Noise-Robust Speech Recognition Using Deep Neural Network

Bo Li

Department of Computer Science
School of Computing

National University of Singapore

A thesis submitted for the degree of

Doctor of Philosophy

2014

NOISE-ROBUST SPEECH RECOGNITION USING DEEP NEURAL NETWORK

BO LI

(B.Eng. NWPU)

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Bo Li 05/09/2014

Bo Li

May 9, 2014

Acknowledgments

First of all, I would like to express the utmost gratitude to my supervisor, Dr. Khe Chai Sim, for his guidance, suggestion and criticism throughout my study in National University of Singapore. His responsibility to students is impressive, which has been invaluable to me. I learned a lot from his strictness in mathematics, strong motivation of concepts and clear logic flow during presentation and writing. The firm requirements and countless guidance on these aspects have given me the ability and confidence to carry out the research work of this thesis as well as the work in future. By initiating well-targeted questions, offering experienced suggestions and having constructive discussions, he is without doubt the most important person that has helped me make this work possible!

Special thanks go to Prof. Steve Renals, Prof. Tan Chew Lim, Assoc. Prof. Wang Ye, Prof. Chua Tat Sen and Prof. Ng Hwee Tou for their invaluable feedbacks and suggestions at different stages of my PhD study. Their insight, experience and wide-range knowledge have benefited me a lot. Besides, I would like to thank Prof. Ng Hwee Tou for providing financial support for my study through the MDA supported CSIDM program. I would also like to thank Dr. Golam Ashraf for his guidance in the first two years of my PhD study. His great passion and thrivingness on challenge and creativity influence me a lot.

I owe my thanks to my colleagues in the Computational Linguistic Lab for the help and encouragements they have given to me. Particular thanks must go to Guangsen Wang, Shilin Liu, Xuancong Wang, Thang Luong Minh and Lahiru Thilina Samarakoon for various discussions. There are many other individuals to acknowledge, but my thanks go to, in no particular order, Xiong Xiao, Lei Wang, Dau-Cheng Lyu, Xiaohai Tian and Bolan Su. I must also thank the technical service team for their excellent work in maintaining the computing facilities and the staff of the Deck canteen for their kindness especially when I was frustrated.

I cannot imagine a life in Singapore without the support from my wife, Xiaoxuan

Wang. She has shared my excitement, happiness as well as disappointment and sadness. Her support both emotionally and financially is the source of the energy for me to finish my study.

Finally, the biggest thanks go to my parents to whom I always owe everything! For many years, they have offered everything possible to support me, despite my lack of going back home since I entered college.

Table of Contents

Summary

Speech-based services are becoming widely adopted in real world applications. Developing Automatic Speech Recognition (ASR) systems that would be much more robust against variations and shifts in acoustic environments, external noise sources and communication channels is of crucial importance to the success of speech-based applications. Recently, Deep Neural Networks (DNNs) have been successfully integrated into ASR systems. Although they have much better generalization capabilities against variations than conventional systems, the gap between the performance on clean and noisy speech is still large. Additionally, many existing noise-robust feature extraction techniques and speech enhancement algorithms have been found to be ineffective for DNNs.

In this thesis, we address the DNN-based noisy speech recognition problem by learning robust representations. A Mean Variance Normalization technique is first developed to improve the robustness of the normalized feature representations. It integrates independently estimated noise statistics using the Vector Taylor Series model compensation. This technique is hence referred to as the VTS-MVN. It reduces noise variations in original feature representations and makes them more suitable for acoustic modeling. Due to the borrowed noise statistics, the gain is limited. DNNs' discriminative learning and complex nonlinearity further prevent the incorporation of the widely adopted noise model. We thus investigate DNNs' implicit environment modeling capability by employing a long temporal span of speech information. The change of the input dimension leads to a dramatical increase in the model size. A Deep Split Temporal Context (DSTC) system is then proposed. It models each sub-context separately and generates multiple representations that collectively yield better phonetic predictions.

The VTS-MVN and the DSTC implicitly improve the input representation robustness by learning reliable parameter estimations. To explicitly address the noise variations in input features, we revisit the missing feature theory and develop a DNN-based spectral masking system. Effective noise reductions and strong complementariness have been observed. By further addressing the training and testing mismatch problem, we can achieve the best performance on two benchmark tasks. DNN itself learns levels of representations to disentangle variations. The success of our spectral masking technique suggests its limitations in factoring out noise specific variations, which may still exist in those automatically learned hidden representations. An Ideal Hidden-activation Mask (IHM) is developed to identify and discard noise-prone latent feature detectors. With IHMs, the generated hidden representations are immune to input noise. This IHM has no noise-type dependency and is also more robust against estimation errors. A further analysis of the IHM leads to a noise code technique which simulates the IHM effects by attenuating the sigmoid activation functions with linearly estimated bias shifts. Moreover, the codes capturing environment statistics are estimated within the original DNN's learning framework towards the ultimate phonetic predictions.

Improved noise robustness has been obtained using the proposed techniques on two benchmark tasks, Aurora-2 and Aurora-4. The spectral masking approach successfully yields the best reported performance in the literature on both tasks at the time of writing and is one of the most promising noise-robust techniques for DNN-based ASR systems.

List of Acronyms

AFE	Advanced Front-End
AM	Acoustic Model
ASR	Automatic Speech Recognition
CD	Contrastive Divergence
CMS	Cepstral Mean Subtraction
CMVN	Cepstral Mean Variance Normalization
CNN	Convolutional Neural Network
CSN	Cepstral Sub-bank Normalization
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
DRDAE	Deep Recurrent Denoising AutoEncoder
DSTC	Deep Split Temporal Context
EBP	Error Back-Propagation
EM	Expectation Maximization
FBank	Filter-Bank
fMLLR	feature-based Maximum Likelihood Linear Regression
GMAPA	Generalized Maximum A Posterior spectral Amplitude estimator
GMM	Gaussian Mixture Model
GRBM	Gaussian-Bernoulli Restricted Boltzmann Machine
HEQ	Histogram EQualization
HMM	Hidden Markov Model
IBM	Ideal Binary Mask
IDCT	Inverse Discrete Cosine Transform
IHM	Ideal Hidden-activation Mask
IRM	Ideal Ratio Mask
KL	Kullback-Leibler

LIN	Linear Input Network
LM	Language Model
LVCSR	Large Vocabulary Continuous Speech Recognition
MAP	Maximum A Posterior
MAPA	Maximum A Posterior spectral Amplitude estimator
ME	Mask Estimator
MFCC	Mel Frequency Cepstral Coefficient
MFT	Missing Feature Theory
MLLR	Maximum Likelihood Linear Regression
MLP	Multi-Layer Perceptron
MLSA	Maximum Likelihood Spectral Amplitude estimator
MMSE	Minimum Mean Square Error spectral estimator
MSE	Mean Square Error
MVA	Mean subtraction Variance normalization with Autoregressive moving average filtering
MVN	Mean Variance Normalization
NAT	Noise Adaptive Training
NN	Neural Network
OOV	Out-Of-Vocabulary
PER	Phoneme Error Rate
PLP	Perceptual Linear Predictive
PMC	Parallel Model Combination
RASTA	Relative Spectra
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SNR	Signal-to-Noise Ratio
SPR	Single Pass Re-training
SS	Spectral Subtraction
STC	Split Temporal Context
T-F	Time-Frequency
VTS	Vector Taylor Series
WER	Word Error Rate
WSJ	Wall Street Journal

List of Tables

List of Figures

List of Symbols

E	the energy function
LC	the local SNR criterion for binarizing mask values
Z	the partition function
α	the momentum weight
β	the threshold parameter of the IRM
$\boldsymbol{A}$	the transformation matrix
$\boldsymbol{B}$	the mask basis matrix
$\boldsymbol{C}$	the Discrete Cosine Transform
$\boldsymbol{C}^{\dagger}$	the pseudo-inverse Discrete Cosine Transform
$\boldsymbol{F}$	the diagonal matrix involved in VTS
$\boldsymbol{I}$	the identity matrix
$\boldsymbol{J}$	the Jacobian matrix
$\boldsymbol{T}$	the transformation matrix
$\boldsymbol{W}$	the weight matrix of a neural network layer
$\boldsymbol{\Sigma}$	the covariance matrix of a Gaussian
$\boldsymbol{\mu}$	the mean vector of a Gaussian
$\boldsymbol{a}$	the visible bias vector of an RBM
$\boldsymbol{b}$	the bias vector
$\boldsymbol{c}$	the noise code vector
$\boldsymbol{d}$	the target posterior probability vector
$\boldsymbol{h}$	the hidden activation vector of a neural network layer
$\boldsymbol{o}$	the speech observation vector
$\boldsymbol{p}$	the posterior output vector of a neural network
ϵ	the floor constant
η	the learning rate
γ	the HMM state posterior
κ	the threshold parameter of the IHM

λ	the slope parameter of the IHM
$\mathcal{D}$	the KL-divergence distance measure
$\mathcal{E}$	the model cost function
$\mathcal{O}$	the speech observation sequence
$\mathcal{S}$	the HMM state sequence
$\mathcal{W}$	the word sequence
$\phi(x)$	the sigmoid function
$\psi(x)$	the softmax function
σ	the standard deviation of a visible unit
τ	the parameter update index
θ	the complete set of model parameters
ξ	the slope parameter of the IRM
ζ	the input offset of sigmoid functions
c	the Gaussian weight coefficient
m	the mask computed at each feature component
n	the time-domain speech and noise sample index
q	the similarity value between two hidden activations
r	the SNR computed at each T-F unit
s	the HMM state
t	the time frame index
u	the channel distortion signal
v	the visible input vector
x	the clean speech signal
y	the noisy speech signal
z	the additive noise signal

List of Publications

Journals

- **Bo Li**, Khe Chai Sim; *A Spectral Masking Approach to Deep Neural Network based Robust Speech Recognition*, [under review] Transactions on Audio, Speech, and Language Processing, IEEE/ACM, 2013.

Conferences

- **Bo Li**, Khe Chai Sim; *Modeling Long Temporal Contexts for Robust DNN-based Speech Recognition*, submitted to Interspeech, ISCA, 2014.
- **Bo Li**, Khe Chai Sim; *An Ideal Hidden-Activation Mask for Deep Neural Networks based Noise-Robust Speech Recognition*, in Proceedings of ICASSP, IEEE, 2014.
- **Bo Li**, Khe Chai Sim; *Improving Robustness of Deep Neural Networks via Spectral Masking for Automatic Speech Recognition*, in Proceedings of ASRU, IEEE, 2013.
- Zhiyan Duan, Haotian Fang, **Bo Li**, Khe Chai Sim, Ye Wang; *The NUS Sung and Spoken Lyrics Corpus: A Quantitative Comparison of Singing and Speech*, in Proceedings of APSIPA, IEEE, 2013.
- **Bo Li**, Yu Tsao, Khe Chai Sim; *An Investigation of Spectral Restoration Algorithms for Deep Neural Networks based Noise Robust Speech Recognition*, in Proceedings of Interspeech, ISCA, 2013.
- **Bo Li**, Khe Chai Sim; *Noise Adaptive Front-End Normalization based on Vector Taylor Series for Deep Neural Networks in Robust Speech Recognition*, in Proceedings of ICASSP, IEEE, 2013.
- **Bo Li**, Khe Chai Sim; *A Two-stage Speaker Adaptation Approach for Subspace Gaussian Mixture Model based Nonnative Speech Recognition*, in Proceedings of Interspeech, ISCA, 2012.
- Guangsen Wang, **Bo Li**, Shilin Liu, Xuancong Wang, Xiaoxuan Wang, Khe Chai Sim; *Improving Mandarin Predictive Text Input by Augmenting Pinyin Initials with Speech and Tonal Information*, in Proceedings of ICMI, ACM, 2012.

- Itamar Arel, Shay Berant, Tsvi Slonim, Ami Moyal, **Bo Li**, Khe Chai Sim; *Acoustic Spatial-temporal Modeling using Deep Machine Learning for Robust Phoneme Recognition*, in Proceedings of Signal Processing Conference, AFEKA, 2011.
- **Bo Li**, Khe Chai Sim; *Hidden Logistic Linear Regression for Support Vector Machine based Phone Verification*, in Proceedings of Interspeech, ISCA, 2010.
- **Bo Li**, Khe Chai Sim; *Comparison of Discriminative Input and Output Transformations for Speaker Adaptation in the Hybrid NN/HMM Systems*, in Proceedings of Interspeech, ISCA, 2010.

Chapter 1

Introduction

From prehistory to the multimedia digital age, speech communication has been the dominant mode of human social bonding and information exchange. With the advancement of technology, various machines and devices have been invented and adopted to ease humans' lives. The vision of communicating with these machines in speech has been a collective dream for many decades. Automatic Speech Recognition (ASR), the transcription of speech signals into word sequences, is the first step towards speech communication with machines. In contrast to the development of the first speech synthesizer in 1936 by AT&T, the first automatic speech recognizer, a simple digit recognizer, appeared in 1952 [1]. In 1969, John Pierce of Bell Labs said that ASR will not be a reality for several decades. However, the 1970s witnessed a significant theoretical breakthrough in speech recognition - Hidden Markov Models (HMMs) [2, 3]. Since then, the multidisciplinary field of ASR has proceeded from its infancy to its coming of age and into a quickly growing number of practical applications and commercial markets. HMMs were extensively investigated and became the most successful technique for acoustic modeling in speech recognition. The maximum likelihood based Expectation Maximization (EM) algorithm and the forward-backward (Baum-Welch) algorithm have been the principal means by which the HMMs are trained with data for more than 30 years. Over the past few years the striking progress in large-scale speech recognition has been attributed to the successful development and application of discriminative learning [4, 5, 6, 7]. Moreover, the success in learning Deep Neural Networks (DNNs) has further boosted the recognition performance towards humans' expectations since 2009 [8]. It has been reported that a Phoneme Error Rate (PER) of 17.7% has been achieved in 2013 [9] on the benchmark TIMIT phoneme recognition task, on which the expected human performance is 15% PER [10].

With the introduction and development of advanced statistical models and dramatically increased computing power, significant progress in ASRs has been achieved.

Continuous speech recognition has become the main research interest after simple connected and isolated word recognition was well dealt with. The size of the recognition vocabulary increased from 998 words in the Resource Management task (1988-1992) to 20000 in the Wall Street Journal (WSJ) task (1993-1995). A recognition system with a vocabulary size of the order of the WSJ task is often referred to as a Large Vocabulary Continuous Speech Recognition (LVCSR) system. With the rise of deep neural networks for speech recognition, many industry level systems have been deployed such as Google's voice search and YouTube's video transcription, Apple's Siri etc. These systems usually have even bigger dictionaries [11]. Besides the vocabulary size, the difficulty of evaluation tasks has also been increased in other aspects to approximate a more realistic and practical recognition problem. For example, the acoustic environment of the evaluation data has changed from a quite laboratory condition to realistic noisy ones. More natural and spontaneous speech with severe signal degradation, such as conversational telephone speech, has also been introduced to the evaluation since 1998. Up to now, the state-of-the art ASR systems are built for the spontaneous natural continuous large vocabulary speech.

As speech recognition tasks become more and more difficult, many challenging problems of acoustic modeling emerge. One of the main challenges is the diverse acoustic conditions of the recorded speech data. Speech might be recorded in different acoustic environments or with different channel distortions. Though these acoustic conditions do not reflect the words people speak, the additional non-speech variations introduced could confuse the statistical ASR systems and usually cause severe performance degradation. This happens because of the mismatches between the data used for acoustic model training and the testing speech that we want to recognize. It is usually unavoidable for practical applications, especially under noisy conditions, as noise is inherently unstable. It is also impossible to have training data that could cover all possible noise environments. Although the recently developed DNNs have been shown to have much better generalization capabilities than traditional Gaussian Mixture Models (GMMs), their degradation under adverse environments is still severe and below humans' expectations. With the rapid adoption of DNNs in industrial level applications, their noise robustness needs to be more and more urgently addressed. This work began by investigating various noise robustness techniques successfully developed for the GMM-HMM systems. However, due to the inherently different model formulations between a discriminative DNN and a generative GMM, most of those techniques are either ineffective or inapplicable. Techniques specific to DNNs are in high demand. A noise-robust representation learning framework is hence proposed in this work and several techniques are successfully developed. They include the Vector Taylor Series - Mean Variance Normalization (VTS-MVN), the Deep Split Temporal Context (DSTC), the spectral masking approach for improving the input feature noise robustness, the Ideal Hidden-activation Mask (IHM) and the noise code technique to learn robust latent representations. Greater details will be presented in the remaining chapters. In this chapter, we will review the basic ASR system and discuss the model and the problem to be studied.

1.1 Automatic Speech Recognition

The task of a speech recognition system is to generate a word sequence from a given speech signal, which is commonly represented as a waveform. Mathematically, the ASR is formulated as an optimization problem:

$$\mathcal{W}^* = \arg\max_{\mathcal{W}} p(\mathcal{W}|\mathcal{O}) = \arg\max_{\mathcal{W}} p(\mathcal{O}|\mathcal{W})\, p(\mathcal{W}) \tag{1.1}$$

where $\mathcal{W}^*$ represents the target word sequence we would like to obtain, $\mathcal{W}$ represents all the possible word sequences and $\mathcal{O}$ represents the speech signal we have observed. The ASR problem is to find the most probable $\mathcal{W}$ given the speech observation $\mathcal{O}$. Using Bayes rule, it is further decomposed into two sub-components: the probability of the speech observation given a word sequence, $p(\mathcal{O}|\mathcal{W})$, and the probability of the corresponding word sequence, $p(\mathcal{W})$.

Based on the above mathematical foundation, a conventional engineering approach to the ASR problem includes following components: a feature extraction module, an acoustic model, a lexicon and a language model. The general processing pipeline is illustrated in Figure 1.1. The feature extraction module pre-processes and transforms the speech signal into a new set of feature representations that have discarded unnecessary variations and maintained only the linguistic-related information. This representation is then forwarded to the acoustic model which generates a likelihood representation of the input. The likelihood is commonly in the granularity of the phoneme or sub-phoneme units. The likelihood representation is further combined with the language model through the mapping defined by the lexicon to form a probabilistic search space. By searching for the word sequence that has the highest probability, we can finally obtain the output word representation of the original input speech signal.

Feature Extraction

Analogue speech signals are usually sampled by hardware devices into digital waveform signals which have rather high dimensions. For example, for the telephone speech with an 8kHz sampling frequency and 8-bit sample size, there will be full 8000 8-bit values at each second. Moreover the large variations in the time-domain waveform signals

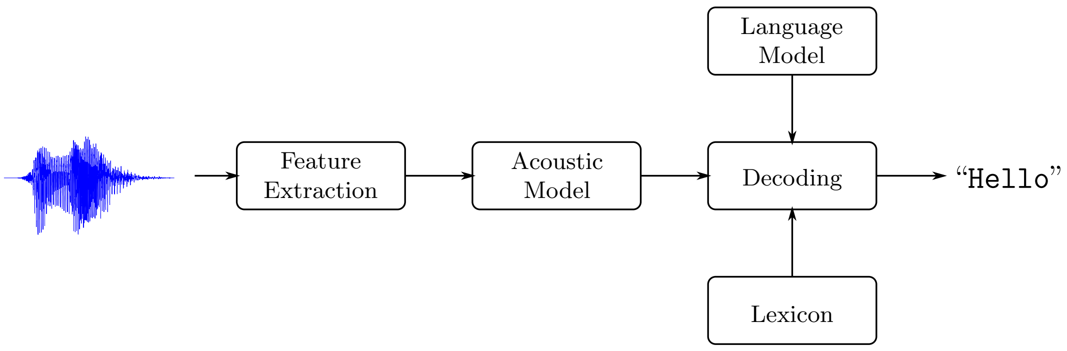


Figure 1.1: The generic automatic speech recognition system architecture.

also prohibit their direct use in speech recognition systems. A compact frequency-domain representation is preferable. The most widely adopted feature representation is the cepstral domain Mel Frequency Cepstral Coefficient (MFCC). The computation process for MFCCs is illustrated in Figure 1.2.

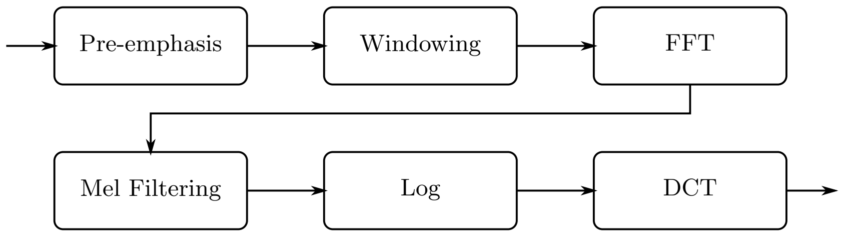


Figure 1.2: Major computational components for the MFCC feature extraction.

A pre-emphasis filter is firstly applied to the original speech signals using the first order difference. A windowing function is then carried out to slice the signal into overlapping segments with fixed length and hop size. Usually, we use 25ms for the window size and 10ms for the hop size. Each segment is usually referred to as a speech frame. In our case, there will be 100 frames per second and 15ms overlapping between successive frames for smooth transitions. The Hamming window function is adopted to taper the samples inside each window so that discontinuities at the window edges are attenuated. The short time Fast Fourier Transform (FFT) is further employed to convert the time domain signals into frequency representations for improved compactness, which can be conveniently presented as a spectrogram for visual inspection. Motivated by the process of human speech perception, this frequency representation is first mapped onto the Mel frequency scale and then recombined inside each equidistant channel with a triangular shaped frequency window. Consecutive channels are half-overlapped also to maintain smooth changes from one channel to another. Motived by the fact that we do not hear loudness on a linear scale, the logarithm compression function is adopted [12, 13]. Flooring thresholds are also commonly employed to adjust the feature value ranges. This representation is commonly referred to as the log-Mel domain Filter-Bank (FBank) representation. Through this processing, the feature dimension for each frame has been largely reduced to only $20 \sim 30$; but this is still a little high for the traditional GMM-HMM systems. A Discrete Cosine Transform (DCT) is further adopted to both de-correlate the FBank feature dimensions and further reduce the dimensionality. The resulting feature is usually referred to as the MFCC feature, which commonly has a dimension of 13. As a time series signal, sequential information is crucial to ASRs. Dynamic features [14] that capture the temporal information in the speech are often appended. The first order and second order dynamic features (also known as the delta and accelerator coefficients) may be computed. They have been shown to be particularly useful in addressing the conditional independence assumption of HMMs. Namely, the observation probability of a particular feature frame is independent of others given the HMM state.

Lexicon

A lexicon defines the mapping between a word and its corresponding linguistic unit representation. For example, the entry for the word `Hello` and its phoneme representation is given by

$$\boxed{\texttt{Hello} \rightarrow \textbf{HH AH L OW}}$$

Figure 1.3: Phoneme representation of the word “Hello”.

For word-based systems, the lexicon is trivially a self-mapping. While for phonetic ones, the CMUDict [1] is one of the most commonly used lexicons in speech recognitions.

Furthermore, the lexicon also determines the vocabulary for an ASR system, which is the set of possible words the recognizer could output. Words that do not appear in the lexicon are called Out-Of-Vocabulary (OOV) words. The OOV word rate is measured against a corpus of texts that represent the domain within which the recognizer will operate. Too high an OOV rate would render the ASR system useless. The vocabulary size has a direct impact on the system performance. Increasing the vocabulary size reduces the OOV rate; at the same time, it also enlarges the search space and decoding complexity.

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Acoustic Model

An Acoustic Model (AM) captures feature variations for different linguistic units in ASR systems. The choice of speech units depends on specific applications. There is usually a trade-off between the number of speech units and the size of the final acoustic model. For small or medium vocabulary isolated word recognition, word-based models may be used; while for a large vocabulary system, a phoneme or sub-phoneme model is more preferable. Besides, the amount of training data available also affects the choice of speech units. With sufficient training data, context dependent models are always better in capturing the co-articulation effects in speech.

