Washington D.C., 1962.
[68] T. Sainath, Mohamed, A. R., Kingsbury, B., & Ramabhadran, B. (2013, May). Deep convolutional neural networks for LVCSR. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on (pp. 8614-8618). IEEE.
[69] F. Seide, G. Li, D. Yu. Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. Proc. Interspeech, 2011.
[70] M. Slaney. Auditory toolbox. Interval Research Corporation, Tech. Rep, 10, 1998.
[71] A. Stolcke, X. Anguera, K. Boakye, O. Cetin, A. Janin, M. Magimai-Doss, C. Wooters, and J. Zheng. The SRI-ICSI Spring 2007 Meeting and Lecture Recognition System. In Proceedings of the Second International Workshop on Classification of Events, Activities, and Relationships (CLEAR 2007)
[72] T. Tsai and N. Morgan. Longer Features: They Do a Speech Detector Good. Proc. Interspeech 2012.

[73] A. Varga, and H. Steeneken. Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. Speech Communication, vol. 12, Issue 3, July 1993, pp. 247-251.

[74] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time-delay neural networks. Acoustics, Speech and Signal Processing, IEEE Transactions on 37, no. 3 (1989): 328-339.

[75] D. Yu, L. Deng, F. Seide, and G. Li. Discriminative Pretraining of Deep Neural Networks" (granted 2016, US patent # 9235799)

[76] X. Zhang, M. G. Heinz, I. C. Bruce, and, L. H. Carney. A phenomenological model for the responses of auditory-nerve fibers: I. Nonlinear tuning with compression and suppression. J. Acoust. Soc. Am., vol. 109, no. 2, pp. 648– 670, Feb 2001.

[77] S. Zhao, S. Ravuri, and N. Morgan. Multi-Stream to Many-Stream: Using Spectro-Temporal Features for ASR. Proc. Interspeech 2009, 2951-2954

[78] Q. Zhu, B. Chen, F. Gr´ezl, and N. Morgan. Improved MLP structures for data-driven feature extraction for ASR. In Proc. INTERSPEECH 2005, Lisbon, Portugal, Sept. 2005.

[79] Q. Zhu, B. Chen, N. Morgan, and A. Stolcke, “On using MLP features in LVCSR,” in Proc. Interspeech, 2004, pp. 921–924

[80] G. Zion, N. Ding, S. Bickel, P. Lakatos, C.A. Schevon, G.M. McKhann, R.P. Goodman, R. Emerson, A.D. Mehta, J.Z. Simon, D. Poeppel, C.E. Schroeder. Mechanisms underlying selective neuronal tracking of attended speech at a "cocktail party". Neuron 77(5): 980-91, 2013