To model the time structure of speech signals, HMMs are commonly adopted in the ASR community. A linear three-hidden-state HMM (Figure 1.4) is usually used for each linguistic unit. Those hidden states correspond to the starting, middle and ending parts of a phonetic unit. Two dummy states, which do not consume any observations, also exist to ease the concatenation of different phonetic HMMs together to form higher level ones. For example, the concatenation of the sequence of HMMs corresponding to the phonemes of a word would yield the HMM for that word. Similarly, a sentence HMM could be constructed from the word HMMs. For each HMM state, a GMM is normally used to represent the distribution of all the speech features corresponding to that specific state. The acoustic model probability $p(\mathcal{O}|\mathcal{W})$ for the GMM-HMM could be computed by:

$$p(\mathcal{O}|\mathcal{W}) = \prod_i p(\mathcal{O}_i|\mathcal{W}_i)p(\mathcal{W}_i|\mathcal{W}_{i-1}), \tag{1.2}$$

where $\mathcal{W}_i$ is the ith word in the sequence $\mathcal{W}$ and $\mathcal{O}_i$ is the feature sequence corresponding to the the word $\mathcal{W}_i$. $p(\mathcal{W}_i|\mathcal{W}_{i-1})$ is the transition probability from word $\mathcal{W}_{i-1}$ to $\mathcal{W}_i$. It is commonly provided by a separate model, *i.e.* the language model, which is independent of the acoustic model and will be discussed in the following section. With a lexicon, we could further have

$$p(\mathcal{O}_i|\mathcal{W}_i) = \prod_j p(\mathcal{O}_{i,j}|\mathcal{W}_{i,j})p(\mathcal{W}_{i,j}|\mathcal{W}_{i,j-1}), \tag{1.3}$$

where $\mathcal{W}_{i,j}$ is the jth linguistic unit of the word $\mathcal{W}_i$ and $\mathcal{O}_{i,j}$ is the feature sequence corresponding to the linguistic unit $\mathcal{W}_{i,j}$. The transition probability between linguistic units, $p(\mathcal{W}_{i,j}|\mathcal{W}_{i,j-1})$ is usually set to 1 for simplicity. Furthermore, down to the frame level, we have

$$p(\mathcal{O}_{i,j}|\mathcal{W}_{i,j}) = \prod_t p(\boldsymbol{o}_t|s_t)\,p(s_t|s_{t-1}), \quad \text{with} \quad p(s_1|s_0) = 1.0 \tag{1.4}$$

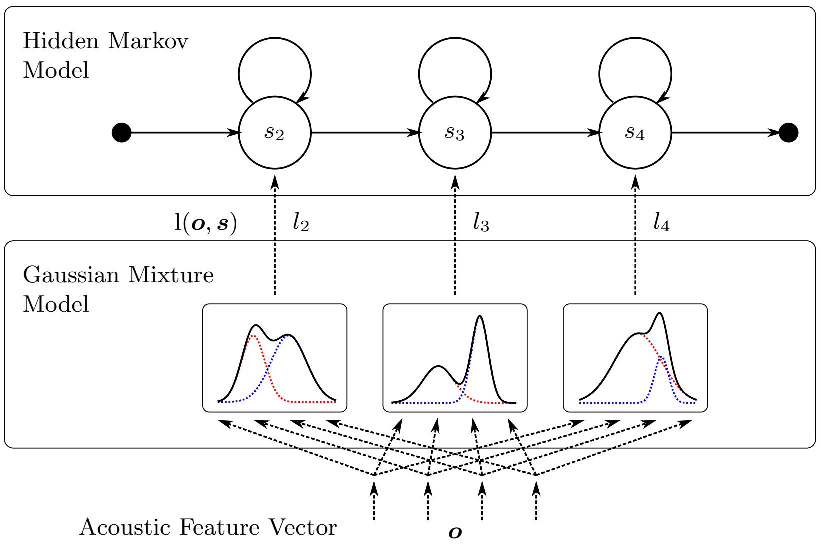


Figure 1.4: The GMM-HMM speech recognition system architecture.

where s_t is the HMM state that the tth feature frame of $\mathcal{O}_{i,j}$ belongs to and $p(s_t|s_{t-1})$ is the HMM state transition probability from state s_{t-1} to state s_t. Combining equations (1.2), (1.3) and (1.4), for a length T feature sequence $\mathcal{O}$, we could simply compute the likelihood using the following formula:

$$p(\mathcal{O}|\mathcal{W}) = p(\mathcal{O}|\mathcal{S}) \tag{1.5}$$

$$= \prod_{t=1}^{T} p(\boldsymbol{o}_t|s_t)\, p(s_t|s_{t-1}), \tag{1.6}$$

where $\mathcal{S}$ is the state sequence derived from the word sequence $\mathcal{W}$ using the lexicon and the HMM. The transition probability $p(s_t|s_{t-1})$ is a mix of the word transition probability provided by the language model, the pronunciation probability provided by the lexicon and the HMM state transition probability provided by the acoustic model. For a GMM-HMM AM, each state is modeled by a mixture of Gaussians and the emission probability could be computed by

$$p(\boldsymbol{o}_t|s_t) = \sum_m c_{stm} \mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{stm}, \boldsymbol{\Sigma}_{stm}), \tag{1.7}$$

where c_{stm}, $\boldsymbol{\mu}_{stm}$ and $\boldsymbol{\Sigma}_{stm}$ are the weight coefficient, the mean vector and the covariance matrix of the mth Gaussian in the GMM corresponding to the state s_t.

Language Model

A general form of stochastic Language Models (LMs) may be used to compute the probability that a sentence comes from a specific language as follows:

$$\begin{aligned} p(\mathcal{W}) &= p(\{\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_N\}) \\ &= p(\mathcal{W}_1) \prod_{i=2}^{N} p(\mathcal{W}_i | \{\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_{i-1}\}) \end{aligned} \tag{1.8}$$

where $\mathcal{W}_i$ represents the ith word in the length N word sequence $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_N\}$. The most widely used language model is the n-gram LM. Despite its simplicity, the n-gram language model has been proven to be remarkably powerful and resilient. It approximates the probability of a word to be dependent only on the $n-1$ most recent history. Mathematically,

$$p(\mathcal{W}_i | \{\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_{i-1}\}) \approx p(\mathcal{W}_i | \{\mathcal{W}_{i-n+1}, \mathcal{W}_{i-n+2}, \cdots, \mathcal{W}_{i-1}\}). \tag{1.9}$$

Typical forms of n-gram are the bigram ($n = 2$), trigram ($n = 3$) and 4-gram ($n = 4$) LMs.

1.2 Deep Neural Networks for ASR

The HMM has always been the gold standard in speech recognition systems for dealing with the temporal variabilities of speech signals. The GMM is popular in modeling the acoustic variations for each state of the HMM. GMM-HMM ASR systems are effective under many circumstances, but they do suffer from some major limitations. For example, it is difficult to model the temporal dependencies among the adjacent feature frames in GMMs and most commonly the feature dimensions are assumed to be independent so that a diagonal covariance for the Gaussian is sufficient. Besides, to model non-Gaussian distributions, such as a plane in a high dimensional space, a large bunch of Gaussians are required for a good approximation. There have always been attempts to overcome these limitations by adopting more advanced statistical models. Between the end of the 1980s and the beginning of 1990s [15], some researchers proposed to replace GMMs with Neural Networks (NNs) [15, 16, 17] for generating state posteriors rather than likelihoods.

The use of NNs have several potential advantages over GMMs. Firstly, NNs are capable of directly modeling a long span of acoustic feature vectors. The temporal dependencies between feature frames together with the correlations among different feature dimensions could be well captured. Secondly, they are discriminative classifiers, which model the classification boundaries rather than the data distributions. This could avoid the improper data distribution assumptions brought by generative models such as GMMs. NNs also allow an easy way of combining diverse features and use far more samples to constrain each parameter, as usually one single model is used to generate all the linguistic class posteriors.

Despite the advantages of NNs' over GMMs, they did not become the main stream technique for ASR systems. One major problem is the difficulty of learning a sufficiently large model that is capable of robustly predicting the HMM state posterior vectors with hundreds or thousands of dimensions. Another aspect lies in the hardware computation capability that is also limiting the learning of complex NNs. Before 2006, the hybrid NN-HMM has only been shown to outperform the conventional GMM-HMM systems for context independent phoneme recognitions and cannot beat state-of-the-art GMM-HMM LVCSR systems with various optimization techniques applied. The breakthrough of training NNs with more than two hidden layers, namely Deep Neural Networks (DNNs), in the machine learning community has triggered revolutionary changes in various research communities and also generated great interest from industries. They have opened up a new paradigm, deep learning, in machine learning for artificial intelligence. This breakthrough has been one of the three technical advances that have appeared on the front page of the New York Times in recent years [18]. The other two happened when a computer beat the world's number 1 chess player [19] and when Watson beat the world's best Jeopardy players [20]. Speech recognition is one of the early adopters of deep learning techniques and the first success occurred in 2009 [8]. The hybrid NN-HMM system using DNNs for acoustic variation modeling, which will be referred to as the hybrid DNN-HMM system in the remaining part of the thesis, has been shown to largely outperform the sophisticatedly optimized GMM-HMM systems in many applications. [21] showed that the DNN-based AMs dramatically outperform GMMs on a small-scale phoneme recognition task. It was later extended to a large vocabulary voice search task in [22] and similar improvements were reported. Research groups such as Microsoft [22, 23, 24], Google [11, 23], IBM T. J. Watson [23, 25] etc. have also observed impressive gains from using DNN AMs on large vocabulary continuous speech recognition tasks.

These advances in speech recognition technology speed up the adoption of ASR systems in real world applications such as Apple's Siri, Google and Microsoft's voice search etc. As speech recognition technology is transferred from the laboratory to the marketplace, robustness in recognition is becoming increasingly important. Robustness refers to the need of maintaining good recognition accuracies even when the quality of the input speech is degraded, or when the acoustical, articulatory, or phonetic characteristics of speech in the training and testing environments differ. Obstacles to robust recognition include acoustical degradation produced by additive noise, the effects of linear filtering, nonlinear transduction or transmission, as well as impulsive interfering sources, and changes in articulation produced by the presence of high-intensity noise sources. Creating and developing systems that would be much more robust against the variabilities and shifts in acoustic environments, reverberations, external noise sources, communication channels (e.g., far-field microphones, cellular phones), speaker characteristics (e.g., speaker style, nonnative accents, speaking rate), and language characteristics (e.g., styles, dialects, vocabulary, topic domain) has always been the dream of ASR researchers. Despite the impressive improvements of DNNs over GMMs, large degradation still exists when there are mismatches between training and testing speech. Hence, the training samples are often expected to contain large variations, with the hope of covering all possible noise conditions. However, in practice, it is impossible to obtain such a large training corpus due to the inherent variability of noise.

1.3 Major Contributions

To tackle the noise robustness problem of DNNs, state-of-the-art techniques proposed for the conventional GMM-HMM systems are firstly investigated. However, many of those techniques have been found to be ineffective for DNNs.

In this thesis, a DNN-specific noise-robust representation learning framework is proposed. It addresses the robustness problem by generating different levels of noise-invariant representations. Two general types of representations are studied, namely the input feature representations and the DNN-generated hidden representations.

To improve the noise-robustness of the input feature representations, we have developed a Vector Taylor Series - Mean Variance Normalization (VTS-MVN) technique to improve the reliability of the normalized input representation, a Deep Split Temporal Context (DSTC) algorithm to model the long-temporal context-expanded input representation and a DNN-based spectral masking approach to reduce the noise variations in the input spectral feature representation.

Following the idea of masking away noise variations, we further propose an Ideal Hidden-activation Mask (IHM) for the hidden representations. Different from the spectral masking, the IHM operates on the distributed latent representations automatically learned by the DNN and identifies latent feature detectors that are invariant to variations caused by noise. A further analysis of the IHM leads to a noise code technique that simulates the IHM effects by attenuating the sigmoid activation functions with linearly estimated bias shifts. In this way, the code vectors capturing environment statistics can be estimated within the original DNN AM towards the ultimate phonetic predictions. No extra DNN for mask estimations is required.

All the proposed techniques are evaluated on two benchmark noisy speech recognition tasks, Aurora-2 and Aurora-4. Improved noise-robustness has been obtained and the spectral masking approach has been shown to yield the best reported results on both tasks at the time of writing.

Details about these techniques will be presented in the following chapters and the structure of this thesis is firstly explained in the following section.

1.4 Organization of Thesis

The remaining part of this thesis is organized as follows:

Chapter 2 first discusses how the noise affects the speech signal and then reviews state-of-the-art noise-robust speech recognition techniques successfully developed for the GMM-HMM systems. They are grouped into three categories, namely the feature-based enhancement, the model-based compensation and the uncertainty-based schemes. This review will server as the foundation for our following exploration of noise-robust techniques for the DNN-HMM system.

Chapter 3 starts with the detailed formulation of the DNN acoustic model. A further justification of the noise-robust problem of DNNs is conducted. It narrows down to two major noise variations that we will focus on in this study, namely the additive noise and the channel distortion. Following that, ineffectiveness of many existing GMM-based noise-robust techniques is reported and the importance of developing DNN specific techniques is discussed. Motivated from the development of deep learning algorithms, a representation learning framework is proposed to address the DNN AM's noise robustness. A preliminary study of the noise effects on those different levels of representations is conducted which confirms the feasibility of the proposed approach.

Chapter 4 presents three techniques we have successfully developed to improve the noise robustness of the input representations. They are the Vector Taylor Series - Mean Variance Normalization (VTS-MVN) for the normalized representation, the Deep Split Temporal Context (DSTC) for the context-expanded representation and the spectral masking for the input spectral representation.

Chapter 5 describes two techniques that address the noise variations in DNNs' automatically learned latent representations. The first one, the Ideal Hidden-activation Mask (IHM), extends the spectral masking approach into DNNs' hidden-activation domains. Further understanding of the IHM leads to the second technique, the noise code, which integrates the masking effect into the DNN acoustic model's hidden activation functions directly.

Chapter 6 justifies the various noise-robust representation learning techniques introduced in this thesis on two benchmark noisy speech recognition tasks, Aurora-2 and Aurora-4. Clear performance improvements have been obtained. A performance comparison between our proposed techniques and those reported in the literature is presented at the end of this chapter.

Chapter 7 concludes the thesis by emphasizing the major contributions and discussing some potential future research directions.

Chapter 2

Noise-Robust Speech Recognition

Understanding the distortions noise brings to speech, the difficulties it presents to current models and the solutions successfully developed to conventional GMM-based systems are of great importance to the success of finding new noise-robust algorithms for the DNN-based ASRs. In this chapter, a generic environmental model is firstly described and state-of-the-art noise robust techniques developed for conventional GMM-HMM systems are thoroughly reviewed. These techniques are grouped into three broad categories, namely the feature-based enhancement, the model-based compensation and the uncertainty-based schemes. For the feature-based approaches, different noise-robust feature parameterizations and speech enhancement algorithms are discussed. Commonly adopted model-based compensations are then reviewed, which include the Single Pass Re-training (SPR), the Maximum Likelihood Linear Regression (MLLR), the Parallel Model Combination (PMC) and the Vector Taylor Series model compensation (VTS). Following that, uncertainty-based techniques that treat the unknown environment as uncertainties in speech signals are presented. As one of the uncertainty-based schemes, the Missing Feature Theory (MFT) is revisited , which is motivated from the human speech perception process. At the end, a brief discussion on the estimation of environment model parameters concludes this chapter.

2.1 Model of the Environment

Noise is inherently unpredictable which makes it impossible to name and list all the noise types that a speech recognizer could encounter. Fortunately, noise may be approximately characterized by a model of the acoustic environment. The production of the underlying speech signal is influenced by stress, emotion and noise. What is spoken can then be colored by additive background noise, channel distortions either due to the microphone or network, and finally possible noise at the near end of the speech

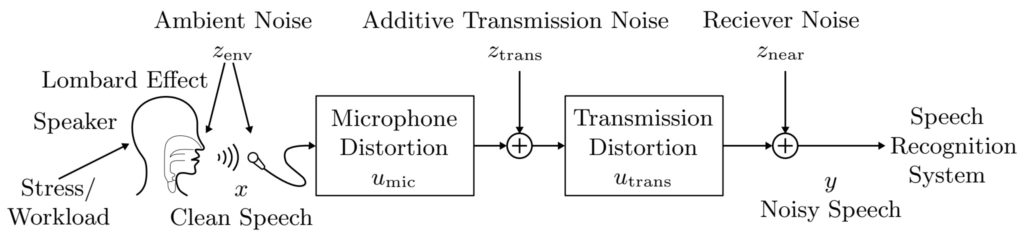


Figure 2.1: Noise sources and distortions that can affect speech.

recognition system. This is summarized in a model from [26] shown in Figure 2.1 and equation (2.1).

$$y(n) = \left[\left(\left\{\left[x(n)\Big|^{\text{Workload}}_{\begin{smallmatrix}\text{Stress}\\ \text{Noise}\end{smallmatrix}}\right]_{z_{\text{env}}(n)} + z_{\text{env}}(n)\right\} * u_{\text{mic}}(n) + z_{\text{trans}}(n)\right) * u_{\text{trans}}(n)\right] + z_{\text{near}}(n) \tag{2.1}$$

This model accounts for changes in speech production due to the task workload, stress or surrounding noise by conditioning $x(n)$ on these factors. The last factor, noise, is the cause of the Lombard effect: as the level of noise increases, speakers tend to hyper-articulate and emphasize vowels while consonants become distorted [27]. It is well known that recognition performance degrades significantly for stressed speech, such as Lombard, angry or loud speech compared to neutrally produced speech [28, 29], which recognizers are trained on. Attempts to address these effects have been beneficial [30, 31]; however, in this work, their effects on speech production will not be directly dealt with.

In the model given in equation (2.1), a major source of corrupted noise is the additive ambient environmental noise, $z_{\text{env}}(n)$, present when the user is speaking. The combined speech and noise signal is then captured and filtered by the microphone impulse response, $u_{\text{mic}}(n)$, which can be another large source of distortion. Transmission may also add noise, represented by $z_{\text{trans}}(n)$ and $u_{\text{trans}}(n)$, although it is expected to be small. The noise at the receiver side $z_{\text{near}}(n)$ is also expected to be minimal. Hence, equation (2.1) may be simplified by combining the various additive and convolution noise sources into a single additive noise variable, $z(n)$, and a linear channel convolution noise variable, $u(n)$. Doing so gives this standard, commonly adopted model [32, 33, 34, 35, 36] of the noisy acoustic environment in time domain as shown in Figure 2.2. The noisy signal is now given by

$$y(n) = x(n) * u(n) + z(n) \tag{2.2}$$

where $y(n)$ is the noise-corrupted speech and $x(n)$ is the clean speech. Note that $z(n)$ is a microphone and channel filtered version of the actual ambient noise $z_{\text{env}}(n)$ present

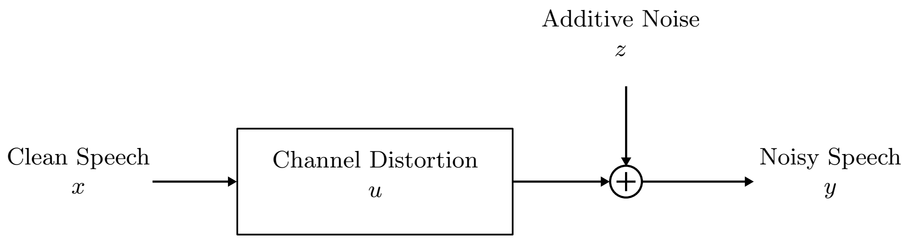


Figure 2.2: Simplified noisy acoustic environment model.

with the speaker and therefore dependent on $u(n)$; still for simplicity, they are assumed to be independent.

With this noise environment model, after applying the front-end processing steps discussed in Section 1.1, we could determine the interaction between speech and noise both in the FBank domain:

$$\boldsymbol{y}^{(\text{FBank})} = \boldsymbol{x}^{(\text{FBank})} + \boldsymbol{u}^{(\text{FBank})} + \log\left(1 + \exp(\boldsymbol{z}^{(\text{FBank})} - \boldsymbol{x}^{(\text{FBank})} - \boldsymbol{u}^{(\text{FBank})})\right) \tag{2.3}$$

and in the MFCC domain:

$$\boldsymbol{y}^{(\texttt{MFCC})} = \boldsymbol{x}^{(\texttt{MFCC})} + \boldsymbol{u}^{(\texttt{MFCC})} + \boldsymbol{C}\,\log\Big(1 + \exp\left(\boldsymbol{C}^{\dagger}(\boldsymbol{z}^{(\texttt{MFCC})} - \boldsymbol{x}^{(\texttt{MFCC})} - \boldsymbol{u}^{(\texttt{MFCC})})\right)\Big) \tag{2.4}$$

where $\boldsymbol{x}_t^{(\text{FBank})}$, $\boldsymbol{y}^{(\text{FBank})}$, $\boldsymbol{u}^{(\text{FBank})}$, $\boldsymbol{z}^{(\text{FBank})}$ are the FBank representations and $\boldsymbol{x}^{(\texttt{MFCC})}$, $\boldsymbol{y}^{(\texttt{MFCC})}$, $\boldsymbol{u}^{(\texttt{MFCC})}$, $\boldsymbol{z}^{(\texttt{MFCC})}$ are the MFCC representations of the clean speech, noisy speech, channel and additive noise. log and exp functions operate in an element-wise manner that yield a vector of the same dimensionality as the input vector. $\boldsymbol{C}$ and $\boldsymbol{C}^{\dagger}$ are the DCT transform and its pesudo-inverse. Equations (2.3) and (2.4) clearly show that the corrupted speech is a complicated non-linear function of the channel, noise and clean speech.

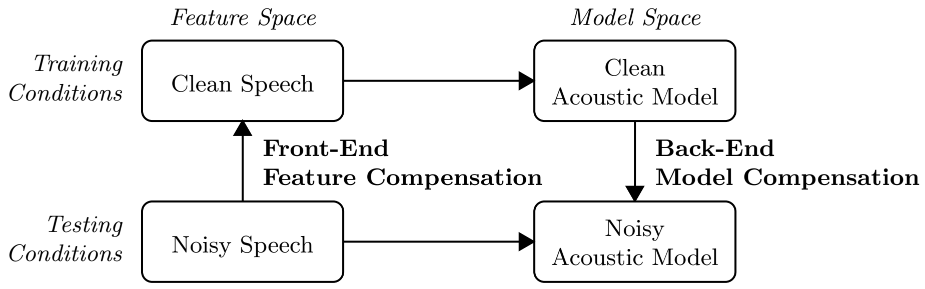


Figure 2.3: Methods of reducing the acoustic mismatches.

To robustly recognize noise corrupted speech, ideally, a noise invariant speech parameterization should be found. This has not been proven to be possible for widely varying levels of noise. Hence in the literature, most techniques focus on reducing the mismatch between the training and usage conditions. They can be grouped into two distinct approaches as shown in Figure 2.3. Front-end noise compensation approaches modify noise corrupted observations to provide an estimate of the feature vector that more closely resembles the clean speech found in training. These estimates can then be decoded using the clean-trained acoustic models. Back-end acoustic model compensation updates the clean-trained acoustic models to a corrupted model set that better matches the noise-corrupted observations in the target environment. Many of the adaptation techniques may also be used for noise robustness.

2.2 Feature-based Compensation

As shown in Figure 2.3, one approach to improve ASR robustness is to remove the training and testing mismatch in the feature space. That is to de-noise the incoming observations to obtain the matched pseudo-clean speech observations. This de-noising results in features that better match the original clean speech that the acoustic model is trained on. For enhancement, it is often the case that the corrupted speech is mapped deterministically to a clean speech estimate, given some estimate of the noise. Figure 2.4 outlines the standard feature compensation process. There are various methods to compute the pseudo-clean speech features, which can be broadly classified into those that enhance the spectral domain, and those that compensate the cepstral parameters.

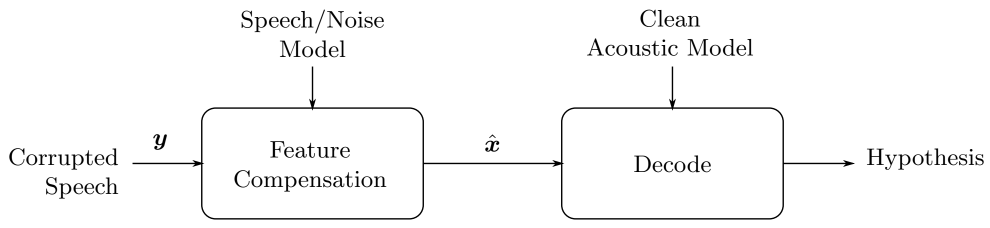


Figure 2.4: The standard feature compensation process.

2.2.1 Noise-Robust Features

A straightforward solution to the problem of environmental noise is to build a system that is immune to it. The shift from using log-spectral features, *i.e.* FBanks, to cepstral features such as MFCCs [12] and Perceptual Linear Predictives (PLPs) [37], could be considered as moving towards a more robust parameterization. However, those parameters on their own are not immune to noise. A relative spectral (RASTA) processing technique has hence been developed for PLP features, namely the RASTA-PLP, to make them less sensitive to slowly changing or steady-state noise factors in speech [38]. In the framework of noise-robust speech recognition, an inherently robust front-end would remove the dependency of the observations from the noise and allow decoding with the noisy observations directly. Many front-end parameterizations have been proposed for their robustness against noise, including Cepstral Mean Subtraction (CMS) [39], Cepstral Mean Variance Normalization (CMVN) [40], Cepstral Sub-band Normalization (CSN) [41], Mean subtraction Variance normalization with Autoregressive moving average filter (MVA) [42], Histogram EQualization (HEQ) [43] and Advanced Front-End (AFE) [44]. The CMS is the simplest parameterization and the CMVN has an additional variance normalization. These two methods are more generic and the CMVN is de facto a common practice for the neural network's inputs to guarantee a zero-mean and unit-variance input feature distribution. The CSN further assumes that the high frequency band of the decomposed speech are mainly noise and thus could be simply zeroed out. Only the lower frequency band is normalized using CMVN. Furthermore, instead of normalizing the whole utterance based on the complete statistics, MVA uses an autoregressive moving average filter to gradually carry out the CMVN normalization. This would be more adaptive to utterances with fast noise changes. While the HEQ has a slightly different assumption that the mismatch could be reduced by simply matching the overall training and testing distributions, it may thus limit its effectiveness and the histogram-based distribution modeling also limits its performance. The AFE tries to explicitly estimate the noise and remove it from the noisy speech. Although these approaches could yield gains for noisy speech, some can degrade the performance in clean environments. Moreover, this additional processing further complicates the speech-noise interaction function.

2.2.2 Feature Enhancement

An early method of addressing additive noise is Spectral Subtraction (SS) [45]. The noise magnitude spectrum is estimated from frames that are classified as not having speech. This estimate of the noise can then be subtracted from the corrupted signal to yield an enhanced feature vector assuming the noise is additive and varies slowly in time. A general form for SS is

$$|\hat{x}_{f,t}|^{\alpha} = \max(|y_{f,t}|^{\alpha} - \mathcal{E}\{|z_{f,t}|^{\alpha}\}, \epsilon) \tag{2.5}$$

where $\hat{x}_{f,t}$ and $y_{f,t}$ are the spectrum value of the estimated clean speech and input noisy speech. $\mathcal{E}\{|z_{f,t}|\}$ is the expected value of the noise spectrum. Power SS results from $\alpha = 2$ and magnitude SS at $\alpha = 1$ [46]. They remove the additive noise in the power spectral domain or magnitude spectral domain respectively. This technique is fairly effective although the negative spectra problem it results must be addressed with the floor constant ϵ, and a voice activity detector is needed to provide a background noise estimate. Magnitude SS assumes the speech and noise are in phase, which is generally not true. By contrast, power SS assumes the noise and speech are uncorrelated, which should give better results.

The enhancement can also be improved by having a more detailed model of the speech rather than a simple global one. This motivates state-based speech enhancement where improved results can be attained by aligning a simple front-end HMM to the corrupted speech and using the state statistics to more informatively enhance the speech using Wiener filters. The corrupted speech models of the front-end HMM can be recursively estimated from a combination of the clean and noise models using an EM algorithm as suggested in [47]. Since the corrupted state sequence maps to the clean sequence in a one-to-one fashion, the clean speech state sequence can be obtained. This allows for better estimates of the clean and noisy speech statistics, by using the state rather than global statistics, for use in the enhancement process. Enhancement with auto-regressive HMMs of speech is studied in [48, 49, 50].

As discussed in [51], speech enhancement can be viewed as minimizing the average distortion between an estimator of the clean speech vector $\check{\boldsymbol{x}}_t$ and the hidden, true clean speech vector $\boldsymbol{x}_t$. If the distortion measure is the Euclidean norm $\|\cdot\|$ then this leads to the following MMSE estimate of clean speech

$$\hat{\boldsymbol{x}}_t = \arg\min_{\check{\boldsymbol{x}}_t} \mathcal{E}\{\|\boldsymbol{x}_t - \check{\boldsymbol{x}}_t\|^2 | \mathcal{O}\}. \tag{2.6}$$

2.3 Model-based Compensation

Rather than updating the features, the acoustic model parameters can be compensated to match the noisy test conditions. This is the other main noise robustness approach illustrated in Figure 2.3. An obvious example of updating the models is to re-train them with data from the new environment. This may be referred to as *matched* or multi-pass training. While matched training usually yields the best results in a variety of papers surveyed [52, 53, 54], it is not very practical since large amounts of noisy training data are required and the noise condition may vary. Artificial methods of corrupting the training data have been explored which also yield good results. Samples of noise, such as those from the NOISEX-92 database [55], can be added to the clean training data to generate noise-corrupted training data. This provides good results for levels of noise down to 6∼10dB. However, matched training cannot easily address changing noise conditions. Adding a variety of noise samples to clean training data is known as multi-style or multi-condition training [56, 57], which has shown to improve noise robustness [58].

Due to the unpredictable nature of noise, it is not possible to account for all noise conditions that may be encountered by including them in the training data. Thus other acoustic model compensation methods that update the model parameters may be categorized as either: adaptive, where sufficient corrupted speech data are available to update the acoustic models to match the noisy speech observations; or predictive, where a noise model is combined with the clean speech models to provide a corrupted speech acoustic model using some model of the acoustic environment. If sufficient data from the target environment is available a Single Pass Re-training (SPR) would give a matched model, otherwise Maximum A Posterior (MAP) [59] and Maximum Likelihood Linear Regression (MLLR) transforms can be considered. Besides these adaptive forms, predictive forms such as Parallel Model Combination (PMC) and Vector Taylor Series (VTS) are also very effective. The noisy speech acoustic models can be predicted from the clean acoustic models by combining them with a model of the noise. With the compensated noisy acoustic model, decoding could be performed directly using unaltered noisy observations. The next few subsections will discuss various methods of deriving the noisy model. Firstly the adaptive methods SPR and MLLR are reviewed, and then we will discuss the predictive forms PMC and VTS.

2.3.1 Single Pass Re-training

When re-training acoustic models directly with corrupted speech training data, the state posteriors may be poor due to noise. This will reduce the variations between states and blur the boundaries between distinct regions of speech [33]. SPR [33] is a method of re-estimating the acoustic models that avoids these issues. If a stereo database is available, the state posteriors can be estimated using clean speech while the distribution parameters are estimated using the noise-corrupted data. For example the noise compensated model mean may be estimated as follows

$$\hat{\boldsymbol{\mu}}_{sm} = \frac{\sum_{t=1}^{T} \gamma_{smt}\, \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_{smt}} \tag{2.7}$$

where γ_{smt} is the component posterior obtained from clean observation data. This represents an ideal form of model compensation since the state posteriors and component weights are estimated from clean data, but the distribution parameters are the ML estimates for noisy data.

With SPR though, the corrupted speech distributions may still be badly modeled since each Gaussian distribution is only shifted and scaled, whereas corrupting noise may yield a bimodal distribution. This is a general problem for all model compensation techniques that yield a Gaussian distribution as the compensated distribution for each Gaussian in the uncompensated acoustic model. In reality, a stereo database is not usually available and the matched noisy data with labels is also often limited. For data from the target environment without labels, we could first recognize them and use the erroneous transcripts as training supervisions. The errors, however, would cause an imperfect model estimation and sometimes may even lead to degradation. SPR is also a limited off-line compensation technique not suitable for varying acoustic environments. It is unfeasible to have the entire training database on-line and corrupt it using samples of the current noise to re-train the model parameters. Nevertheless, SPR, when possible, is a useful method for evaluating model compensation schemes since it provides a reasonable upper limit baseline.

2.3.2 Maximum Likelihood Linear Regression

MLLR adaptation [60, 61] estimates an affine transformation of the acoustic model parameters. The transformation maximizes the likelihood of the available adaptation data. Since the amount of adaptation data is usually limited compared to the amount of data available for training the acoustic models, it is useful to share the data such that a single transform is estimated from observations associated with many components. MLLR transforms have the following form

$$\hat{\boldsymbol{\mu}}_{sm} = \boldsymbol{A}^{(r_m)}\,\boldsymbol{\mu}_{sm} + \boldsymbol{b}^{(r_m)} \tag{2.8}$$

This transforms the component mean $\boldsymbol{\mu}_{sm}$, estimated in training conditions, to an adapted mean $\hat{\boldsymbol{\mu}}_{sm}$, such that it matches the test adaptation conditions. The superscript r_m indicates that the transform applied to acoustic model component m is based on the regression class r that component m belongs to. The total number of class R is usually small, especially compared to the number of model components. The clustering may be represented using a regression class tree, an example of which is shown in Figure 2.5.

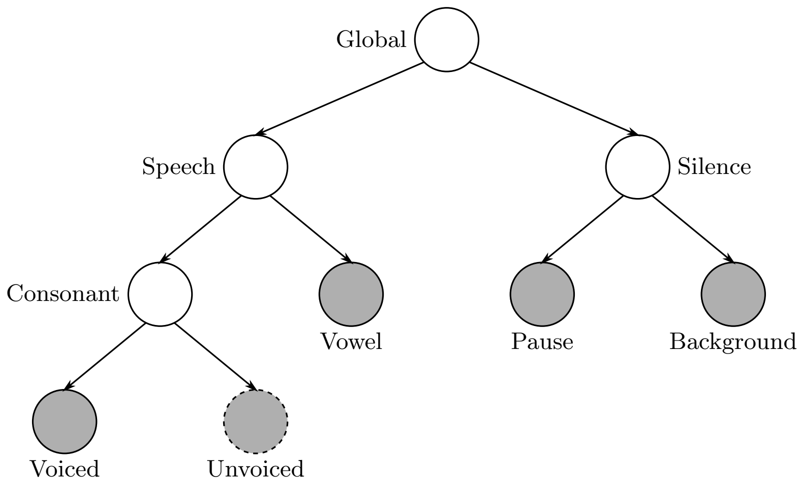


Figure 2.5: An example regression tree for adaptation.

If sufficient data is available, then all the base classes, the leaf nodes in the tree, may each have their own transform. However if there is insufficient data to reliably estimate a transform, as indicated by the dashed node, the class may revert back to using the transform of its parent. The adaptation data for estimating this transform is aggregated from its children. How much data is sufficient for a transform to be estimated is empirically determined by a split threshold; this depends on the complexity of the transform, *e.g.* a diagonal matrix transform requires less data than a full matrix. The use of a regression tree gives an elegant way to scale the number of transforms to the available data. In the example tree, the unvoiced consonant models would use the consonant transform, which is trained on the combined data from voiced and unvoiced consonant observations. In practice a data-driven clustering approach is commonly used to generate regression trees [62, 63]. This may be achieved through k-means clustering and a Kullback-Leibler (KL) divergence measure [63, 64] or simpler centroid-splitting with an Euclidean distance measure [65].

MLLR is often compared to MAP adaptation [59]. MAP produces an adapted model set that may be considered a weighted combination of well-trained, but mismatched, prior models and those estimated from the limited matched test adaptation data. It was shown that MLLR is more effective with less adaptation data than MAP for speaker adaptation, however, with an adequate amount of data MAP outperforms MLLR [13]. This is because MAP has greater flexibility to individually update each acoustic model component.

A slightly different form of affine transformation is further proposed by constraining the transformation of both the mean and variance model parameters to be the same

$$\hat{\boldsymbol{\mu}}_{sm} = \boldsymbol{A}^{(r_m)}\,\boldsymbol{\mu}_{sm} + \boldsymbol{b}^{(r_m)} \tag{2.9}$$

$$\hat{\boldsymbol{\Sigma}}_{sm} = \boldsymbol{A}^{(r_m)}\,\boldsymbol{\Sigma}_{sm}\boldsymbol{A}^{(r_m)\top}. \tag{2.10}$$

This constrained form is called Constrained MLLR [60] or feature-based MLLR (fM-LLR) [66] as the transforms can be efficiently applied in the feature space.

The estimation of these adaptation transforms always requires a transcription of the adaptation data. If the transcription is known, the adaptation is supervised; otherwise, in an unsupervised training, an initial recognition pass over the data gives a hypothesized transcription. A moderate initial recognition error rate is crucial; otherwise the adaptation may even degrade the performance.

2.3.3 Parallel Model Combination

PMC combines separate noise and speech models to form a corrupted speech model directly for use in the recognition process. It assumes the component posteriors remain unchanged in noisy speech [67]. Therefore only the model component distributions need to be updated. In non-iterative forms of PMC, each clean speech model component is combined with the noise model via a mismatch function to yield an updated component. Specific additive, convolutional, additive and convolutional, and bandwidth limited channel mismatch functions can be found in [52]. The log-normal approximation is a popular and efficient choice that assumes the sum of two log-normal distributions is approximately log-normal, however it cannot be applied with delta and delta-delta parameters due to the resulting complexity of the forms [68]. Another approximation is the log-add, which may be used to update the component means of the static dimensions

$$\begin{aligned}\mu_{y,i}^{(sm)} &= \log\left(\exp(\mu_{x,i}^{(sm)}) + \exp(\mu_{z,i})\right)\\ &= \mu_{x,i}^{(sm)} + \log\left(1 + \exp(\mu_{z,i} - \mu_{x,i}^{(sm)})\right)\end{aligned} \tag{2.11}$$

where all the parameters belong to the log-spectral domain.

As discussed with SPR, the transform of each Gaussian component in the clean model to reflect the noise does not give a good model of the overall corrupted speech distribution. The iterative PMC addresses this issue by representing each component with multiple components and iteratively re-estimating the GMM modeling the corrupted speech based on state alignments from the clean speech model [33]. This increases the number of components in the overall system. Alternatively, a data-driven iterative PMC [33] directly estimates the corrupted speech distribution by drawing sample corrupted speech vectors from combinations of the clean and noise models to re-estimate the GMM on a per state basis. The efficient log-add approximation can be used to combine the model and the overall number of components can remain unchanged; however, around 25$\sim$1000 observations need to be generated per Gaussian in the system [52]. This data-driven PMC could give results equivalent to matched systems at levels below 20dB Signal-to-Noise Ratio (SNR) [67]. However, this iterative estimation is computationally expensive.

2.3.4 Vector Taylor Series Model Compensation

As the discussion of PMC shows, deriving a corrupted speech output distribution, given the clean acoustic model and a noise model, is not straightforward. Directly determining the expected value of equation (2.4) is problematic due to the non-linear effect of noise on cepstral speech features. For convenience it is repeated here without the domain superscript for brevity and we use inverse DCT rather than the pseudo-inverse here for the purpose of theoretical derivation:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{u} + \boldsymbol{C}\mathbf{log}\Big(\mathbf{1} + \mathbf{exp}\big(\boldsymbol{C}^{-1}(\boldsymbol{z} - \boldsymbol{x} - \boldsymbol{u})\big)\Big). \tag{2.12}$$

Many approximations to this function have been proposed, such as selecting the maximum of either the noise or speech, *i.e.* noise masking [69] or PMC as discussed in the previous section. Another approach is to linearize it with a truncated VTS [34, 70, 71] to individually update each model component. The first-order VTS approximation of the static corrupted speech may be expressed as

$$\boldsymbol{y}_{\text{vts}} = \boldsymbol{y}|_{\mu_0^{(sm)}} + \boldsymbol{J}_x^{(sm)}(\boldsymbol{x} - \boldsymbol{\mu}_x^{(sm)}) + \boldsymbol{J}_z^{(sm)}(\boldsymbol{z} - \boldsymbol{\mu}_z) + \boldsymbol{J}_u^{(sm)}(\boldsymbol{u} - \boldsymbol{\mu}_u) \tag{2.13}$$

where $|_{\mu_0^{(sm)}}$ indicates that the Taylor series expansion is evaluated at the point $\mu_0^{(sm)} = \{\boldsymbol{\mu}_x^{(sm)}, \boldsymbol{\mu}_z, \boldsymbol{\mu}_u\}$ with the clean speech component mean $\boldsymbol{\mu}_x^{(sm)}$, the additive noise mean $\boldsymbol{\mu}_z$ and channel noise mean $\boldsymbol{\mu}_u$.

The Jacobian matrices involved in equation (2.13) could be computed as follows

$$\begin{aligned}
\boldsymbol{J}_x^{(sm)} &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\bigg|_{\mu_0^{(sm)}} \\
&= \left[\boldsymbol{\nabla}_{\boldsymbol{x}} y_1\Big|_{\mu_0^{(sm)}} \quad \cdots \quad \boldsymbol{\nabla}_{\boldsymbol{x}} y_i\Big|_{\mu_0^{(sm)}} \quad \cdots \quad \boldsymbol{\nabla}_{\boldsymbol{x}} y_{D_s}\Big|_{\mu_0^{(sm)}}\right]^\top \\
&= \begin{bmatrix} \frac{\partial y_1}{\partial x_1}\Big|_{\mu_0^{(sm)}} & \cdots & \frac{\partial y_1}{\partial x_{D_s}}\Big|_{\mu_0^{(sm)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_{D_s}}{\partial x_1}\Big|_{\mu_0^{(sm)}} & \cdots & \frac{\partial y_{D_s}}{\partial x_{D_s}}\Big|_{\mu_0^{(sm)}} \end{bmatrix} = \boldsymbol{I} - \boldsymbol{C}\boldsymbol{F}\boldsymbol{C}^{-1} && (2.14) \\
\boldsymbol{J}_u^{(sm)} &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{u}} = \boldsymbol{J}_x^{(sm)} && (2.15) \\
\boldsymbol{J}_z^{(sm)} &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}} = \boldsymbol{C}\boldsymbol{F}\boldsymbol{C}^{-1} && (2.16)
\end{aligned}$$

where D_s is the number of static components in the feature vectors. $\boldsymbol{I}$ is the identity matrix. $\boldsymbol{C}$ and $\boldsymbol{C}^{-1}$ are the DCT and IDCT matrices. The elements of the diagonal matrix $\boldsymbol{F}$ are computed as

$$\begin{aligned}
f_{ii} &= \frac{\exp((\boldsymbol{c}_{\bar{i}}^{-1})^\top(\boldsymbol{z} - \boldsymbol{x} - \boldsymbol{u}))}{1 + \exp((\boldsymbol{c}_{\bar{i}}^{-1})^\top(\boldsymbol{z} - \boldsymbol{x} - \boldsymbol{u}))}\bigg|_{\mu_0^{(sm)}} \\
&= \frac{\exp((\boldsymbol{c}_{\bar{i}}^{-1})^\top(\boldsymbol{\mu}_z - \boldsymbol{\mu}_x^{(sm)} - \boldsymbol{\mu}_u))}{1 + \exp((\boldsymbol{c}_{\bar{i}}^{-1})^\top(\boldsymbol{\mu}_z - \boldsymbol{\mu}_x^{(sm)} - \boldsymbol{\mu}_u))}
\end{aligned} \tag{2.17}$$

The term $\boldsymbol{c}_{\bar{i}}$ is a row vector corresponding to the ith row of the DCT matrix $\boldsymbol{C}$. The terms f_{ii} vary from 0 to 1 depending on the ratio of the speech to the noise. If the noise level $\boldsymbol{\mu}_n$ is greater than the speech $\boldsymbol{\mu}_x^{(sm)}$ in the log-spectral domain, then $f_{ii} \to 1$ and $\boldsymbol{J}_x^{(sm)}$ tends to zero; otherwise if little noise is present, $f_{ii} \to 0$ and $\boldsymbol{J}_x^{(sm)}$ tends to identity. The term $\boldsymbol{J}_z^{(sm)}$ behaves in the opposite manner to $\boldsymbol{J}_x^{(sm)}$.

With this linear approximation, we could hence derive the noisy mean of all the Gaussians using maximum likelihood estimations. Whereas for the noise variance, an iterative first-order gradient-based optimization scheme is often used.

2.4 Uncertainty-based Scheme

Another category of techniques integrate both the front-end and the back-end jointly and treat the unknown target environment as uncertainties in speech. They have been loosely applied in a variety of contexts to describe various robustness techniques for ASR. The concept of uncertainty decoding is distinct from uncertain observation decoding [72, 73] and uncertain model parameters [74]. For Missing Feature Theory (MFT) [75, 76], data imputation with soft data is an observation uncertainty approach. In contrast, data marginalization can be construed as a limited form of front-end uncertainty decoding, restricted to the spectral domain, and where features are either completely certain or uncertain. These different uncertainty-based techniques are elaborated in the following subsections.

2.4.1 Observation Uncertainty

Feature compensation schemes, such as speech enhancement, provide an estimate of the clean speech to the decoder. This assumes the enhancement is exact and the estimate is the true value. However, it may be reasonable to consider that the de-noising process is not exact and there is some residual uncertainty that may be passed to the decoder. Hence in the observation uncertainty approach, instead of using a point estimate of the features as shown in Figure 2.4, the clean speech posterior is passed to the decoder as shown in Figure 2.6.

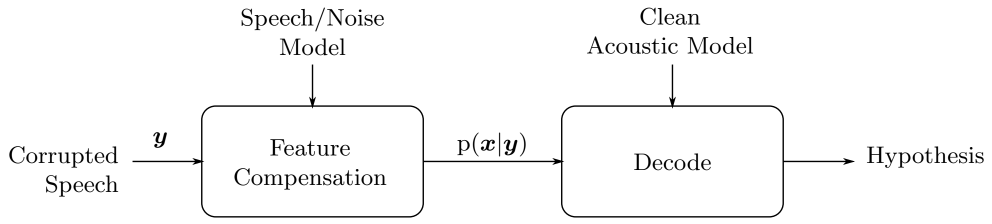


Figure 2.6: Feature compensation with uncertain observations.

If the clean speech estimate is now considered a multivariate Gaussian distribution, the decoding likelihood requires an integration over the true clean speech space. Some enhancement schemes have been extended to provide this uncertainty, such as samples computed from the formants [77], a polynomial function of the SNR [73], a parametric model of the clean speech [78], Weiner filtering [79] and a particle filter [80]. Although it is widely used, including the variance has not been well motivated in the literature.

Perhaps this is why the variances need to be scaled before being added in the model-based feature enhancement with uncertainty [81]; the variances are considered too large [82], adding the variance of the delta-delta features did not improve results [78, 82] or there is degradation compared to the non-uncertainty form in high SNRs [79].

2.4.2 Uncertainty Decoding

Uncertainty decoding first appears in the context of SPLICE [83] and Algonquin [84]. The unknown noise parameter is directly integrated out through the whole noise parameter space. The integration is commonly approximated by the corrupted speech conditional distribution. To avoid a 3-dimensional decoding, it assumes the noise is stationary and a single noise condition is involved. Ideally, the form of the corrupted speech conditional distribution should be independent of the acoustic model complexity and make the marginalization with the clean speech models tractable. If the conditional distribution takes a Gaussian-distributed form, then the integral is also a Gaussian distribution with a variance that is the sum of the variances of the two parts of the integrand. Hence uncertainty decoding may be viewed as passing the corrupted conditional density function to the decoder as shown in Figure 2.7. Examining this distribution in more detail may yield insights into what approximations are appropriate to best model it with parameters that are efficient to compute, yet minimizing the cost of updating the acoustic model.

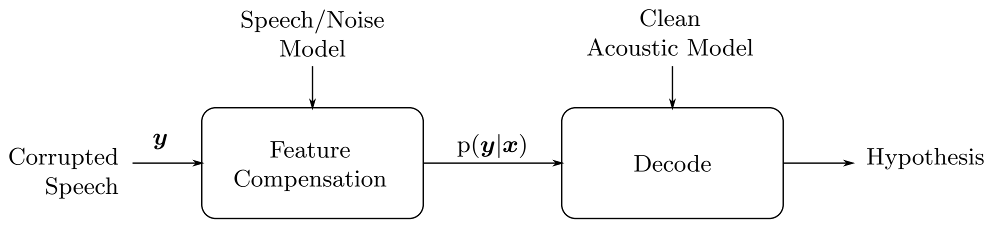


Figure 2.7: Uncertainty decoding.

There are several approaches to model the corrupted speech conditional distribution. Using a joint distribution of the clean and corrupted speech to derive it leads to joint uncertainty decoding [85]. Approximating it through an application of Bayes' rule and using the SPLICE form of the clean speech posterior gives the SPLICE with uncertainty form [83].

2.4.3 Missing Feature Theory

Missing Feature Theory (MFT) treats heavily noise-corrupted elements of a spectral domain feature vector as unreliable (missing) and those less distorted as reliable (present) [75, 76]. This is motivated from studies indicating humans can recognize speech from a "very small proportion of clean frequency channels at any point in time" [86]. Detecting missing areas of speech is a key aspect of MFT. It is done using a variety of possible measures including SNR-based ones [75, 86], "harmonicity", a combination [87] or a Bayesian classifier using a variety of features [88]. It is conducted at a spectral level because de-correlating transforms such as the DCT spreads single unreliable spectral channels to all dimensions in the cepstral space. Once parameters have been labeled as missing or present, the missing ones can be restored [88] or marginalized over [75, 87]. Thus missing feature techniques fall under two approaches: imputation and marginalization. Both identify regions of the noisy spectral feature vector $\boldsymbol{y}$ that are missing. For example

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_{\mathrm{p}} \\ \boldsymbol{y}_{\mathrm{m}} \end{bmatrix} \tag{2.18}$$

where $\boldsymbol{y}_{\mathrm{p}}$ are the components that are considered present, and $\boldsymbol{y}_{\mathrm{m}}$ are the missing values. The total number of elements on both the left and right side of the equation are the same.

The two approaches differ in how to handle missing areas. Imputation replaces missing values with estimated values. The reconstructed feature vector is then used as if it was a clean speech vector, and is thus similar to enhancement schemes. Marginalization classifies solely on $\boldsymbol{y}_{\mathrm{p}}$ by marginalizing out the missing components. Decoding proceeds only with the present features. While this form of decoding with missing features is efficient, a form of bounded marginalization gives much better results by providing a bound on the integration. It was concluded that marginalization gave superior accuracy to imputation in the spectral domain [75, 88]. However, marginalization requires changes to the recognizer and is limited to using only spectral features whereas imputation can be used as a general front-end enhancement system by transforming the restored features into the cepstral domain [88]. The difficulty for marginalization in MFT is that it is carried out in the spectral domain, while most state-of-the-art systems operate in the cepstral domain. It also unnecessarily applies a hard decision on the reliability of the features, whereas uncertainty decoding is domain-agnostic and also avoids this hard decision.

In [86, 87], MFT imputation has been modified to use a soft mask. Instead of applying a hard decision to each channel, the decision is a weighted sum of the present and missing outcomes. It has also been extended by considering the features as soft data [86]. This applies to unreliable, missing features and is similar to observation uncertainty methods described here. Distributions are evaluated as forms for the evidence probability density function rather than a standard Gaussian. The delta form is equivalent to data imputation; the uniform distribution was found to be better than the bounded Gaussian. Overall, [76] has found that recognition with data imputation transformed to the cepstral domain is superior to spectral domain marginalization. It also concludes that marginalization approaches in the cepstral domain are generally ineffective as shown in [89].

2.5 Noise Estimation

For many noise compensation techniques a model of the noise is necessary. Frequently, an additive noise model is estimated from background, non-speech areas, such as the first and last 10 $\sim$ 30 frames of each utterance. This has worked well for Algonquin enhancement [90], VTS feature compensation [91], MBFE [92], and Weiner filtering [79]. However, a robust voice activity detector is required and generally detecting speech becomes more difficult as the noise level increases. Furthermore, while this approach may provide a good model for short utterances, some sentences may be sufficiently long that the noise environment changes while speech continues to be spoken. Even on the Aurora-2 [58], which is a short artificially corrupted digit string recognition task, some gains are obtained by updating the model during the speech; for example, in [93] the noise model is updated every 100 frames.

It is not straightforward to estimate a convolutional noise model using just the background segments of an utterance. In conjunction with a background estimated additive noise model, the channel noise may be estimated over the entire utterance using EM [94]. In contrast, [34] provides an EM-based framework to estimate both the means of the additive and convolutional noise in a maximum likelihood fashion in the log-spectral domain for only the static features. This allows for unsupervised noise estimation of the full noise model whilst the speaker is still speaking. The form of maximization of the static additive and convolutional noise means described here is based on the maximum likelihood formulation introduced in [34], but in the cepstral domain. The first-order VTS approximation may be used to express the static corrupted speech mean as a function of initial and new additive and convolutional noise means

$$\begin{aligned}\hat{\boldsymbol{\mu}}_y^{(sm)} &\approx \mathcal{E}\Big\{\boldsymbol{y}|_{\mu_0^{(sm)}} + \boldsymbol{J}_x^{(sm)}(\boldsymbol{x}-\boldsymbol{\mu}_x^{(sm)}) + \boldsymbol{J}_z^{(sm)}(\boldsymbol{z}-\boldsymbol{\mu}_z) + \boldsymbol{J}_u^{(sm)}(\boldsymbol{u}-\boldsymbol{\mu}_u)\Big\} \\ &= \boldsymbol{\mu}_y^{(sm)} + \boldsymbol{J}_z^{(sm)}(\hat{\boldsymbol{\mu}}_z-\boldsymbol{\mu}_z) + \boldsymbol{J}_u^{(sm)}(\hat{\boldsymbol{\mu}}_u-\boldsymbol{\mu}_u) \qquad (2.19)\end{aligned}$$

assuming that the speech and noise are independent. The terms with the Jacobian matrices will vanish when the estimated value and the current value of the noise means converge. The noise means are estimated in an ML fashion such that when they are combined with the clean speech acoustic model, they maximize the likelihood of some corrupted speech data from the mismatched test condition. The clean acoustic model parameters and the static additive noise variance are unchanged throughout the noise mean estimation process. The component posterior is computed from the complete data set which requires a hypothesis from an initial decoding run. The noisy speech acoustic model used to compute the posteriors is generated by combining the clean speech model and the current noise model using VTS compensation, but only for the static cepstral dimensions and with the zero-order form of the corrupted speech mean. The maximization step differs by using the form given in equation (2.19). To find updates of the additive and convolutional noise, the auxiliary function is differentiated with respect to the parameter sought and equated to zero to solve. A key simplifying factor is that the Jacobian matrices are considered constant although they are functions of the noise. The detailed derivations could be found in [36].

2.6 Summary

This chapter reviews the existing literature of noise-robustness techniques developed mainly for the GMM-HMM systems. They could be generally categorized into three broad classes, namely feature-based enhancement, model-based compensation and uncertainty decoding. For feature-based approaches, there is no assumption of the back-end models used for AMs. They are hence directly applicable to the DNN-HMM systems. While for the model-based approaches, Gaussian models are required. Moreover, the complex nonlinear interaction between noise and speech and the layered nonlinear transformations involved in the DNN make them much more difficult to be integrated into DNNs. For the uncertainty decoding, similarly, the model-based uncertainty decoding is hard to be incorporated into DNNs. But the MFT techniques mainly focus on features and can be investigated for the use in DNNs.

Chapter 3

Deep Neural Network

The Deep Neural Network (DNN) is a layered model mimicking the hierarchical structure of the human perception system. It shares the same biological motivation as the well known Multi-Layer Perceptron (MLP). The major difference comes from the depth of the model and how those many layers are optimized. Historically, MLPs use only one or two hidden layers due to the limited computation power and difficulties in optimizing models with too many layers. Those have only become practical recently through the use of huge clusters or GPU-based machines and the discovery of layer-wise training either generatively or discriminatively. Each layer in a DNN nonlinearly transforms its input representation into a higher level, more abstract representation that better models the underlying factors of the data. With multiple layers' nonlinear transformations, different levels of representation are learned. Those from the lower layers of the DNN usually capture more detailed feature variations in the original observation space; those from the higher layers reflect more about the structure and abstract concepts that lead to better discrimination among observations from different classes. In this chapter, we first review the MLP model and then discuss the DNN. Following that, we present how the DNN is adopted for acoustic modeling, *i.e.* the hybrid DNN-HMM AM. With this hybrid DNN-HMM AM, we will justify its noise robustness and investigate the effects of conventional noise-robustness techniques. Based on those investigations, we propose to address DNN's noise-robustness from a representation learning perspective.

3.1 Deep Neural Network Acoustic Model

3.1.1 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is a feed forward neural network model that maps sets of input data onto a set of appropriate outputs. It is one of the widely used neural network structures in speech recognitions, due to its simplicity compared with other

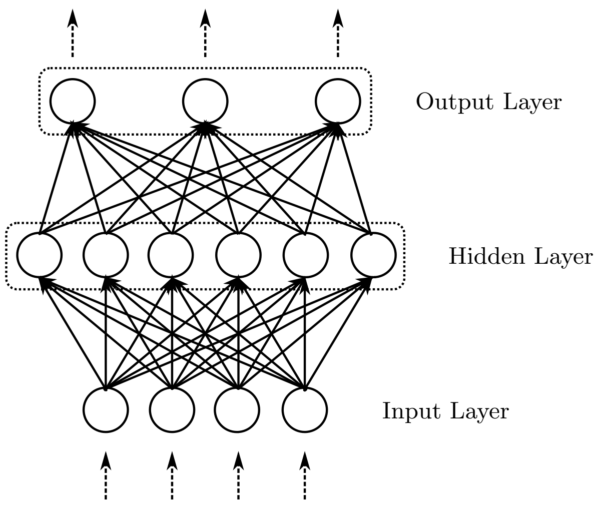


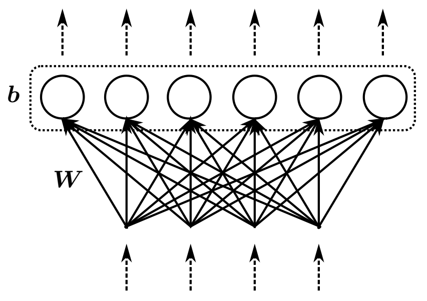
Figure 3.1: The structure of a neural network with 1 hidden layer.

types of neural networks such as the Recurrent Neural Network [95] and the Time Delay Neural Network [96, 97, 98]. In this work, we mainly focus on MLPs and will also refer to them as Neural Networks (NNs) for simplicity.

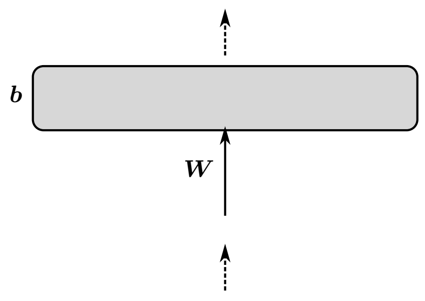
An NN typically consists of multiple layers of nodes with each layer connected to another in a feed forward manner. These inter-layer connections can be of either dense or sparse. In practice, a full connectivity is commonly adopted to simply avoid deciding which connection link to discard. Based on each layer's functionality, it is usually referred to as an input layer, a hidden layer or an output layer, as shown in Figure 3.1. The NN with more than two hidden layers is now commonly referred to as a deep NN (DNN). The input layer of the NN has no computation capability as it simply attaches the observations to the network. Each hidden layer of the NN takes in the activations of the layer below and computes a new set of nonlinear activations for the layers above. The output layer generates either a value in regression or a posterior vector in classification using the activations from the last hidden layer. The NN could thus be further deemed as a cascade of many simpler nonlinear computation layers as illustrated in Figure 3.2.

For an NN with L computational layers, there will be one input layer, $(L-1)$ hidden layers and one output layer. As the input layer has no computation involved, it is hence not counted. The input to the lth hidden layer is the activation of the layer below, $\boldsymbol{h}_{l-1}$, with $\boldsymbol{h}_0$ being the input. Each hidden layer computes the activation $\boldsymbol{h}_l$ via a linear transformation using a weight matrix $\boldsymbol{W}_l$ and a bias vector $\boldsymbol{b}_l$ followed by a nonlinear squashing function $f_l(\boldsymbol{x})$ [99], *i.e.*

$$\boldsymbol{h}_l = f_l(\,\boldsymbol{W}_l\,\boldsymbol{h}_{l-1} + \boldsymbol{b}_l\,) \quad \text{for} \quad 1 \leqslant l < L \tag{3.1}$$



(a) Graphical model representation.



(b) Simplified representation.

Figure 3.2: A single computation layer of neural networks.

where the nonlinear function $f_l(\boldsymbol{x})$ usually operates in an element-wise fashion on the input vector. The most commonly used hidden activation function is the sigmoid logistic function, which is defined as follows for a K-element input vector $\boldsymbol{x}$:

$$\begin{aligned}\phi(\boldsymbol{x}) &= \text{sigmoid}(\boldsymbol{x}) \\ &= \begin{bmatrix} \frac{1}{1+\exp(-x_1)} & \frac{1}{1+\exp(-x_2)} & \cdots & \frac{1}{1+\exp(-x_K)} \end{bmatrix}^\top . \end{aligned} \tag{3.2}$$

Although different hidden layers or even different units in the same hidden layer could have different activation functions, the same nonlinearity is commonly used in all the hidden units for simplicity without affecting the NN's approximation capability. Given the input, the computation of the hidden activations is independent. Each sigmoid hidden unit could be deemed as carrying out a logistic linear regression feature extraction process [100] which refines the input representation to a better one. With multiple hidden layer transformations, the final set of hidden activations *i.e.* $\boldsymbol{h}_{L-1}$ is a high level abstract representation of the observation. It has reduced noise variations and more task-specific information. Using this representation, the discrimination among different classes is much clearer and the prediction can be done using a weak classifier.

The output layer *i.e.* the Lth layer of the NN thus acts as the functional role of the whole NN that is to predict either a value or a class label. It simply carries out a similar linear regression as hidden layers do using a weight matrix $\boldsymbol{W}_L$ and a bias vector $\boldsymbol{b}_L$. However, a different task-dependent nonlinear function is usually adopted. For regression tasks, a linear or sigmoid function is often used; while for classification tasks, the softmax function is adopted which converts the values of arbitrary ranges into a probabilistic representation. The generated output values could be interpreted as posterior probabilities for each of the classes given the input observation. For a

K-element vector $\boldsymbol{x}$, the softmax function is defined as

$$\begin{aligned}\psi(\boldsymbol{x}) &= \text{softmax}(\boldsymbol{x}) \\ &= \frac{1}{\sum_{j=1}^{K}\exp(x_j)}\begin{bmatrix}\exp(x_1) & \exp(x_2) & \cdots & \exp(x_K)\end{bmatrix}^{\top}.\end{aligned} \tag{3.3}$$

The computations involved in an L-layer NN could be summarized in the following equations:

$$\boldsymbol{h}_l = \phi(\,\boldsymbol{W}_l\,\boldsymbol{h}_{l-1} + \boldsymbol{b}_l\,) \quad \text{for} \quad 1 \leqslant l < L \tag{3.4}$$

$$\boldsymbol{p} = \psi(\,\boldsymbol{W}_L\,\boldsymbol{h}_{L-1} + \boldsymbol{b}_L\,) \tag{3.5}$$

where $\boldsymbol{h}_0$ is the input vector forwarded to the NN and $\boldsymbol{p}$ is the output posterior vector generated from the NN assuming a classification task.

The parameters for an L-layer NN are $\{(\boldsymbol{W}_1, \boldsymbol{b}_1), (\boldsymbol{W}_2, \boldsymbol{b}_2), \ldots, (\boldsymbol{W}_L, \boldsymbol{b}_L)\}$. They are usually randomly initialized and then discriminatively updated using the Error Back-Propagation (EBP) algorithm [101]. It evaluates the prediction cost at the output layer by measuring the discrepancy between the target outputs and the actual outputs produced for each training case and back-propagates the error derivatives through all the hidden layers to the input. For classification problems, a natural cost function, $\mathcal{E}$, is the cross entropy between the target probability $\boldsymbol{d}$ and the output of the NN $\boldsymbol{p}$,

$$\mathcal{E} = -\sum_i d_i \log p_i \tag{3.6}$$

where the target probabilities d_i, typically taking values of one or zero, are the supervision information provided to train the NN. By computing the gradient of the cost function with respect to each of the model parameters, we could update the parameters for the lth layer by

$$\boldsymbol{W}_l(\tau) = \boldsymbol{W}_l(\tau-1) + \Delta\boldsymbol{W}_l(\tau) \tag{3.7}$$

$$\boldsymbol{b}_l(\tau) = \boldsymbol{b}_l(\tau-1) + \Delta\boldsymbol{b}_l(\tau) \tag{3.8}$$

where

$$\Delta\boldsymbol{W}_l(\tau) = \alpha\,\Delta\boldsymbol{W}_l(\tau-1) + \eta\,\frac{\partial\mathcal{E}}{\partial\boldsymbol{W}_l(\tau-1)} \tag{3.9}$$

$$\Delta\boldsymbol{b}_l(\tau) = \alpha\,\Delta\boldsymbol{b}_l(\tau-1) + \eta\,\frac{\partial\mathcal{E}}{\partial\boldsymbol{b}_l(\tau-1)}. \tag{3.10}$$

The learning rate η controls the speed of weight changes at each update iteration τ and the momentum coefficient α smooths the gradient computed for the current weight, thereby damping oscillations across ravines and speeding progress down ravines. For both efficiency and reliability considerations, a small random mini-batch of training samples rather than the whole training set is commonly used at each update. To avoid over-fitting, large weights can be penalized in proportion to their squared magnitude, or the learning can simply be terminated at the point at which the performance on a held-out validation set starts degrading [15].

Theoretically speaking, single-hidden-layer NNs are capable of approximating any function, given that sufficient hidden units are available [102, 103]. However, it is impractical to train a single hidden layer NN with infinite hidden units due to the limited training samples and computation resources. Using multiple hidden layers with moderate dimensions has been shown to yield better performance. However the gradient-based EBP is only effective for one or two hidden layers [104, 105, 106]. With more than two hidden layers, the gradient diminishing problem usually leads to local optima for the EBP algorithm [107]. Those models generalize poorly on unseen data. However, NNs with many hidden layers and units per layer are flexible models and are capable of modeling very complex and highly nonlinear relationships between inputs and outputs. It is rather important for high quality acoustic modeling. At the same time, the increased amount of model parameters may capture some spurious regularities that are an accidental property of particular examples in the training set. This is also one of the major reasons for poor generalization. Weight penalties or early stopping could be adopted to address the over-fitting problem but only by removing much of the modeling power. Very large training sets [108] can reduce the over-fitting while preserving the modeling power, but they make the training very computationally expensive. A better training method that could fully explore the training information to build multiple layers of nonlinear feature abstractions would be the key.

3.1.2 Deep Neural Network

DNNs are effectively MLPs with many ($\geqslant 2$) hidden layers. The computation of an L-layer DNN is the same as those listed in equations (3.4) and (3.5). The key is how the diminishing gradient problem of the EBP learning algorithm is addressed. In 2006, a fast learning algorithm was proposed for Deep Belief Nets (DBNs) [109], which provides a practical way of building deep layered neural networks and triggered great interest in learning deep models. The essence of learning deep models lies in the unsupervised generative pre-training utilizing Restricted Boltzmann Machines (RBMs). This generative pre-training puts the model into a space that is near to a better optimum. It hence enables the learning of deep models with better generalizations. For DNNs adopted in this work, we simply borrow the RBM pre-training process for model initialization and then fine-tune it using the standard EBP algorithm with supervision labels.

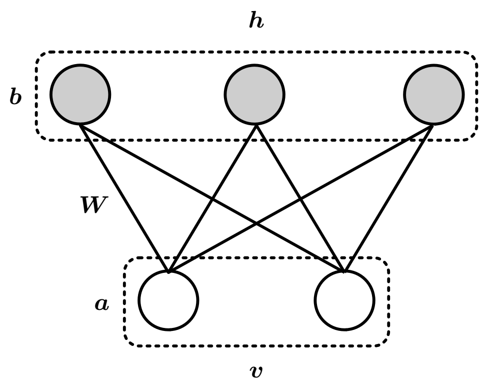


Figure 3.3: A Restricted Boltzmann Machine.

Generative Pre-Training

Instead of directly learning the model that discriminates between different classes, we start from understanding the underlying structure of the data. This is achieved through a layer-wise learning algorithm [109] that gradually estimates a new layer of nonlinear feature transformation on top of the representations generated from the existing layers. More specifically, at each time an RBM layer is fitted to the current "data".

An RBM is an undirected generative model (Figure 3.3) consisting of a layer of stochastic binary visible units that represent the binary input data and a layer of stochastic binary hidden units that model the significant non-independences between the visible units [110]. The connections only exist between the visible and the hidden units and there are no visible-visible or hidden-hidden connections. This structure avoids the "explaining away" problem [111] often encountered in other latent variable models. The RBM belongs to the Markov Random Field (MRF) model family, but it differs from most MRFs in several ways: it has a bipartite connectivity structure, it does not usually share weights between different units, and a subset of the variables are unobserved, even during training. Moreover, this bipartite-structured undirected RBM is ideal for layer-wise pre-training.

Mathematically, an RBM defines the joint probability of the observable vector, $\boldsymbol{v}$, and the latent vector, $\boldsymbol{h}$, via an energy function, E,

$$p(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}\exp(-E(\boldsymbol{v},\boldsymbol{h})), \quad Z = \sum_{\boldsymbol{v}',\boldsymbol{h}'} \exp(-E(\boldsymbol{v}',\boldsymbol{h}')) \tag{3.11}$$

where Z is the so-called partition function.

Efficient RBM Learning

The RBM energy function defining the joint configuration $(\boldsymbol{v}, \boldsymbol{h})$ of the visible and hidden units is given by

$$\begin{aligned} E(\boldsymbol{v}, \boldsymbol{h}) &= -\boldsymbol{a}^\top \boldsymbol{v} - \boldsymbol{b}^\top \boldsymbol{h} - \boldsymbol{h}^\top \boldsymbol{W} \boldsymbol{v} \\ &= -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} h_j w_{ji} v_i \end{aligned} \tag{3.12}$$

where v_i and h_j are the binary states of visible unit i and hidden unit j, a_i and b_j are the corresponding biases, and w_{ij} is the weight between them. The RBM model parameters are $\{\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}\}$. Based on the joint probability defined in equation (3.11), the probability that a particular visible vector, $\boldsymbol{v}$, is generated from the model could be obtained by summing over all possible hidden vectors

$$p(\boldsymbol{v}) = \frac{1}{Z} \sum_{\bar{\boldsymbol{h}}} \exp(-E(\boldsymbol{v}, \bar{\boldsymbol{h}})). \tag{3.13}$$

The derivative of the log probability of a training set with respect to a model parameter is surprisingly simple

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{v}^{(n)})}{\partial w_{ji}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \tag{3.14}$$

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{v}^{(n)})}{\partial a_i} = \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \tag{3.15}$$

$$\frac{1}{N} \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{v}^{(n)})}{\partial b_j} = \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}} \tag{3.16}$$

where N is the size of the training set and the angle brackets are used to denote expectations under the distribution specified by the subscript that follows. The simple derivative in the above equations leads to a simple learning rule for performing stochastic steepest ascent in the log probability of the training data:

$$\Delta w_{ji} = \eta(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}) \tag{3.17}$$

$$\Delta a_i = \eta(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}) \tag{3.18}$$

$$\Delta b_j = \eta(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}}) \tag{3.19}$$

where η is the learning rate. Momentum terms such as those in equations (3.9) and (3.10) are also commonly included, which are omitted in the above equations just for simplicity.

The absence of direct connections between hidden units in an RBM makes it much easier to get an unbiased sample of $\langle v_i h_j \rangle_{\text{data}}$. Given a randomly selected training case, $\boldsymbol{v}$, the binary state of each hidden unit j, h_j, is set to one with the probability of

$$p(h_j = 1|\boldsymbol{v}) = \phi(b_j + \boldsymbol{w}_{j\cdot}\boldsymbol{v}) = \phi(b_j + \sum_i w_{ji} v_i) \tag{3.20}$$

and $v_i h_j$ is then an unbiased sample of $\langle v_i h_j \rangle_{\text{data}}$. Similarly, the absence of direct connections between visible units also ease the computation of the probability that the state of a visible unit i is turned on given a hidden vector $\boldsymbol{h}$. It can be computed as

$$p(v_i = 1|\boldsymbol{h}) = \phi(a_i + \boldsymbol{h}^\top \boldsymbol{w}_{\cdot i}) = \phi(a_i + \sum_j h_j w_{ji}). \tag{3.21}$$

Getting an unbiased sample of $\langle v_i h_j \rangle_{\text{model}}$, however, is much more difficult. It can be done by starting at any random state of the visible units and performing Gibbs sampling until convergence. Each epoch of Gibbs sampling consists of updating all of the hidden units in parallel using equation (3.20) followed by updating all of the visible units in parallel using equation (3.21). The convergence usually requires a rather long or infinite time. A much faster learning procedure named Contrastive Divergence (CD) has been proposed in [110]. It starts by setting the states of the visible units to a training vector. Then the binary states of the hidden units are all computed in parallel using equation (3.20). Once the binary hidden states have been chosen, a reconstruction of the visible units is produced by setting each v_i to one, if the probability given by equation (3.21) is above 0.5 and 0 otherwise. Finally the hidden units are updated again using the reconstructed visible states. This whole procedure is a full iteration of the contrastive divergence (CD-1). The change in a weight parameter could then be obtained by

$$\Delta w_{ji} = \eta(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{CD-1}}) \tag{3.22}$$

$$\Delta a_i = \eta(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{CD-1}}) \tag{3.23}$$

$$\Delta b_j = \eta(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{CD-1}}) \tag{3.24}$$

Contrastive divergence works well even though it is only a crude approximation to the gradient of the training data log likelihood [110]. Using more iterations of Gibbs sampling before collecting the statistics could give better generative RBMs; but for the purposes of pre-training, CD-1 is sufficient. To suppress sampling noise in the learning, real-valued probabilities rather than binary samples are generally used in the reconstruction of the visible units and the subsequent states of the hidden units. But it is important to use sampled binary values for the first computation of the hidden states because the sampling noise actually acts as a very effective regularizer that avoids over-fitting [112].

Modeling Real-Valued Data

For the ASR problem, real-valued input features are commonly used. To extend the binary RBM for real-valued data, Gaussian noise is added to each visible unit and the RBM energy function is modified to accommodate this modification, which leads to the Gaussian-Bernoulli RBM (GRBM):

$$E(\boldsymbol{v}, \boldsymbol{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} h_j w_{ji} \frac{v_i}{\sigma_i} \tag{3.25}$$

where σ_i is the standard deviation of the Gaussian noise for the visible unit i. The two conditional distributions required for CD-1 learning could be derived accordingly:

$$p(h_j = 1|\boldsymbol{v}) = \phi(b_j + \sum_i w_{ji} \frac{v_i}{\sigma_i}) \tag{3.26}$$

$$p(v_i|\boldsymbol{h}) = \mathcal{N}(v_i; a_i + \sigma_i \sum_j h_j w_{ji}, \sigma_i^2) \tag{3.27}$$

where $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2. Learning the standard deviations of a GRBM is problematic for reasons described in [112]. For pre-training using CD-1, a common practice is to normalize the data so that each coefficient has zero mean and unit variance. We could then simply remove the variable σ by setting it to 1. Furthermore, the noise adding to the visible reconstructions is also bypassed to avoid the errors in determining the noise levels.

Discriminative Fine-Tuning

After training one RBM on the original data, the inferred states of the hidden units can be used as new data for training another RBM that learns to model the dependencies between the hidden units of the first RBM. This procedure can be repeated as many times as desired to produce many layers of nonlinear feature detectors that represent progressively more complex statistical structure in the original data. The pre-trained RBMs are then stacked to form a single multilayer generative model, a DBN [109]. The whole DBN is then updated jointly. Only the bidirectional connections of the top RBM remain and the other RBMs are converted to top-down directed layers(Figure 3.4).

For classification, there is no need to maintain the generative capability of the DBN. Furthermore, some researchers find that the generative pre-training is also not crucial[113, 114]. By growing the network layer by layer using the basic EBP algorithm,

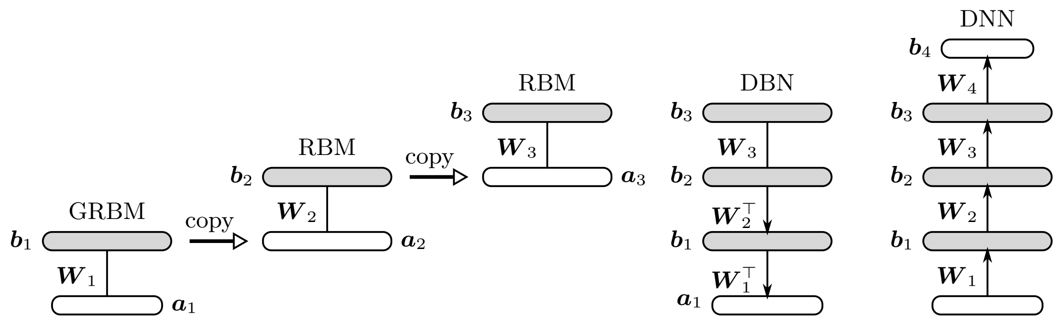


Figure 3.4: A comparison among a Restricted Boltzmann Machine (RBM), a Deep Belief Net (DBN) and a Deep Neural Network (DNN).

they could obtain similar performance to those models pre-trained using RBMs. In this thesis, we still keep the pre-training process and only jettison the probabilistic framework to use the trained stack of RBM weights in the reverse direction as a way of initializing all the feature detection layers in a discriminative DNN (Figure 3.4). This is because the pre-training helps the DNN learning to start from a better seed model using a data-driven approach rather than figuring out manually on different datasets. Besides, the task independent pre-training also plays an important role in the techniques we have developed to improve DNNs' robustness, such as the sharing of RBMs among different sub-context DNNs in the Deep Split Temporal Context (DSTC) system (*cf.* Section 4.2) and between the acoustic model and the mask estimator in the spectral masking system (*cf.* Section 4.3). In this study, after DNN is initialized using pre-trained RBMs, a final randomly initialized softmax layer is usually appended. As conventional shallow MLPs, the whole DNN model is then discriminatively updated using the standard EBP algorithm.

3.1.3 Hybrid DNN-HMM AM

Since the late 1980s there have been attempts to adopt NNs for speech recognition. Some researchers initially tried to emulate HMMs with NNs [115, 116], which strengthened the idea that NNs could be effectively used for speech recognition. Although at that time the emulation of HMMs did not overcome their limitations, the sequential training algorithm introduced in their work have recently been shown to further improve DNNs' performance. Based on the probabilistic interpretation of the NNs' outputs [15], they were proposed to replace the GMMs for the estimation of the state-posteriors for HMMs , *i.e.* $p(\boldsymbol{s}|\boldsymbol{o})$. To compute a Viterbi alignment or to run the forward-backward decoding algorithm within the HMM framework, we need to convert posteriors to likelihoods $p(\boldsymbol{o}, \boldsymbol{s})$ by dividing the frequencies of each HMM states, $p(\boldsymbol{s})$ [15]. It has been shown in the literature that this conversion is not crucial to many recognition tasks; however, it may be important when the target classes are highly unbalanced.

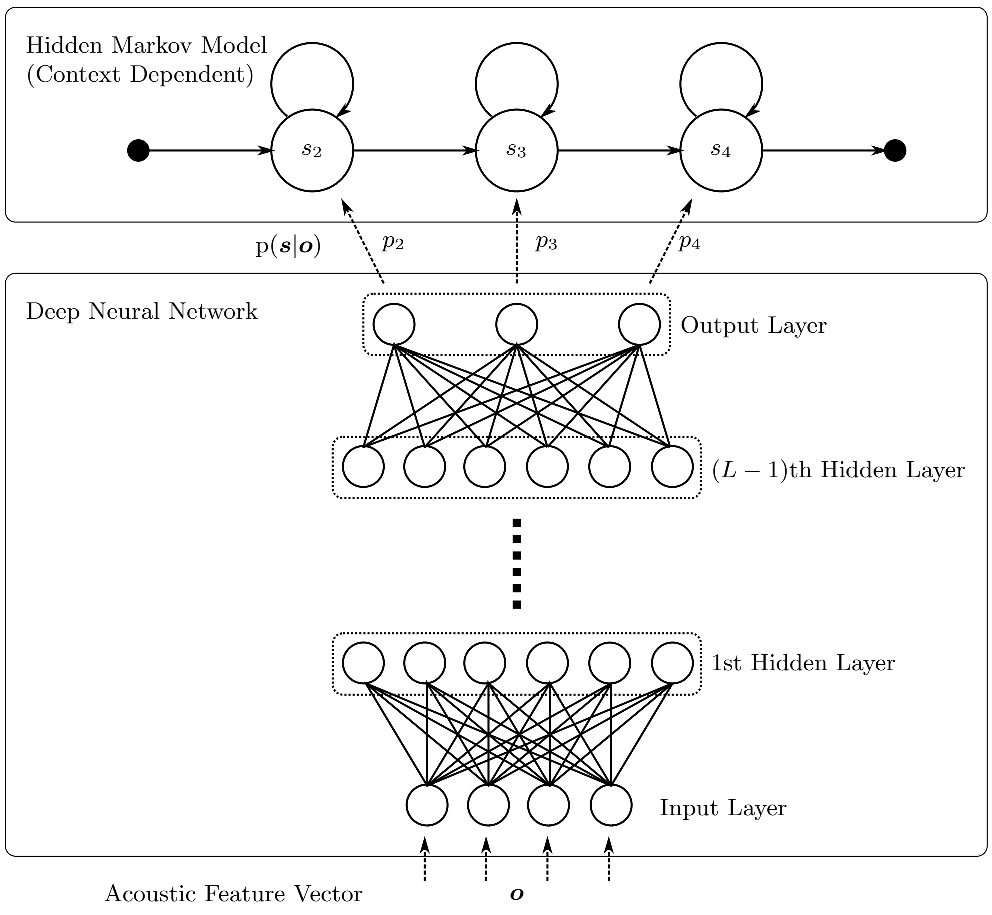


Figure 3.5: The hybrid DNN-HMM system architecture.

Due to the difficulty in training deep NNs before 2006, the hybrid NN-HMM is only used to predict context independent phoneme states. The dimension of these posteriors are usually around one hundred. With the development of deep learning algorithms, DNNs have been adopted to replace the shallow NNs in the hybrid NN-HMM system for speech recognition since 2009 [8], which will be referred to as the DNN-HMM model. After obtaining promising performance on the benchmark TIMIT phoneme recognition task, DNNs are used to predict posteriors for thousands of context dependent phoneme states in the conventional large vocabulary speech recognition systems. Dramatic improvements over the complicatedly engineered GMM-HMM systems have been reported by many research groups [23], which clearly demonstrates the effectiveness of DNNs for speech recognition. This is quickly making DNN the new standard for speech recognition systems, which is the main subject of our investigation in this work. For future references, this standard context dependent hybrid DNN-HMM system is illustrated in Figure 3.5.

Table 3.1: WER(%) performance of the multi-style GMM and DNN on Aurora-2.

Test Set	Condition	GMM	DNN
Clean		0.6	0.4
A	seen noise	12.3	4.6
B	unseen noise	10.4	5.3
C	seen & unseen noise + channel	17.9	5.1

3.2 DNN AM's Noise Robustness

Although DNNs have already largely outperformed GMMs on various ASR tasks, their performance in real world scenarios is still far from humans' expectations. In [23], the hybrid DNN-HMM system reduced the Word Error Rate (WER) of the GMM-HMM system from 52.3% to 47.6% on a 1400-hour YouTube transcription task. Despite the relative 9% WER reduction, the system is still making mistakes almost half of the time. Further investigation of the problem reveals that the training and testing mismatch is the major reason for the limited performance of the DNNs. These mismatches come from a large number of potential variations in the data, such as acoustic environments, sound reverberations, external noise sources, communication channels, speaker characteristics, language characteristics and etc. Hence to improve DNNs performance for real world applications, these mismatch problems must be addressed. In our study, we mainly focus on two of the above mentioned variations, namely the additive noise and the channel distortions.

To validate that these two factors do degrade the DNN AM's performance, we evaluate it on two benchmark noisy speech recognition tasks: the noisy digit speech recognition task, Aurora-2 [58], and the medium vocabulary noisy speech recognition task, Aurora-4 [117]. Details about these two corpora will be discussed in Section 6.1.1 and 6.1.2 respectively. In these corpora, the additive noise and the channel distortions are added manually to simulate the real world noisy speech. This artificial noisy speech could help us rule out other factors and focus specifically on a particular noise variation. The DNN AM investigated here is trained from the multi-style training data, which could give the model a sense of noisy speech. Depending on each specific test scenario, the noise may be different and we can study the generalization capabilities of DNNs across different noise types.

The recognition performance for these two tasks are tabulated in Table 3.1 and Table 3.2. Detailed experimental setups could be found in Section 6.2.1. From these two tables, DNNs consistently outperform GMMs by a lot. On Aurora-2, the multi-style trained DNN AM has a performance of 0.4% WER on clean data; while for the same set of noise conditions, the testing performance is already far away (4.6% *vs.* 0.4%). On noise types that have not been encountered during the DNN's training, further

Table 3.2: WER(%) performance of the multi-style GMM and DNN on Aurora-4.

Test Set	Condition	GMM	DNN
A	clean	8.7	5.0
B	noise	16.6	8.8
C	channel	18.8	9.0
D	noise + channel	31.9	20.1

degradation occurs (5.3% *vs.* 4.6%). Similarly for the additional channel distortions, it degrades further (4.6% *vs.* 4.6%). From these experiments, even if the DNN is trained on the matched noisy data, its performance is far from what could be obtained for clean speech. On this particular dataset, it performs more than 10 times worse on noisy speech than on clean speech, not mentioning the further degradation brought by unseen noise and channel distortions.

On the Aurora-4 dataset, the training and testing has the same set of noise types and channel distortions. No mismatches exist. Comparing set B and C with set A, clear degradation is observable when either additive noise or channel distortions exist and channel distortions have slightly worse performance than additive noise. Furthermore, when both these two types of variations exist in the speech, the WER increases up to about 4 times of it in the case of clean speech.

All these studies suggest that the two noise factors, additive noise and channel distortions, do degrade the DNN AM's performance dramatically. Recognizing the matched noisy speech is already a challenging task for DNNs. With mismatched noise conditions, they perform even worse and the problem is even challenging. The large performance differences between clean and matched noisy speech may also imply that simply collecting more and more noisy speech is not guaranteed to improve DNNs' performance on noisy speech to reach the level of performance in the clean-like case. Techniques specifically addressing the noise variations are crucial to DNNs robustness on noisy speech.

We start with investigations of existing feature-based noise-robust techniques. As they only modify features and have no back-end model requirements, we could directly apply them to our DNN AMs. Two broad categories of techniques are explored. They are feature normalization approaches and feature enhancement techniques.

3.2.1 Conventional Noise-Robust Features

Feature normalization algorithms aim at removing noise corruptions in speech features extracted from noisy speech. Various normalization methods and feature extraction processes refined with those normalization methods have been shown to be effective for the GMM-HMM AMs in the literature. To further understand the DNN's noise

Table 3.3: WER (%) performance of different robust feature extraction methods in both GMM-HMM and DNN-HMM systems on Aurora-2.

System			Test Set			Avg.
			A	B	C	
GMM	MFCC	CMS	7.29	**7.45**	6.86	7.27
		CMVN	6.87	7.50	7.31	7.21
		CSN	6.35	6.97	6.53	6.63
		HEQ	6.91	7.32	7.05	7.10
		MVA	6.27	6.95	**6.52**	6.59
		AFE	**5.86**	6.65	7.06	**6.42**
DNN	MFCC	CMS	5.80	6.77	5.78	6.18
		CMVN	5.35	**6.42**	**5.72**	**5.85**
		CSN	5.27	6.67	5.81	5.94
		HEQ	5.62	6.62	5.95	6.09
		MVA	5.24	6.66	5.84	5.93
		AFE	**5.06**	6.50	7.15	6.05
DNN	FBank	MVN	**4.55**	**5.68**	**5.62**	**5.22**

robustness, we evaluate the following techniques for the DNN-HMM AMs: Cepstral Mean Subtraction (CMS), Cepstral Mean and Variance Normalization (CMVN), Cepstral Sub-band Normalization (CSN), Histogram EQualization (HEQ), Mean subtraction Variance normalization with Autoregressive-moving-average filtering (MVA) and Advanced Front-End (AFE). The comparison is illustrated in Table 3.3. For GMM AMs, those complex engineered features do reduce WERs; however, for DNN AMs, the simple CMVN is sufficient. The CMVN processing is actually a necessary feature processing step that is already required by the DNN itself. This is because for our first layer GRBM, we restricted all the visible units to have unit variances. These results may suggest that with DNNs' superior variation modeling capability, some feature engineering steps are not necessary. Furthermore, if we use FBank features rather than the MFCCs together with Mean Variance Mormalization (MVN), we could get even better performance. This further confirms that for DNNs, many feature engineering techniques developed for the GMMs with limited modeling capacities are not necessary and sometimes they even harm the DNNs' performance, as any feature extraction corrupts the original signals.

3.2.2 Speech Enhancement Techniques

Instead of those normalization-based noise reduction techniques, various algorithms using the speech and noise interaction model to estimate the clean speech features from the noisy ones have been successfully developed in the literature. These are usually referred to as feature-based speech enhancement techniques. Many existing

Table 3.4: WER (%) performance of different feature enhancement algorithms for the clean-data trained AMs on Aurora-2.

AM	SNR	Baseline	MMSE	MLSA	MAPA	GMAPA
GMM	clean[127]	0.36	0.39	0.34	0.36	**0.33**
	Avg.[127]	40.56	31.72	36.88	31.91	**29.14**
DNN	clean	**0.32**	0.60	0.53	0.64	0.94
	20dB	1.53	1.47	**1.38**	1.49	1.81
	15dB	3.15	3.11	**2.93**	3.12	3.58
	10dB	8.25	8.31	**7.91**	8.32	8.71
	5dB	20.29	20.60	**19.33**	20.64	19.83
	0dB	43.55	43.94	**41.92**	44.13	42.44
	Avg.	15.35	15.48	**14.46**	15.54	15.28

GMM-HMM ASR systems employ enhancement schemes as a pre-processor to improve the speech quality. Generally speaking, speech enhancement algorithms can be grouped into three categories, namely filtering, spectral restoration, and speech model techniques [118]. We mainly investigate the spectral restoration approach for DNN AMs, which estimates a gain function to perform the noise reduction in the frequency domain. Specifically, the Minimum Mean Square Error spectral estimator (MMSE) [119, 120, 121, 122], Maximum A Posterior spectral Amplitude estimator (MAPA) [118, 123, 124], Maximum Likelihood Spectral Amplitude estimator (MLSA) [118, 125, 126] and the Generalized MAPA (GMAPA) [127] are justified. The performance of those techniques on the clean-trained DNN is tabulated in Table 3.4 and on the multi-style trained one is illustrated in Figure 3.6.

When AMs are trained on the clean speech, all the enhancement methods have large improvements on the GMM; however, for the DNN, only the MLSA is effective in obtaining clear gains. Methods like MMSE and MAPA perform even worse than the baseline system without any enhancement. It may be due to the fact that DNNs are capable enough to reduce noise variations through their multiple layered representation learning. When the features are enhanced in an imperfect way, the errors incorporated may override the gain the enhancement brings to a powerful model. Furthermore, if the DNN AM is trained on multi-style data, the gain from the MLSA enhancement also diminishes when more hidden layers are used (Figure 3.6).

3.3 A Representation Learning Framework

From those initial investigations, DNNs are good at modeling the training data variations. With multi-style data, DNNs could dramatically improve the performance on noisy speech; but the performance is still several times higher than what could be ob-

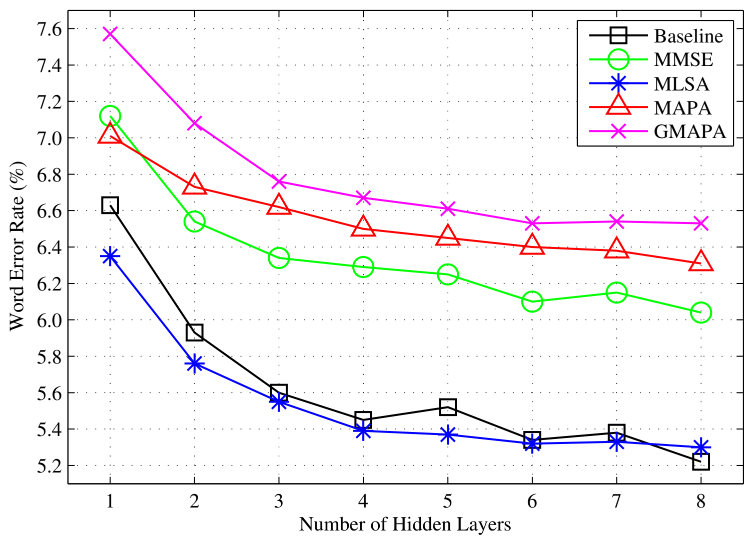


Figure 3.6: Effectiveness of spectral restoration techniques on multi-style trained DNNs on Aurora-2.

tained on clean speech. Addressing the noise corruption directly rather than simply adding more training data is still compulsory. Conventional feature-based approaches such as normalization and enhancement techniques lose their effectiveness on DNNs and sometimes they even hurt DNNs' performance. While for model-based techniques, they all assume the Gaussian-based AMs and are difficult to be ported to DNNs. Furthermore, due to DNN's multiple layers' nonlinearities, integrating the environment model (equation (2.3) or (2.4)) into DNNs will lead to a complex optimization problem without closed-form solutions.

In this thesis, instead of treating the DNN as a black-box model that converts input features to output posteriors, we interpret it as a stack of nonlinear feature transformation layers, each of which converts its input feature into a higher level abstract representation that has less noise variations and better discrimination. These transformation layers are sequentially integrated and jointly estimated to optimize the output predictions towards the supervision labels. Each level of the transform layers captures different aspects of the data and collectively they make the DNN a much better model in representing the data. Unlike the conventional feature engineering process, the human designed features focus only on representing the data and are independent of the ultimate tasks. For this representation learning in DNNs, we have the unsupervised stage of RBM pre-training to understand the data characteristics and moreover, we also have the second stage of supervised fine-tuning that further adjusts the transformations to yield superior classification performance for the final classification layer. With these two stages of learning, the lower layers commonly learn more local information and the higher layers capture the abstract structures. Similar observations have been made in vision research. An example of different levels of representations learned by DNNs for the face recognition problem is depicted in [128]. At the lowest level, the DNN takes in pixels and first learns to detect edges from them. With different edge patterns, another level of object part representations is captured, such as eyes, noses etc. for face recognition. Finally, the DNN combines those parts to represent different faces.

3.3.1 Layered Representation Learning in DNN AM

In the signal processing literature, various speech representations have been developed. Most of them are based on our limited understanding of the human speech perception process. In the conventional ASR systems, they are mainly the time-domain waveform, the frequency-domain spectral feature (including the power spectrum and the FBank) and the cepstral feature representations. With the fast adoption of DNNs in ASR systems, many researchers have started questioning whether MFCCs are the best features for DNNs. Many results have shown that FBanks usually yield better performance than MFCCs [21, 129, 130, 131]. One probable reason is that with the improved modeling capability of DNNs, relatively raw representations with less information loss are more preferable. There are also some other interesting work [132] that tries to abandon all the signal processing and build models directly on the time-domain waveform signals. Although moderate performance has been achieved, they usually involve a rather complex model training compared to the conventional processing. To the knowledge of the author, before we obtain a thorough understanding of how we humans process the large amount of raw information, it would not be viable for computers to deal with those raw information by simply collecting more data and increasing the model size. With improved machine learning techniques and statistical models, we may need to discard some of the human-designed feature engineering steps and let the model automatically learn them from the data. In the literature, Convolutional Neural Network (CNN) has been developed for many applications with the emphasis on learning invariant features [133, 134, 135, 136]. It uses local filtering and pooling techniques to obtain translation invariant internal feature representations. Ultimately, it could be treated as a DNN with special model structures. In this study, we mainly focus on the noise robustness of the basic DNNs. The features we adopted as DNNs' inputs are spectral domain FBank features rather than the commonly used MFCC features in GMM based systems.

From the perspective of representation learning, an L-layer DNN AM could be summarized as carrying out the following series of representation transformations on top of the human engineered spectral features:

- The input spectral feature representation: $\boldsymbol{o}_t$

- The normalized feature representation: $\boldsymbol{o}'_t$

$$\boldsymbol{o}'_t = f_{\texttt{MVN}}(\boldsymbol{o}_t) = \boldsymbol{\Sigma}^{-\frac{1}{2}}(\boldsymbol{o}_t - \boldsymbol{\mu}) \tag{3.28}$$

- The context expanded representation with the window length of $(2w+1)$: $\boldsymbol{h}_{0,t}$

$$\boldsymbol{h}_{0,t} = \begin{bmatrix} \boldsymbol{o'}^{\top}_{t-w} & \cdots & \boldsymbol{o'}^{\top}_{t} & \cdots & \boldsymbol{o'}^{\top}_{t+w} \end{bmatrix}^{\top} \tag{3.29}$$

- The hidden-activation representation: $\boldsymbol{h}_{l,t}$

$$\boldsymbol{h}_{l,t} = \phi(\boldsymbol{W}_l\, \boldsymbol{h}_{l-1,t} + \boldsymbol{b}_l), \quad \text{for} \quad 1 \leqslant l < L \tag{3.30}$$

- The output posterior representation: $\boldsymbol{p}_t$

$$\boldsymbol{p}_t = \psi(\boldsymbol{W}_L\, \boldsymbol{h}_{L-1,t} + \boldsymbol{b}_L) \tag{3.31}$$

The subscript t is the frame index for each feature vector in the speech feature sequences. $(\boldsymbol{\Sigma}, \boldsymbol{\mu})$ are the MVN parameters where the diagonal covariance is commonly used. $(\boldsymbol{W}_l, \boldsymbol{b}_l)$ are the transformation parameters for the lth layer in the L-layer DNN.

3.3.2 Noise Robustness in Different Representations

With all these different representations involved in the DNN AMs, understanding how the noise factors affect them is important to finding effective solutions. However, due to the different generation processes involved in each of the representations, they vary greatly both in dimensions, ranges and magnitudes. All these increase the difficulty of obtaining a consistent evaluation measure to justify their reactions to noise factors. Instead of using a numerical measurement, we visualize these different representations of a speech utterance under both the clean and noisy conditions in Figure 3.7. Intuitively, the noise corruption largely decreases the discrimination information of those linguistic units. The background energy is increased and the speech discrimination is decreased. The structure information for those linguistic units becomes less distinguishable.

For the human engineered representations, including the waveform, the power spectrum (129D) and the FBank (24D), the noise corruptions are more observable. While for the 1024D hidden representations learned by DNNs, namely the H1, H2, H3 and H4 activations, the noise effects are less intuitive. One major reason for this is the arrangement of the hidden feature dimensions. Although the ordering of those hidden feature dimensions has no effect on the DNN model, it is crucial to human eyes, as humans are more efficient in handling well structured patterns. A good topic to study

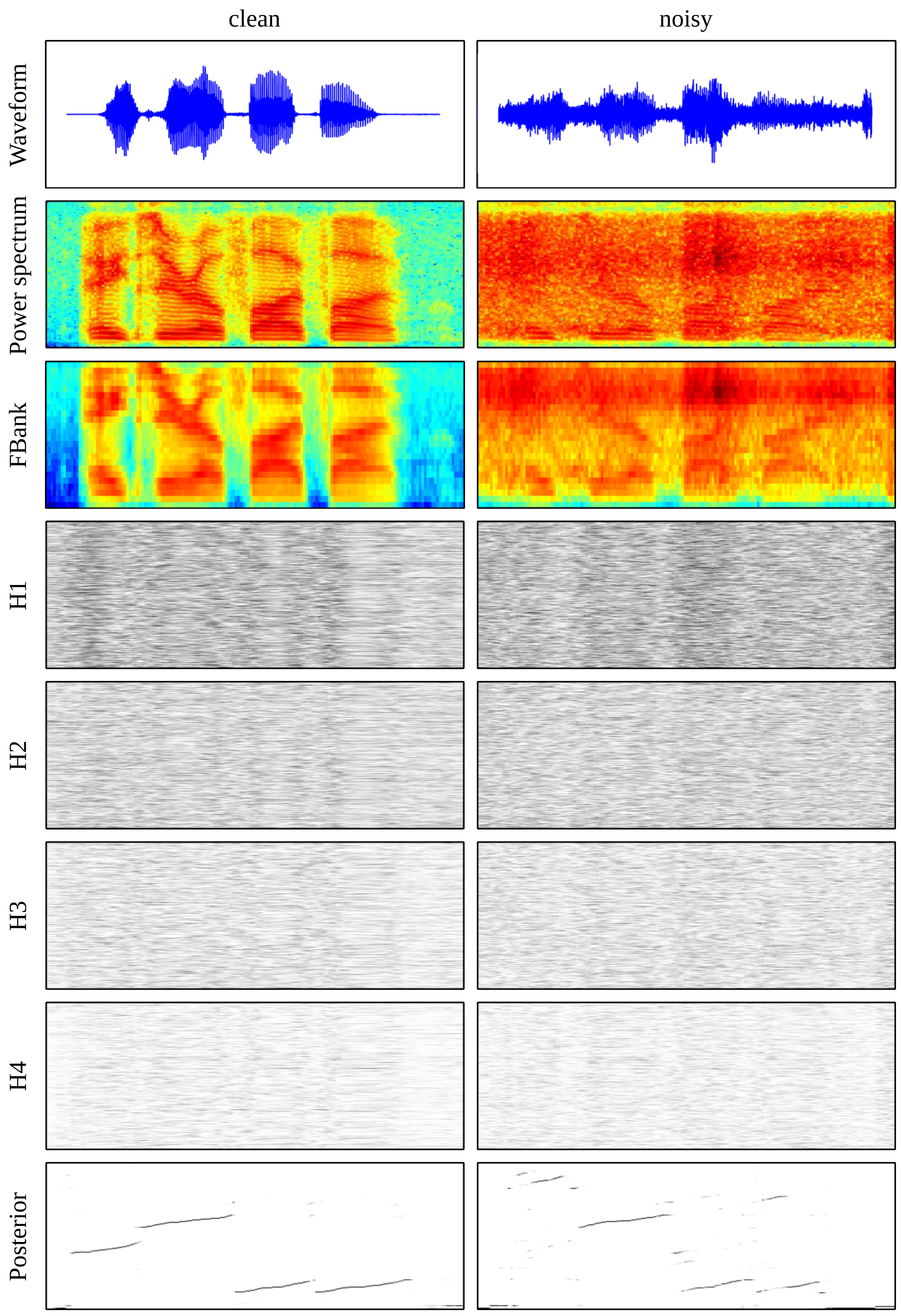


Figure 3.7: Different representations of the utterance “8055” under clean and noisy (train noise with 0dB SNR) conditions.

may be how to better visualize the hidden representations. As this thesis focuses on improving machine learning models rather than human understandings, we will not explore in that direction. Although it is hard to find detailed differences between the clean and noisy hidden representations, one general observation is that the noisy hidden activations are more dense. It means there are many hidden units that are inactive for the clean speech now become activated due to noise. Finally, for the 181D posterior representation, which is generated from the output layer of the DNN and represents the probability of the input feature vector belongs to each state of the whole word HMMs, the four digits are clearly observable for the clean speech. For the noisy speech, we could clearly see that the first digit and the initial part of the fourth digit are incorrectly recognized. Moreover, the high sparsity in the posteriors further confirms the sharpness of the posteriors generated by DNNs. This sharpness means that DNNs are very confident in their predictions even if they are wrong. It is good when the predictions are correct; but if wrong, they are difficult to correct, unless there is an even stronger language model constraint.

Simply speaking, from this intuitive study of various representations involved in the DNN AMs for speech recognition, we have found the existence of noise variations through out the hierarchy. To address the noise robustness of DNNs, improving the noise robustness of these representations is a promising direction.

3.3.3 Learning Robust Representations for DNN

The performance of many machine learning methods is heavily dependent on the choice of data representation on which they are applied. For example [137], in arithmetics, addition is much simpler than multiplication in our conventional decimal representation of numbers. However, if we use the set of prime factors to represent each number, multiplication would be simplified to a union of two sets. But addition would become much more difficult. For that reason, much of the actual effort in deploying machine learning algorithms goes into the design of preprocessing pipelines that result in a hand-crafted representation of the data that can support effective machine learning. Such feature engineering is important but labor-intensive and highlights the weakness of many traditional learning algorithms: their inability to extract and organize the discriminative information from the data. Feature engineering is a way to take advantage of human ingenuity and prior knowledge to compensate for that weakness. In order to expand the scope and ease the applicability of machine learning, it would be highly desirable to make learning algorithms less dependent on feature engineering [138, 139]. Deep learning algorithms such as DNN are exactly the kind of representation learning procedure that discovers multiple levels of representations, with higher-level features representing more abstract aspects of the data. More importantly, it usually employs shallow (single-layer) representation learning as subroutines.

In speech recognition, we have found that the multiple levels of representations learned by DNNs could dramatically improve the ASR performance. In addition, we have also observed that there is still space for improvement in those representations, especially under noisy conditions. Hence in this thesis, we will work towards the same objective, that is, to obtain better representations for improved ASR performance from a slightly different perspective. Specifically, we want to further refine the representations to be invariant to noise corruption. The main reason for this comes from our observation that existing representations learned by DNNs are capable of yielding very good performance when used on clean speech. But when the noise factor is introduced into speech signals, dramatic degradation occurs. This may suggest that the existing representations are already good enough to capture the linguistic discriminations in the speech signal and what they lack is the robustness to additional noise factors. Hence exploring techniques to improve the noise-robustness of the various representations discussed in Section 3.3.1 would be promising.

Based on the types of representations, we group them into two categories: namely the input representations which are independent of the DNN's layer parameters and the hidden representations that are computed using the DNN's layer parameters. The input representations include the input feature representation $\boldsymbol{o}_t$, the normalized representation $\boldsymbol{o}'_t$ and the context expanded representation $\boldsymbol{h}_{0,t}$. Techniques specific to improve the input representations' noise robustness are presented in Chapter 4. The hidden representations are the hidden activations from those $L-1$ hidden layers of the DNN AM. They are further refined using algorithms developed in Chapter 5. The output-posterior representation $\boldsymbol{p}_t$ is only used for the final decision making. Although it is also possible to address the mismatches caused by noise in the output-posterior representations [140], the gains are relatively small due to the difficulty incurred by their sharpness. Hence, in this study, the output-posterior representation is not investigated.

3.4 Summary

In this chapter, the DNN model and how its usage in acoustic modeling in ASRs are reviewed. Despite the significant improvements obtained by DNNs over the conventional GMMs, we have observed that two of the variation factors, namely additive noise and channel distortion, dramatically degrade the DNN's performance. By further investigating the conventional noise robustness techniques, few gains can be obtained for DNN AMs. The superior variation modeling capabilities in DNNs yield improved performance, and at the same time also increase the difficulty of making further improvements. By studying the effects of noise on different levels of representations, we find that there are still noise variations throughout the representation hierarchy. We hence propose to address the issue of the DNN AM's noise robustness by reducing noise variations in different representations.

Chapter 4

Noise-Robust Input Representation Learning

In this chapter, we focus on improving the noise robustness of the input feature representations for the DNN AM. We start with an investigation into the reliability of the mean and variance normalization parameters for generating the normalized feature representations (equation (3.28)). By treating these parameters as a single Gaussian front-end of DNNs, the conventional GMM-based VTS model compensation technique becomes applicable. A VTS-based feature normalization technique is hence developed. Following that, we investigate the possibility of using wider input features for context expanded representation (equation (3.29)). It is inspired by the fact that human auditory periphery has a short-term memory of the order about 200ms, which is much longer than the usual 25ms-long speech frames used for ASRs. Meanwhile, the DNN is superior in modeling variations. With this long span of speech inputs, it may be capable of identifying a better environment estimation by itself, which would probably further improve recognition performance. This leads us to develop the Deep Split Temporal Context (DSTC) modeling technique. With these two techniques, clear performance improvements have been observed on the Aurora-2 task. However, with more training data, such as on the Aurora-4 task, the gain is relatively small. To further address the noise variations in the input features explicitly, a DNN-based spectral masking approach that mimics human speech perception's "separation-prior-to-recognition" process is developed. It removes noise-dominant time-frequency components in the power spectrum used for the FBank feature extraction. This technique, to our knowledge, has yielded the best reported recognition performance on both the Aurora-2 and the Aurora-4 tasks as the time of writing.

4.1 VTS-based Feature Normalization

Based on the initial investigations presented in Section 3.2, DNNs have demonstrated superior capabilities in modeling acoustic variations over conventional GMMs. Many robust feature extraction algorithms and speech enhancement techniques successfully developed for GMMs seem automatically learned by DNNs from multi-style data. To further improve their robustness against noise, we need techniques that are more targeted for DNNs.

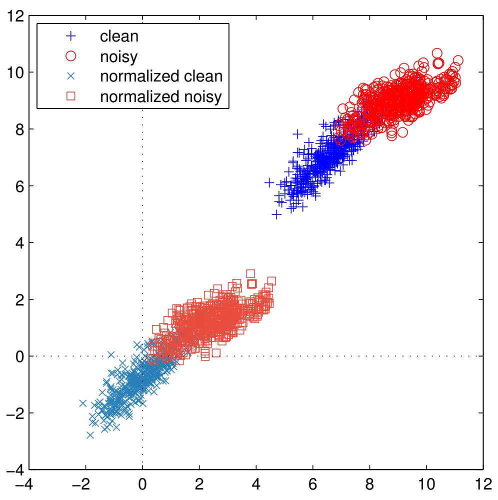
In [141], the Recurrent Neural Network (RNN) has been shown to generalize much better than GMMs and MLPs on the Aurora-2 task [58]. It has similar performance to DNNs but requires much more training computation due to its increased complexity. In [142], a Deep Recurrent Denoising AutoEncoder (DRDAE) is trained on the stereo data to reconstruct clean utterances from noisy input features. It has been shown to outperform the SPLICE denoising algorithm [143] and the hand-engineered AFE denoising system [144]. The DRDAE makes no assumption on how the noise affects the signal, or the existence of distinct noise environment. It is thus more dependent upon the training data to provide a reasonable sample of noise environments that could possibly be encountered at test time.

Model-based approaches, utilizing explicit models of noise and channel distortions and their interactions with speech, are a well-established and continually evolving research paradigm in noise-robust speech recognition. The VTS compensation method [145] and the corresponding Noise Adaptive Training (NAT) [146] have been widely adopted in Gaussian-based GMM-HMM systems. Due to the many layers of nonlinearities, and non-Gaussian-based formulation, deriving an analytical approach to directly compensate the DNN model is much more difficult. In [147], a Factorial Hidden Restricted Boltzmann Machine (FHRBM) is proposed to explicitly model noise distribution, and how such noise affects speech. However, because of un-observed noise parameters in the input layer of the FHRBM, the inference is intractable, and scales exponentially with the number of hidden units. Variational approximations have to be used. Additionally, even in the preliminary experiments reported in [147], the FHRBM did not achieve the best performance.

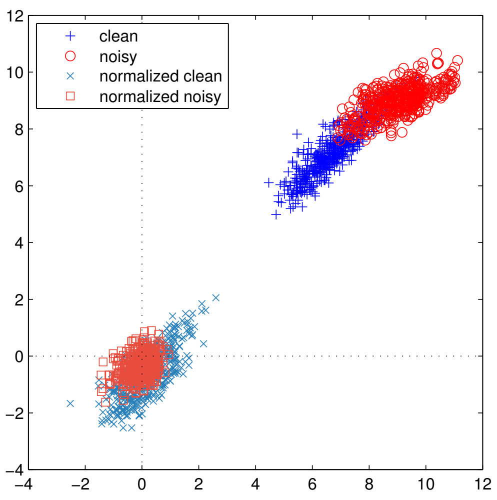
In this section, we tackle the noise robustness problem through learning a more reliable normalized feature representation for the DNN AM. This is achieved by treating the normalization process as a single Gaussian front-end, and applying conventional VTS model compensation to it. The compensation takes in both the Gaussian front-end and an estimation of the current environment, and generates another Gaussian front-end that better represents current noise conditions. With this new Gaussian front-end, normalization is more reliable and the better-normalized representation would then lead to improved recognition performance. To fully benefit from the powerful modeling capability of the DNN and avoid potential mismatches brought by different normalizations, an adaptive training algorithm has been further developed.

4.1.1 Feature Normalization

Features sent to NNs are commonly required to have similar dynamic ranges. It avoids the undesirable problem of imbalance caused by different feature magnitudes. To decrease the additional computation incurred, a simple MVN (equation (3.28)) has become a standard processing step for NNs. Moreover, specifically for DNNs, due to the use of the unit-variance GRBM (equation (3.25)) for the initialization of the input layer, this MVN process is even more important. Conventionally, two types of MVNs are adopted, which are the global MVN and the utterance-based MVN. The global MVN estimates the mean and variance normalization parameters of the entire training data. It assumes that the training and testing data come from the same underlying data distribution, and we can hence use the training estimation for testing data normalization. However, in practice, due to various factors, this assumption is hard to guarantee. The other one, *i.e.* the utterance-based MVN, estimates the parameters from each individual test utterance. This makes no assumption that the training and testing data must come from the same data distribution. But it may have the problem of being unreliable due to having limited data available for parameter estimations.



(a) Global MVN.



(b) Utterance-based MVN.

Figure 4.1: A comparison between the two MVNs using only the first two dimensions of FBank features on Aurora-2.

A comparison between these two MVN approaches is carried out on the Aurora-2 corpus. For an intuitive understanding, we illustrate the data distribution using only the first two dimensions of the FBank feature vectors. The results are depicted in

Figure 4.1. The original data distributions of the clean training and noisy testing speech are depicted with blue plus signs and red circles on the top right part of each figure. A clear distribution mismatch is observable. Using the global MVN (Figure 4.1a), the clean training and noisy testing data are mainly shifted; but the corresponding distributions remain mismatched. With the utterance-based MVN (Figure 4.1b), the training and testing data are both shifted and scaled to have a better match. The corresponding distributions overlap with each other after normalization. It suggests that the mismatch between the clean training and noisy testing data is reduced through the normalization process. If the underlying distribution of training and testing data is the Gaussian distribution with just different mean and variance parameters, this utterance-based MVN would normalize both of them to be the normal distribution and hence remove the mismatch. However, for real speech signals, this Gaussian assumption is hard to maintain. It hence only has limited capability in reducing the distribution mismatches.

Besides the functionality differences between these two MVN processes, they could also incur different time latencies for real time ASR systems. For the global MVN, there is no MVN estimation latency as the parameters are computed a priori and repeatedly used for every testing utterance. However, for the utterance-based MVN, the normalization mean and variance are unavailable before the utterance finishes. The recognition process can only start after the whole utterance is spoken and the MVN statistics are ready. This additional MVN estimation latency is unavoidable for the utterance-based MVN. A simple way to address this latency is to incrementally update the MVN parameters. However, the invalid Gaussian distribution of the data and the limited samples used to update the MVN statistics would limit its effectiveness.

From the above comparisons, both the commonly adopted MVNs have their pros and cons. An MVN that has the same reliability as the global MVN, and also the mismatch reduction capability of the utterance-based MVN, is more desirable. We hence propose using the Vector Taylor Series - Mean and Variance Normalization (VTS-MVN), which utilizes the global MVN to gain prior information for improved reliability, and incorporates the current environment information through the VTS compensation to improve efficacy. This process is similar to the model-based VTS compensation for the GMM-HMM system. The only difference lies in how many Gaussians are involved. The conventional VTS compensation is applied to all the Gaussians in the GMMs for each HMM state, while for our DNNs, the compensation occurs in the single Gaussian normalization front-end. As only one Gaussian is involved, it would be more efficient than the conventional method.

4.1.2 VTS Model Compensation

The model compensation scheme combines the clean-trained model and noise distributions with the mismatch function to find the parameters for the noise-corrupted speech model. The clean model in our VTS-MVN process is the single Gaussian with mean and variance estimated from the whole training data (equation (3.28)). We borrow the noise estimations from the conventional VTS compensation process. The formulation of the speech and noise interaction model, *i.e.* the noise environment model, is discussed in Section 2.1. The mismatch function for the static features in the cepstral feature domain is derived in equation (2.4), which is re-stated here for easy reference:

$$\boldsymbol{y}^{(s)} = \boldsymbol{x}^{(s)} + \boldsymbol{u}^{(s)} + \boldsymbol{C} \log\Big(\boldsymbol{1} + \exp\big(\boldsymbol{C}^{\dagger}(\boldsymbol{z}^{(s)} - \boldsymbol{x}^{(s)} - \boldsymbol{u}^{(s)})\big)\Big) \tag{4.1}$$

where the superscript (`MFCC`) for the feature domain in the original equation is replaced with (s) to distinguish the static coefficients from the dynamic ones (Δ) and (Δ^2) in the following equations. The $\boldsymbol{y}^{(s)}, \boldsymbol{x}^{(s)}, \boldsymbol{u}^{(s)}, \boldsymbol{z}^{(s)}$ are the static coefficients of the cepstral feature vectors corresponding to the distorted speech, clean speech, channel and additive noise, respectively. $\boldsymbol{C}$ and $\boldsymbol{C}^{\dagger}$ are the discrete cosine transform and its pseudo-inverse. For DNNs, the FBank features are more favorable than the MFCC features, so we would like to apply the VTS compensation in the FBank feature domain. However, due to the dependencies among FBank feature dimensions, the underlying GMMs have to adopt full covariances to achieve comparable performance as the MFCC features, which will greatly increase the amount of training data and computations. As the conversion from the FBank spectral features to the MFCC cepstral features is simply a linear transformation using the DCT transform $\boldsymbol{C}$, we hence, in this study, first use the conventional cepstral feature based GMM-HMM systems for VTS compensation. After obtaining the noisy model parameters in the cepstral domain, we simply reverted them back to the FBank spectral domain by multiplying them with the pseudo-inverse DCT transform $\boldsymbol{C}^{\dagger}$. Due to the non-linearity of the mismatch function, it is hard to directly incorporate equation (4.1) into the ASR systems. The first-order VTS approximation [145] of it could be expressed as

$$\boldsymbol{y}^{(s)}_{\text{vts}} = \boldsymbol{y}^{(s)}|_{\mu_0^{(s)}} + \boldsymbol{J}_x(\boldsymbol{x}^{(s)} - \boldsymbol{\mu}_x^{(s)}) + \boldsymbol{J}_z(\boldsymbol{z}^{(s)} - \boldsymbol{\mu}_z^{(s)}) + \boldsymbol{J}_u(\boldsymbol{u}^{(s)} - \boldsymbol{\mu}_u^{(s)}) \tag{4.2}$$

where $|_{\mu_0^{(s)}}$ indicates an evaluation of equation (4.1) at the Taylor series expansion point $\mu_0^{(s)} = \{\boldsymbol{\mu}_x^{(s)}, \boldsymbol{\mu}_z^{(s)}, \boldsymbol{\mu}_u^{(s)}\}$ with the static speech component mean $\boldsymbol{\mu}_x^{(s)}$, the static additive noise mean $\boldsymbol{\mu}_z^{(s)}$ and the static channel noise $\boldsymbol{\mu}_u^{(s)}$. The Gaussian component index m in the original equation (2.13) is omitted as there is only one Gaussian for our DNN normalization front-end. The Jacobian matrices could be computed similarly using equations (2.14), (2.15), (2.16).

Computing the mean of the noisy speech, *i.e.* the expected value of equation (4.2), is straightforward, since it is a linear function of three vectors: the additive noise, the clean speech, and the channel noise. This may be expressed as

$$\begin{aligned}\boldsymbol{\mu}_y^{(s)} &= \mathcal{E}\{\boldsymbol{y}^{(s)}\} \approx \mathcal{E}\{\boldsymbol{y}_{\text{vts}}^{(s)}\} = \boldsymbol{y}^{(s)}\big|_{\mu_0^{(s)}} \\ &= \boldsymbol{\mu}_x^{(s)} + \boldsymbol{\mu}_u^{(s)} + \boldsymbol{C}\log\Big(\mathbf{1} + \exp\big(\boldsymbol{C}^{-1}(\boldsymbol{\mu}_z^{(s)} - \boldsymbol{\mu}_x^{(s)} - \boldsymbol{\mu}_u^{(s)})\big)\Big). \qquad (4.3)\end{aligned}$$

The covariance of the linear corrupted speech function is simply the sum of the transformed covariances of the clean speech, additive noise and channel

$$\begin{aligned}\boldsymbol{\Sigma}_{y,\text{full}}^{(s)} = \mathcal{E}\{\boldsymbol{y}^{(s)}\,\boldsymbol{y}^{(s)\top}\} - \boldsymbol{\mu}_y^{(s)}\,\boldsymbol{\mu}_y^{(s)\top} &\approx \mathcal{E}\{\boldsymbol{y}_{\text{vts}}^{(s)}\,\boldsymbol{y}_{\text{vts}}^{(s)\top}\} - \boldsymbol{\mu}_y^{(s)}\boldsymbol{\mu}_y^{(s)\top} \\ &\approx \boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(s)}\boldsymbol{J}_x^\top + \boldsymbol{J}_z\boldsymbol{\Sigma}_z^{(s)}\boldsymbol{J}_z^\top + \boldsymbol{J}_u\boldsymbol{\Sigma}_u^{(s)}\boldsymbol{J}_u^\top \qquad (4.4)\end{aligned}$$

assuming the clean speech, additive noise and channel noise are independent of each other. The term $\boldsymbol{\Sigma}_z^{(s)}$ denotes the variance of the static additive noise and $\boldsymbol{\Sigma}_u^{(s)}$ the variance of the static channel noise. Since the Jacobian matrices $\boldsymbol{J}_x$, $\boldsymbol{J}_z$ and $\boldsymbol{J}_u$ are full, the corrupted speech covariance matrix will also be full and is usually diagonalized for standard decoders. It is often assumed that the channel noise does not change in each particular utterance. Hence, we have $\boldsymbol{\Sigma}_u^{(s)} = 0$. The static corrupted speech variance could finally be computed by

$$\boldsymbol{\Sigma}_y^{(s)} \approx \operatorname{diag}\Big(\boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(s)}\boldsymbol{J}_x^\top + \boldsymbol{J}_z\boldsymbol{\Sigma}_z^{(s)}\boldsymbol{J}_z^\top\Big). \qquad (4.5)$$

These update formulas assume that the noise-corrupted clean speech Gaussian component may be approximated by another Gaussian distribution. This is clearly not optimal, since the corrupted speech distribution can be bimodal. Nevertheless, for efficiency, this approximation is often maintained.

Standard acoustic models use simple differences or linear regression to compute delta parameters to model the dynamic features of speech. This complicates the compensation of these features for noisy conditions, for example, making it difficult to apply the log-normal approximation to compensate the dynamic covariance matrices. A continuous time approximation [33] is often used to derive the compensated dynamic parameters. The final compensation formulas are summarized here. Assuming that the additive noise is stationary, $\mathcal{E}\{\boldsymbol{z}^{(\Delta)}\} = 0$, and the channel noise is constant, *i.e.* $\boldsymbol{h}^{(\Delta)} = 0$, the delta noisy speech mean may be approximated by

$$\boldsymbol{\mu}_y^{(\Delta)} \approx \mathcal{E}\left\{\frac{\partial \boldsymbol{y}_{\text{vts}}^{(s)}}{\partial t}\right\} = \mathcal{E}\left\{\frac{\partial \boldsymbol{y}_{\text{vts}}^{(s)}}{\partial \boldsymbol{x}^{(s)}}\frac{\partial \boldsymbol{x}^{(s)}}{\partial t} + \frac{\partial \boldsymbol{y}_{\text{vts}}^{(s)}}{\partial \boldsymbol{z}^{(s)}}\frac{\partial \boldsymbol{z}^{(s)}}{\partial t} + \frac{\partial \boldsymbol{y}_{\text{vts}}^{(s)}}{\partial \boldsymbol{u}^{(s)}}\frac{\partial \boldsymbol{u}^{(s)}}{\partial t}\right\} \approx \boldsymbol{J}_x\boldsymbol{\mu}_x^{(\Delta)}. \qquad (4.6)$$

Similarly for the dynamic noisy speech variance,

$$\boldsymbol{\Sigma}_y^{(\Delta)} \approx \mathcal{E}\left\{\frac{\partial \boldsymbol{y}_{\text{vts}}^{(s)}}{\partial t}\frac{\partial \boldsymbol{y}_{\text{vts}}^{(s)}}{\partial t}^{\top}\right\} - \boldsymbol{\mu}_y^{(\Delta)}\boldsymbol{\mu}_y^{(\Delta)\top} \approx \operatorname{diag}\left(\boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(\Delta)}\boldsymbol{J}_x^{\top} + \boldsymbol{J}_z\boldsymbol{\Sigma}_z^{(\Delta)}\boldsymbol{J}_z^{\top}\right). \tag{4.7}$$

Following the same procedure, the accelerate parameters are

$$\boldsymbol{\mu}_y^{(\Delta^2)} \approx \boldsymbol{J}_x\boldsymbol{\mu}_x^{(\Delta^2)} \tag{4.8}$$

$$\boldsymbol{\Sigma}_y^{(\Delta^2)} \approx \operatorname{diag}\left(\boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(\Delta^2)}\boldsymbol{J}_x^{\top} + \boldsymbol{J}_z\boldsymbol{\Sigma}_z^{(\Delta^2)}\boldsymbol{J}_z^{\top}\right). \tag{4.9}$$

To summarize, the noisy speech mean $\boldsymbol{\mu}_y$ and variance $\boldsymbol{\Sigma}_y$ could be approximated using the clean speech mean $\boldsymbol{\mu}_x$ and variance $\boldsymbol{\Sigma}_x$ with the additive noise mean $\boldsymbol{\mu}_z$ and variance $\boldsymbol{\Sigma}_z$, and the channel noise mean $\boldsymbol{\mu}_u$:

$$\boldsymbol{\mu}_y = \begin{bmatrix}\boldsymbol{\mu}_y^{(s)}\\ \boldsymbol{\mu}_y^{(\Delta)}\\ \boldsymbol{\mu}_y^{(\Delta^2)}\end{bmatrix} \approx \begin{bmatrix}\boldsymbol{\mu}_x^{(s)} + \boldsymbol{\mu}_u^{(s)} + \boldsymbol{C}\log\left(\mathbf{1} + \exp\left(\boldsymbol{C}^{-1}(\boldsymbol{\mu}_z^{(s)} - \boldsymbol{\mu}_x^{(s)} - \boldsymbol{\mu}_u^{(s)})\right)\right)\\ \boldsymbol{J}_x\boldsymbol{\mu}_x^{(\Delta)}\\ \boldsymbol{J}_x\boldsymbol{\mu}_x^{(\Delta^2)}\end{bmatrix} \tag{4.10}$$

$$\begin{aligned}\boldsymbol{\Sigma}_y &= \begin{bmatrix}\boldsymbol{\Sigma}_y^{(s)} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_y^{(\Delta)} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_y^{(\Delta^2)}\end{bmatrix}\\ &\approx \operatorname{diag}(\begin{bmatrix}\boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(s)}\boldsymbol{J}_x^{\top} + \boldsymbol{J}_z\boldsymbol{\Sigma}_z\boldsymbol{J}_z & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(\Delta)}\boldsymbol{J}_x^{\top} + \boldsymbol{J}_z\boldsymbol{\Sigma}_z^{(\Delta)}\boldsymbol{J}_z & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{J}_x\boldsymbol{\Sigma}_x^{(\Delta^2)}\boldsymbol{J}_x^{\top} + \boldsymbol{J}_z\boldsymbol{\Sigma}_z^{(\Delta^2)}\boldsymbol{J}_z\end{bmatrix}).\end{aligned} \tag{4.11}$$

The environment distortion parameters $\{\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z, \boldsymbol{\mu}_u\}$ are borrowed from the conventional GMM-HMM systems, and are estimated per test utterance using an iterative EM algorithm [36]. The standard VTS compensation assumes a clean speech model. To utilize the multi-style training data, noise adaptive training (NAT) has been proposed [146].

4.1.3 VTS-MVN

Our proposed VTS-MVN process that integrates the merits of both the global MVN and the utterance-based MVN is illustrated in Figure 4.2. In our VTS-MVN, the commonly constant feature normalization parameters $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ are now time-dependent and will be referred to as $\{\boldsymbol{\mu}^{(\tau)}, \boldsymbol{\Sigma}^{(\tau)}\}$ with τ indicating the time dependency. The

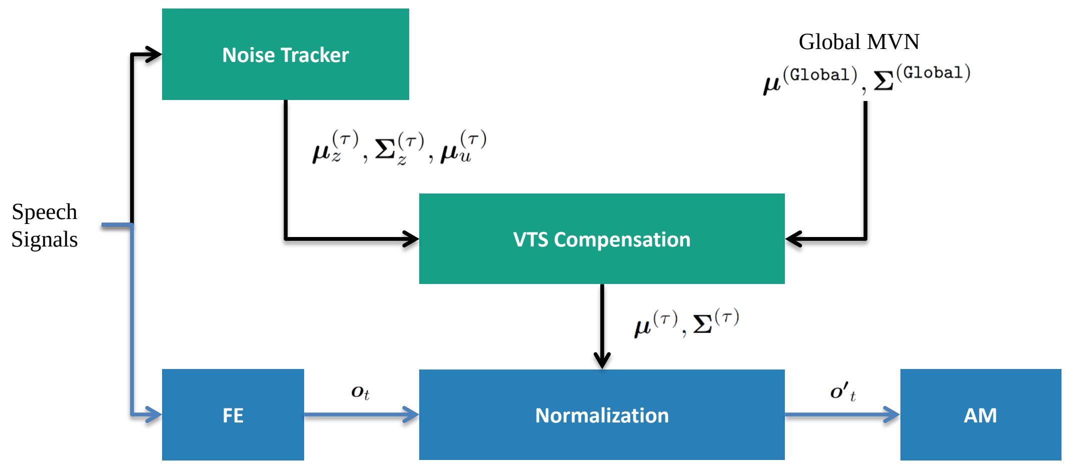

Noise Tracker
Global MVN
$\mu^{(\texttt{Global})}, \Sigma^{(\texttt{Global})}$
$\mu_z^{(\tau)}, \Sigma_z^{(\tau)}, \mu_u^{(\tau)}$
Speech Signals
VTS Compensation
$\mu^{(\tau)}, \Sigma^{(\tau)}$
FE
o_t
Normalization
o'_t
AM

Figure 4.2: A visual illustration of the VTS-MVN process.

feature normalization of equation (3.28) now becomes

$$o'_t = f_{\texttt{MVN}}(o_t) = (\Sigma^{(\tau)})^{-\frac{1}{2}}(o_t - \mu^{(\tau)}). \tag{4.12}$$

Initially, we set $\mu^{(0)} = \mu^{(\texttt{Global})}$ and $\Sigma^{(0)} = \Sigma^{(\texttt{Global})}$. At a certain time τ during the recognition process, the normalization parameters $\{\mu^{(\tau)}, \Sigma^{(\tau)}\}$ are obtained by compensating the global MVN prior with the current environmental parameter estimation $\{\mu_z^{(\tau)}, \Sigma_z^{(\tau)}, \mu_u^{(\tau)}\}$. These parameters are generated from a noise tracking component, which in our setup is the existing GMM-HMM system.

We use different time index variables for the feature vectors and the MVN estimations. It allows different time resolutions to be used, because normally finer time granularities are needed for the feature steam. For noise tracking, if we keep $\tau = 0$, the VTS-MVN reverts to the global MVN as we never update the normalization parameters through out the recognition process. In an extreme case, if we update the VTS-MVN parameters at each time frame, we are effectively doing a frame-dependent MVN which is usually unreliable. If τ is set to be the length of each utterance, and the noise tracker is simply computing the mean and variance of all the seen feature vectors, this system is then effectively doing an utterance-based MVN. Compared to the simple MVN calculation, our VTS-based noise tracker is more reliable. Generally speaking, the granularity of the time index τ, *i.e.* the update frequency of the noise tracker in our VTS-MVN system, effectively balances among reliability, latency and mismatch reduction capabilities. Deciding when to update the estimation is crucial. In our experiments, we use the utterance-based VTS-MVN and focus on evaluating the reliability gains from using the VTS-MVN derived from the prior global MVN. The reason for choosing τ to be the utterance length is that in the Aurora-2 corpus, the average utterance length is around 1.7s including the starting and ending pauses. It mimics the application scenario of recognizing short segmented speech data well, where the simple utterance-based MVN estimation is unreliable.

From another perspective, the normalization process defined in equation (3.28) could be represented as a linear input layer in front of the DNN with the bias $\boldsymbol{b}$ and the weight matrix $\boldsymbol{W}$ as

$$\boldsymbol{b} = \begin{bmatrix} -\frac{\mu_1}{\sigma_1} & -\frac{\mu_2}{\sigma_2} & \cdots & -\frac{\mu_D}{\sigma_D} \end{bmatrix}^\top \tag{4.13}$$

$$\boldsymbol{W} = \operatorname{diag}\Big(\begin{bmatrix} \frac{1}{\sigma_1} & \frac{1}{\sigma_2} & \cdots & \frac{1}{\sigma_D} \end{bmatrix}^\top \Big) \tag{4.14}$$

and

$$\boldsymbol{o}'_t = \boldsymbol{W}\,\boldsymbol{o}_t + \boldsymbol{b}. \tag{4.15}$$

The $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are initialized to the global MVN $\{\boldsymbol{\mu}^{(\texttt{Global})}, \boldsymbol{\Sigma}^{(\texttt{Global})}\}$. With the update of the environment estimations, the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are accordingly updated to $\{\boldsymbol{\mu}^{(\tau)}, \boldsymbol{\Sigma}^{(\tau)}\}$, which then leads the updated layer weight parameters $\boldsymbol{W}^{(\tau)}$ and $\boldsymbol{b}^{(\tau)}$. The final normalized features are

$$\boldsymbol{o}'_t = \boldsymbol{W}^{(\tau)}\,\boldsymbol{o}_t + \boldsymbol{b}^{(\tau)}. \tag{4.16}$$

As the compensation updates the DNN layer weights directly, it could be considered as a simple model-based compensation for the DNN to a certain degree.

4.1.4 Feature-based VTS

From the feature normalization perspective, our VTS-MVN compensates the normalization parameters to generate reliable normalization feature representations, which may look like a feature-based VTS. However, they are quite different. From [145], a GMM that represents the clean speech feature distribution has to be estimated. The pseudo-clean features are computed using the MMSE from the noisy observations. With the first-order VTS approximation, they are computed as

$$\begin{aligned} \boldsymbol{o'}^{(\texttt{MMSE})}_t &= \mathrm{E}(\boldsymbol{o}'_t|\boldsymbol{o}_t) = \int \boldsymbol{o}'_t\, p(\boldsymbol{o}'_t|\boldsymbol{o}_t) d\boldsymbol{o}'_t \\ &= \boldsymbol{o}_t - \sum_{k=0}^{K-1} p(k|\boldsymbol{o}_t)\Bigg(\boldsymbol{\mu}_u + \boldsymbol{C}\log\Big(\boldsymbol{1} + \exp\big(\boldsymbol{C}^{-1}(\boldsymbol{\mu}_z - \boldsymbol{\mu}_{x,k} - \boldsymbol{\mu}_u)\big)\Big)\Bigg) \end{aligned} \tag{4.17}$$

where $p(k|\boldsymbol{y})$ is the posterior probability for the kth Gaussian in the noise-compensated GMM given the noisy feature $\boldsymbol{y}$. The $\boldsymbol{\mu}_{x,k}$ is the mean of the kth Gaussian in the clean GMM. Compared to this feature-based VTS, our approach (equation (4.16)) not only shifts the noisy speech, but also scales it to estimate the pseudo-clean speech. The additional scaling step actually captures the variance changes between the clean and noisy speech. Moreover, multiple Gaussians usually have to be estimated for the feature-based VTS, while only a single Gaussian is involved in our approach.

4.1.5 Adaptive Training

Similar to the GMM-HMM, the VTS compensation assumes that the prior model is to be trained on clean data. When dealing with the multi-style data, the NAT is commonly adopted. Furthermore, as the single Gaussian based global compensation may limit its capability, we hope to relieve this limitation by using the DNN's powerful variation modeling capability through the adaptive training framework. A NAT based on our front-end VTS compensation is thus developed and the detailed training steps are presented in Algorithm 1.

Algorithm 1 Noise adaptive training of the VTS-MVN compensation for the hybrid DNN-HMM speech recognition system.

```
1: Train a DNN model from the multi-style data and estimate the initial environment
   distortion parameters from the beginning and ending frames for each utterance (20
   frames in our experiments);
2: Compensate the current DNN front-end and estimate a new set of distortion pa-
   rameters with the current DNN;
3: Re-train the DNN with the new noise compensated front-end;
4: Go back to step 2 until the recognition accuracy converges on a cross validation
   set. After the adaptive training, the distortion parameters are discarded and only
   the pseudo-clean DNN is kept for testing.
```

4.1.6 Discussions

Our VTS-MVN technique treats the global normalization mean and variance as a prior Gaussian representing the whole training data distribution of the dataset. Although it is a rather crude approximation, it works well in practice (*cf.* Section 6.2.1). For each testing utterance, this Gaussian is compensated using the first-order VTS model compensation technique to yield a noisy Gaussian where the data samples of the testing utterance are believed to be sampled from. The noisy Gaussian is then used to normalize the testing utterance before it is forwarded to DNNs. It is effectively estimating the MVN parameters for each testing utterance, which can directly be accumulated from that utterance, *i.e.* the utterance-based MVN. However, our method is more robust on short segments of speech data and has less latencies. In our approach, the global MVN parameters serve as the prior model, and the environment parameters are estimated from the target utterance using a GMM-HMM system. For the utterance-based MVN, when the speech variation is relatively small across the utterance, we may face the risk of removing both the noise and speech information. Most importantly, experimental results presented in Section 6.2.1 verify that the improved reliability from using this approach could lead to better performance compared to the simple utterance-based MVN.

4.2 Deep Split Temporal Context

The short-term memory of the auditory periphery in mammals [148, 149, 150] appears to be of the order of about 200ms. This means that the human auditory system can effectively utilize rather large time-spans of the audio signals. This is a time-span on a greater order of magnitude than that of the temporal window used in a typical short-term speech analysis for ASR. For example in Section 1.1, the commonly used time-span of each MFCC frame is only 25ms.

DNNs are becoming popular in ASRs. They have been shown to generalize much better than traditional models. The use of context windows to increase the input time-span plays an important role for them to outperform the conventional GMMs. Moreover, from the previous section, the borrowed noise estimations from the GMM-HMM for the whole utterance limit the effectiveness of the VTS-MVN. Estimating noise parameters in DNNs is more desirable. However, due to the discriminative nature of DNN learning, the high error rates in the decoded hypotheses and the limited enrollment data, learning the noise parameters in the DNN learning framework is challenging. As DNNs are superior at modeling variations, we are thus interested in whether they can automatically capture the noise environment variations from the inputs. To investigate the DNNs' environment learning capabilities, a long, or rather long, span of input features has to be used. With more input information, a better environment estimation may be achieved. In the literature, conventional noise tracking methods also utilize longer speech signals. In this work, we hence investigate the use of a long temporal-span context-expanded representations (equation (3.29)) to enable DNNs for a better understanding of the noise environment, which may further lead to improved noise robustness under adverse environments. To model a large input context window, increasing the input layer dimensionality is compulsory. Additionally, the increase of the input dimension also leads to an exponential increase in the number of input variations. The use of many high-dimensional hidden layers is crucial for improving DNNs' capacity. However, this often leads to the problem of over-fitting on small datasets.

Recently, a dropout technique is proposed to reduce the over-fitting by randomly omitting some of the feature detectors on each training case to achieve promising improvements on the TIMIT phoneme recognition task [151]. Similarly, a sparse variant of the DNN has also been proposed earlier for vision tasks [152]. In this work, we are motivated by the Split Temporal Context (STC) system [153] to explore the input independence inside a context window. Based on the recent progress in deep learning, we first simplify the original STC system and further propose a Deep Split Temporal Context (DSTC) system to incorporate the high-level abstraction-learning capabilities of DNNs. Experimental results on Aurora-2 in Section 6.2.2 show that the DSTC system not only generalizes better than basic DNNs but also has lower model complexity.

4.2.1 Split Temporal Context

Before the success of training DNNs, the shallow NN (Figure 4.3(a)) with 1 or 2 hidden layers is commonly adopted in the hybrid NN-HMM systems [15]. Due to its shallow structure, the acoustic modeling capability is limited. One of the early systems that aims to improve its performance is the STC system (Figure 4.3(b)) [153], which assumes the independence of the left and right acoustic context and models them separately. Another NN is used to merge the two sets of partial context predictions to give the final decision. It can robustly model long-term acoustic dependencies and thus improves recognition performance. Recently, there is much interest in understanding the NN's capabilities in learning high-level abstract representations rather than just taking it as a black-box classification model. It can be easily formulated as a layered feature extraction and a linear classification with softmax normalization. It may be better to directly combine the partial context abstractions and leave the uncertainties to the final decision layer rather than making predictions based on the incomplete information. This leads to our revised STC system (Figure 4.3(c)), which combines the hidden representations instead of the output posteriors. Additionally, we hypothesize that with a good feature representation, a linear classifier is sufficient. The merging NN in the original STC is thus simplified into a single softmax layer.

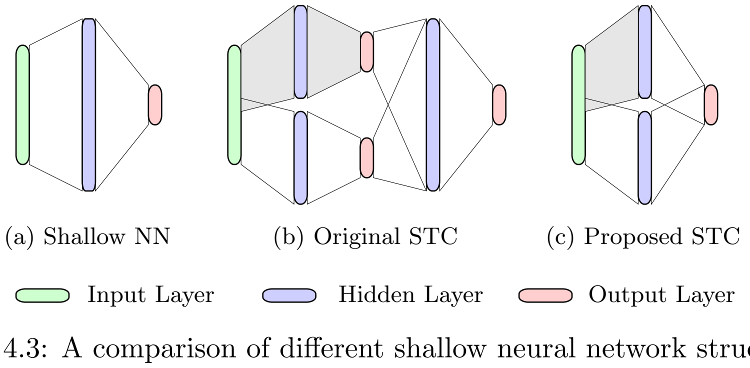


Figure 4.3: A comparison of different shallow neural network structures.

4.2.2 Deep Split Temporal Context

The DNN with multiple hidden layers (Figure 4.4(a)) has shown its superior feature abstraction capabilities, which makes it a perfect fit for the STC system (Figure 4.4(b)). However, one may argue that the DNN is already good enough to capture long term dependencies. It is true that with its multi-layered abstraction, the DNN has already largely outperformed the shallow NN. However, if the training data is relatively small and the potential testing variations are huge, we need to build a large model to guarantee model capacity, which then has the problem of over-fitting. The DSTC system (Figure 4.4(c)) is thus proposed, for addressing this issue. By assuming the independence in the input context window, we can use smaller DNNs to model the partial contexts separately. Due to the human speech production mechanism, there are strong co-articulation effects in speech which are reflected by local context dependencies. However, the long term dependencies are more associated with the grammars and meanings of the language rather than acoustics. It is thus safe to assume independence between short term acoustic contexts within a wide context window, which will also be validated later in the experiments.

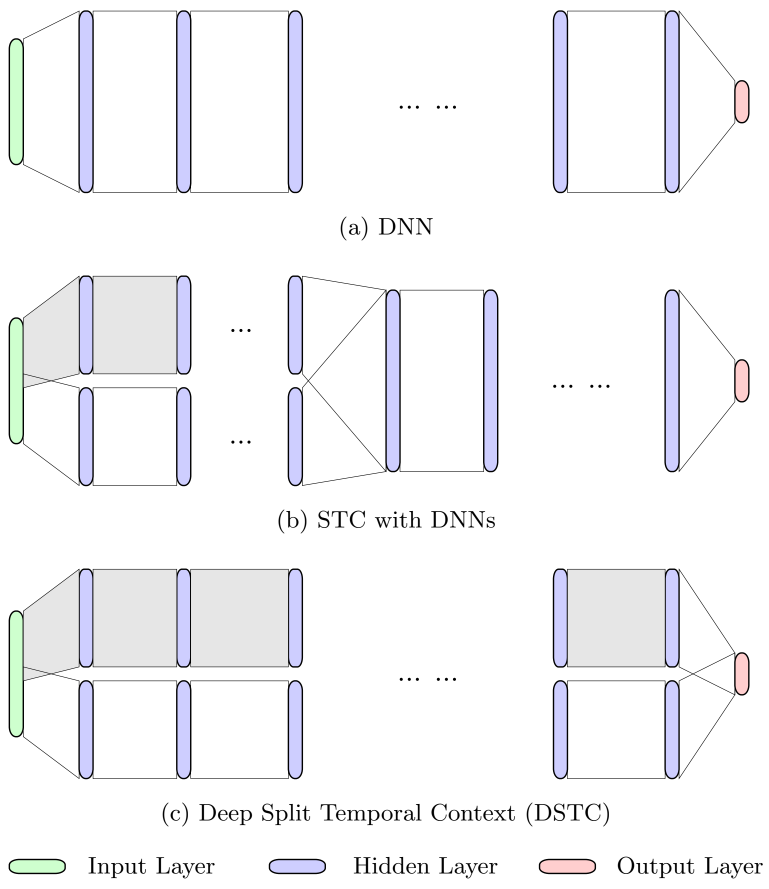


Figure 4.4: A comparison of different deep neural network structures.

4.2.3 Learning Algorithm

Algorithm 2 Training a 2-block (*i.e.* left- and right-context) DSTC system on $(2 * w + 1)$ frames of acoustic contexts with N hidden layers in each DNN using features $\mathbf{o}_t$ and labels $\mathbf{l}_t$ for each time slice t. The final system consists of the left context DNN $\mathcal{M}_{\text{left}}$, the right context DNN $\mathcal{M}_{\text{right}}$ and the final softmax layer $\mathcal{M}_{\text{softmax}}$.

```
1: Initialise $n = 0$, $\mathbf{x} = \{\begin{bmatrix}\mathbf{o}_t^T & \cdots & \mathbf{o}_{t+w}^T\end{bmatrix}^T \mid \text{for all } t\}$, $\mathbf{y} = \{\mathbf{l}_t^T \mid \text{for all } t\}$;
2: step 1. Shared Pre-training of the DNN $\mathcal{M}_0$. begin
3:   while $n < N$ do
4:     Train an RBM using $\mathbf{x}$;
5:     Set $\mathbf{x}$ to the hidden activations of the current RBM;
6:     $n = n + 1$;
7:   end while
8:   Stack the $N$ RBMs together;
9:   Append a randomly initialized classification layer to the stacked RBMs to form
     our initial DNN, $\mathcal{M}_0$;
10: end
11: step 2. Parallel fine-tuning of partial context DNNs. begin
12:   step 2.1. Fine-tune the left context DNN $\mathcal{M}_{\text{left}}$. begin
13:     Set $\mathbf{x} = \{\begin{bmatrix}\mathbf{o}_{t-w}^T & \cdots & \mathbf{o}_t^T\end{bmatrix}^T \mid \text{for all } t\}$;
14:     Fine-tune $\mathcal{M}_0$ until converge;
15:     Remove the final softmax layer to get $\mathcal{M}_{\text{left}}$;
16:     Forward $\mathbf{x}$ through $\mathcal{M}_{\text{left}}$ to generate $\mathbf{h}_{\text{left}}$;
17:   end
18:   step 2.2. Fine-tune right context DNN $\mathcal{M}_{\text{right}}$. begin
19:     Set $\mathbf{x} = \{\begin{bmatrix}\mathbf{o}_t^T & \cdots & \mathbf{o}_{t+w}^T\end{bmatrix}^T \mid \text{for all } t\}$;
20:     Fine-tune $\mathcal{M}_0$ until converge;
21:     Remove the final softmax layer to get $\mathcal{M}_{\text{right}}$;
22:     Forward $\mathbf{x}$ through $\mathcal{M}_{\text{right}}$ to generate $\mathbf{h}_{\text{right}}$;
23:   end
24: end
25: step 3. Train the final softmax layer $\mathcal{M}_{\text{softmax}}$. begin
26:   Set $\mathbf{x} = \begin{bmatrix}\mathbf{h}_{\text{left}}^T & \mathbf{h}_{\text{right}}^T\end{bmatrix}^T$;
27:   Train $\mathcal{M}_{\text{softmax}}$ from random initialization.
28: end
```

The training algorithm for the DSTC system is detailed in Algorithm 2. Although there are multiple partial context DNNs in the DSTC system, the training cost is actually reduced, compared to training a DNN with the same hidden-capacity on the complete context. When training the partial context DNNs, the unsupervised pre-training is shared and only the fine-tuning, which can be further parallelized, differs. In our experiments, we run 200 epochs for the input-to-hidden layer and 100 epochs for all the other layers in the pre-training phase, while maximum 20 epochs are used for fine-tuning. The main computation burden is the unsupervised pre-training. The additional softmax merging layer is a simple linear regression, which is much faster than DNN trainings. Moreover when comparing the DNN (Figure 4.4(a)) and the DSTC (Figure 4.4(c)), we are effectively replacing the DNN's full weight matrices with block diagonal ones, which reduces the number of model parameters. Examining the DSTC structure, one may wonder whether the DSTC's merging at the last hidden layer is optimal. Our experimental results (*cf.* Section 6.2.2) have shown that learning a better set of partial feature representations is much more important than adopting a complex back-end classification model.

4.2.4 Discussions

This study verifies that the increase of the context window size of the context expanded representations (equation (3.29)) improves DNNs' noise robustness by providing sufficient information for DNNs to automatically obtain better environment estimations. Detailed experimental verification will be presented in Section 6.2.2. To address the dramatic increase of the model size caused by the increase of the input dimensions and variations, a DSTC system is proposed. It builds upon small partial context DNNs with shared unsupervised pre-training. It is capable of maintaining a high model capacity while using relatively fewer model parameters. The training cost is much lower than a single fully connected DNN with the same model capacity. Most importantly, the separate modeling of the partial-contexts of the rather long context window of input features does improve the generalization capability of the DNN. As previously discussed, we are motivated by existing work, such as the dropout and the sparse model. However, those techniques do not change the model structure and cannot reduce the training cost. On the contrary, the dropout fine-tuning often requires hundreds of epochs to reach the optimum. Moreover, those techniques can all be applied directly to the partial DNNs of our DSTC system.

4.3 Spectral Masking

DNNs have shown superior capabilities in learning input variations not only from comparisons with traditional GMM-HMM systems but also because of the ineffectiveness of conventional feature-based noise robustness techniques. The previously discussed techniques, the VTS-MVN and the DSTC, do yield improved recognition performance. However, the gain on larger datasets are limited. For the VTS-MVN, it works with the assumption of the unreliability of the utterance-based MVN statistics. When dealing with normal speech where utterances are long, this assumption no longer holds and the benefit of using the VTS-MVN diminishes. For the DSTC, the challenge lies in how to first construct huge models using available computation power that could over-fit on the given corpora. We can only see the benefit of employing DSTC after that as its structure constraints could maintain a high model capacity but with less number of model parameters. Simply speaking, the gain of the DSTC comes from constraining the parameters of over-fitted DNN models. Hence, having over-fitted DNNs on the training data is compulsory for the effectiveness of the DSTC. However, with large amounts of data, building such models is difficult.

In searching for methods that could lead to further performance gains, insights from the human speech perception process may be helpful. The human auditory system is capable of efficiently identifying and separating speech and noise prior to understanding [154]. Therefore, in this section, we investigate this "separation-prior-to-recognition" process via spectral masking for noise-robust speech recognition. Firstly, a DNN-based Mask Estimator (ME) is developed. Estimated masks are then used to transform the noisy speech power spectrum into noise-invariant representations. Due to the use of DNNs for the mask estimation, the ME suffers from the mismatch problem. We propose to adopt the Linear Input Network (LIN) adaptation technique in our system. The LIN is effective in reducing training and testing mismatches for AMs. But the estimation of LINs for the MEs requires stereo data, which is not available during testing. To solve this problem, we modify the DNN to have an RBM input layer, namely the RBM-DNN. A LIN transform could then be estimated for the RBM front-end in an unsupervised manner, using contrastive divergence [109]. To further improve the robustness, we also replace the AM DNN with an RBM-DNN. Most importantly, by sharing the input RBM layer between the AM and the ME, the ME LIN transform could be learned by back-propagating the AM prediction errors.

4.3.1 Spectral Masking System

One important property of auditory nerve responses in human speech perception is that they respond preferentially to certain frequencies [155]. Motivated by the phenomenon of masking in the auditory perception, source segregation in computational auditory scene analysis is achieved by computing a mask to weight the Time-Frequency (T-F) representation, such as the spectrogram of acoustic signals. The mask applies a weight to each T-F unit, such that the spectral-temporal regions that are dominated by speech are emphasized, and regions that are dominated by other sources, such as noise, are suppressed. The values in the mask may be binary or real-valued. In the latter case, the mask value may be interpreted as the ratio of the target energy to the mixture energy, or the probability that the T-F unit belongs to the target speech. A time-frequency weight of this kind was first employed in the binaural source separation algorithm described in [156], and has subsequently been adopted by other researchers [157, 158]. Recently, these methods have also seen many applications in robust ASRs [159, 160].

With stereo data, Ideal Binary Masks (IBMs) [161] have been shown to substantially improve the intelligibility of speech with background noise [162]. The IBMs are computed in the power spectrum domain using:

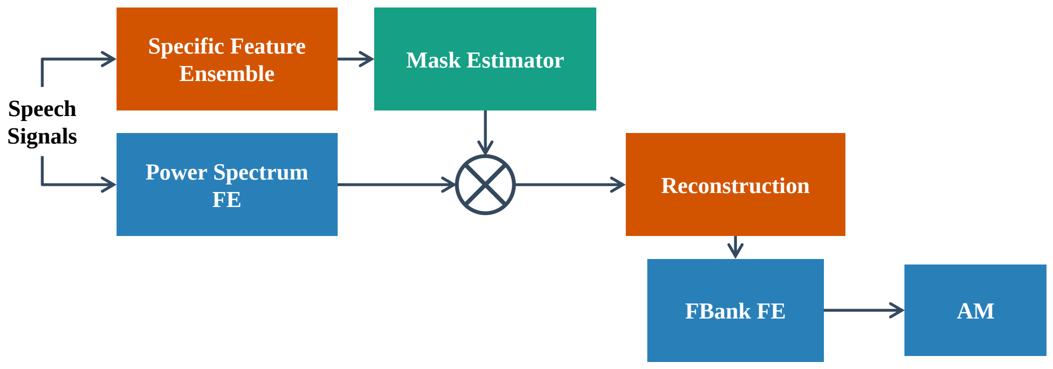
$$m_{t,c}^{(\texttt{IBM})} = \begin{cases} 1 & \text{if } r_{t,c}^{(\texttt{SNR})} > LC \\ 0 & \text{otherwise,} \end{cases} \tag{4.18}$$

where LC is a local SNR criterion [162] and $r_{t,c}^{(\texttt{SNR})}$ represents the local SNR at the time frame t and the frequency channel c. It is computed as

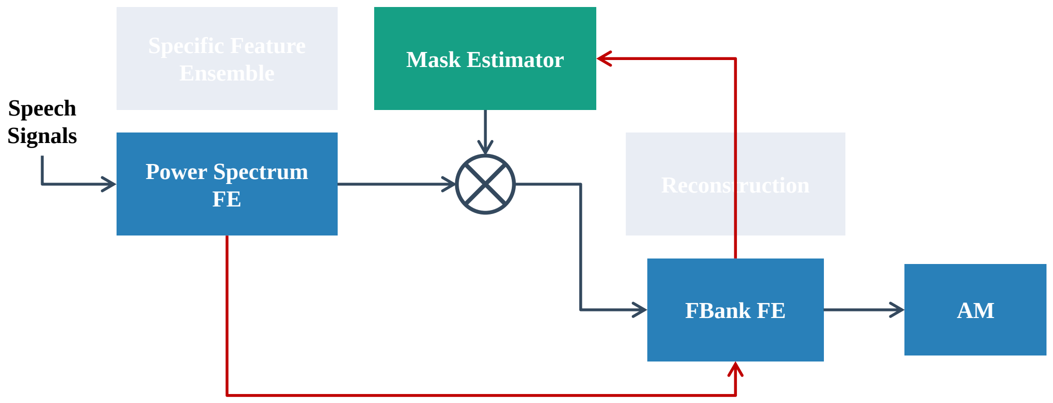
$$r_{t,c}^{(\texttt{SNR})} = 10 \log_{10} \frac{x(t,c)}{z(t,c)} \tag{4.19}$$

and $x(t,c)$ and $z(t,c)$ are the corresponding speech and noise energies. IBMs are used in a direct spectral masking manner to remove noise-dominated T-F units [160, 163]. Optionally, the discarded T-F units can be reconstructed by using the information from the speech-dominated units [76]. IBMs cannot be obtained in practical situations for spectral masking, since they are computed using stereo data. Therefore, various classification-based algorithms for IBM predictions have been developed [164, 165, 166, 167]. With the fast adoption of DNNs in various machine learning tasks, [167] and [168] have replaced their original support vector machine mask estimators with DNNs. In their work, an ensemble of different features are used as DNNs' inputs and the mask estimation is performed in two stages. Firstly, a total number of 27 DNNs using a single-frame input and 1,024 units per hidden layer are trained. In the second stage, a shallow neural network is estimated to give the final mask estimation by combining multiple frames of output predictions from the first stage DNNs. After masking, another reconstruction DNN is used to convert the masked partial spectral features to clean features, which are then used as inputs for the final acoustic model DNN.

In this study, we first propose to simplify the conventional spectral masking system. A visual comparison between the conventional spectral masking system and our simplified one is illustrated in Figure 4.5. The simplification mainly consists of two aspects: firstly, the features used for the prediction of IBM masks are reduced to include only the FBank features which are exactly the same as the input to acoustic models. As FBank features are drawn from human auditory systems, we believe the information captured is sufficient for both the mask estimation and acoustic modeling. Moreover, the use of the same inputs for the mask estimator and the acoustic model allows for the sharing of input transformations which is the key to ensuring the effectiveness of this spectral masking technique. Secondly, the simplification occurs because of how the masking is conducted. Although many researchers discussed the direct use of these spectral masks, feature reconstructions or uncertainty decoding are always adopted to address



(a) The conventional spectral masking system.



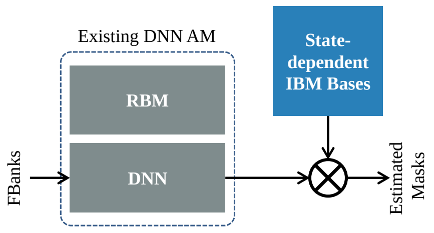
(b) The proposed simplified spectral masking system.

Figure 4.5: The proposed system simplification for spectral masking.

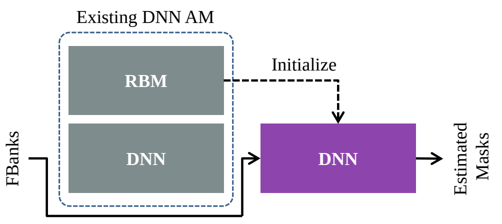
the missing information. From our experience in using enhanced features for DNN AMs, the masked partial features are more preferable to the imperfectly reconstructed pseudo-clean features. Additionally, from [167, 168], both the reconstruction and the acoustic modeling are using NNs (either shallow or deep). It may hence be possible to use a single DNN to jointly reconstruct the features and predict phonetic labels from partial inputs. Another probable reason for direct masking to become applicable would still be the improved modeling capabilities of DNNs over GMMs. Recently, [160] also reports that the use of direct masking is promising.

4.3.2 Mask Estimation

In our simplified spectral masking system (Figure 4.5b), the key component is the mask estimator. The quality of the estimated masks is crucial, as the masks decide which time-frequency information is passed to the acoustic model. Various classification-based algorithms for IBM predictions have been developed [164, 165, 166, 167, 167, 168]. For practical applications, an important consideration is the additional cost brought by

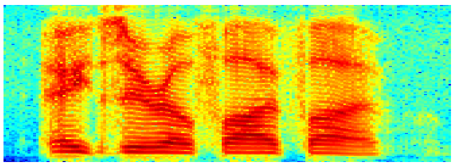


(a) State-dependent estimator.

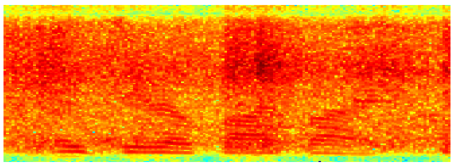


(b) DNN-based estimator.

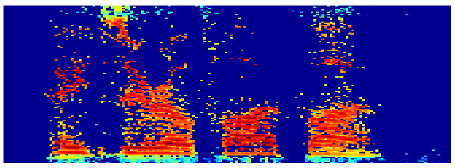
Figure 4.6: Two proposed mask estimators. The models inside the dashed box are those from the original DNN AM.



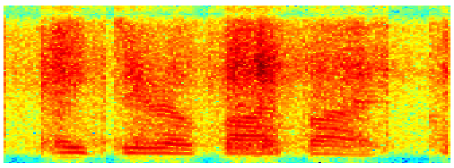
(a) Clean speech



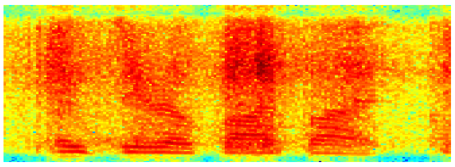
(b) With train noise (SNR=0dB)



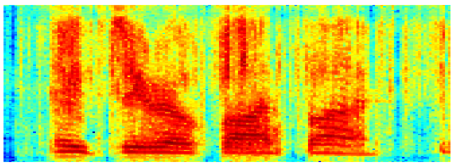
(c) With IBM



(d) With ideal state mask



(e) With state-dependent mask



(f) With DNN-based mask

Figure 4.7: Spectrograms of the same speech “8055” under different conditions.

the mask estimation, which should not be too taxing on the whole system. With this in mind, we propose two IBM estimation algorithms that reuse models from the existing DNN AM. They are the state-dependent (Figure 4.6a) and the DNN-based (Figure 4.6b) mask estimators.

State-dependent IBM Estimation

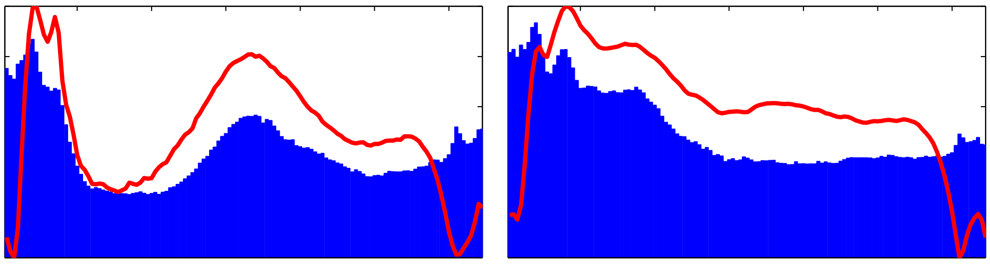
Compared to noise, speech is a kind of signal with more structure, which is also reflected by the capability of clustering speech features into phonetic clusters for recognition using statistical machine learning methods. Although IBMs depend on both speech and noise due to the local SNR computation, structure information about speech would also be crucial for identifying the speech-dominate units. For a specific phonetic state cluster, different IBM realizations should share similar structures. Based on this assumption, we propose a state-dependent IBM estimation approach. Utilizing the phonetic state clustering of the original recognition system, the IBM vectors for each time frame are grouped according to their corresponding speech feature vectors. There are several advantages of borrowing speech clusters rather than directly clustering IBM vectors. First, it saves the computational cost for the additional clustering. Secondly, the spectral feature based clustering would be more robust than the 0-1 based masks and thirdly, it also relieves the clustering process from being constrained by the limited stereo data.

After clustering, a canonical IBM pattern, which will also be referred to as an IBM basis, could then be estimated for each state cluster. In this research, we simply use the mean IBM vector of each state cluster as the basis for that specific phonetic state. The average IBM vector is interpreted as the expected probability for each T-F unit being marked as speech. With this set of state-dependent IBM bases, $\boldsymbol{B}$, the estimated mask vector for each test feature vector $\boldsymbol{o}_t$, is computed by

$$\boldsymbol{m}_t = \boldsymbol{B}\,\boldsymbol{p}_t, \tag{4.20}$$

where $\boldsymbol{p}_t$ is the original DNN AM posterior probability vector computed in equation (3.31) and is directly borrowed as the IBM basis coefficient vector. This process is also depicted in Figure 4.6a. The mask values estimated in this way are in the range of $[0, 1]$, rather than a discrete 0 or 1, as seen in IBMs. In consideration of the possible errors in estimations for $\boldsymbol{B}$ and $\boldsymbol{p}_t$, we take the estimated soft masks directly for spectral masking without binarization. Our approach differs from [169] in the way these phonetic dependent mask patterns are used. Instead of using them to refine a current estimation, we directly compose masks from them.

To gain an intuitive understanding on the effectiveness of our simple state-dependent mask estimation, spectrograms with and without masking are plotted in Figure 4.7. For Figure 4.7d, we use the ideal posterior vector $\boldsymbol{p}_t$ computed from the forced alignment of the test speech with true references to justify the effectiveness of the IBM bases without worrying about errors in posterior predictions. Due to the use of soft masks, noise cannot be completely removed; but compared to Figure 4.7b, the speech formant structure becomes much clearer in the masked spectrum. In practice, we use the existing DNN AM to generate $\boldsymbol{p}_t$ and the spectrogram in Figure 4.7e looks slightly noisier than Figure 4.7d but is still much better than Figure 4.7b. Furthermore, in Figure 4.8 two samples of IBM bases (blue bars) and the corresponding normalized speech spectral envelopes are plotted. A strong correlation could be observed, which also validates our previous assumption.



(a) The 11th HMM state of "6". (b) The 14th HMM state of "0".

Figure 4.8: Comparisons of state-dependent bases (blue bars) and speech spectral envelops (red contour) on Aurora-2.

DNN-based IBM Estimation

In the state-dependent approach, we borrow the posterior information generated from the original DNN. In this section, we revisit the learning procedure for DNNs. A commonly adopted DNN training recipe [112] is to firstly pre-train a stack of RBMs in an unsupervised way and then discriminatively fine-tune the whole DNN. The learned RBMs are capable of extracting general purpose high-level abstractions that are good representations of the original data, and the fine-tuning stage that comes after further optimizes them towards a specific task. Hence, if we optimize these RBMs for the prediction of IBMs rather than phonetic labels in the fine-tuning stage, the network would then be capable of generating masks for any given inputs (Figure 4.6b).

The DNN ME predicts a mask vector $\boldsymbol{m}_t$. Its c-th component, $m_{t,c}$, represents the probability, $P(m_{t,c}^{(\texttt{IBM})} = 1|\boldsymbol{h}_{0,t})$, that the cth power spectral component of the observation $\boldsymbol{o}_t$ is dominated by speech. The DNN input at time t consists of a window of $(2w+1)$ adjacent feature frames after normalization, *i.e.* $\boldsymbol{h}_{0,t} = \begin{bmatrix} \boldsymbol{o'}_{t-w}^{\top} & \cdots & \boldsymbol{o'}_{t}^{\top} & \cdots & \boldsymbol{o'}_{t+w}^{\top} \end{bmatrix}^{\top}$. The computation performed by an L-layer DNN ME is as follows:

$$\boldsymbol{h}_{l,t}^{\texttt{ME}} = \phi(\boldsymbol{W}_l^{\texttt{ME}} \boldsymbol{h}_{l-1,t}^{\texttt{ME}} + \boldsymbol{b}_l^{\texttt{ME}}), \quad \text{for } 1 \leqslant l < L \tag{4.21}$$

$$\boldsymbol{m}_t = \phi(\boldsymbol{W}_L^{\texttt{ME}} \boldsymbol{h}_{L-1,t}^{\texttt{ME}} + \boldsymbol{b}_L^{\texttt{ME}}), \tag{4.22}$$

where $\boldsymbol{W}_l^{\texttt{ME}}$ and $\boldsymbol{b}_l^{\texttt{ME}}$ are the model parameters for the l-th layer in the DNN ME; $\boldsymbol{h}_{l,t}^{\texttt{ME}}$ is the input to the $(l+1)$-th layer. $\phi(x)$ is the sigmoid non-linearity. In training, the ME model parameters $\theta^{\texttt{ME}} = \{(\boldsymbol{W}_l^{\texttt{ME}}, \boldsymbol{b}_l^{\texttt{ME}})|1 \leqslant l \leqslant L\}$ are firstly initialized using the pre-trained RBMs, and then fine-tuned using the standard EBP algorithm [101] to minimize the Mean Square Error (MSE) over the set of training samples $\mathcal{O} = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_T\}$:

$$\theta^{\texttt{ME}} = \arg\min_{\theta^{\texttt{ME}'}} \frac{1}{2} \sum_{t=1}^{T} \sum_{c} (m_{t,c} - m_{t,c}^{(\texttt{IBM})})^2. \tag{4.23}$$

To avoid the potential errors brought by binarizing the estimated masks, we also directly apply the DNN-generated real-valued masks to the noisy speech through a component-wise multiplication. An example of the masked power spectrogram obtained using our mask estimator DNN is illustrated in Figure 4.7f. From the visual comparison, this mask generates the most clean-like spectrogram.

Our DNN-based IBM predictor differs from [167] in two major aspects. Firstly, we use a single DNN initialized with existing pre-trained RBMs to directly predict masks from a window of temporal adjacent acoustic frames. In their approach, a bunch of DNNs are built from scratch to predict the the mask value for each frequency channel, and an additional MLP is involved to smooth out the prediction with temporal information captured in the masks. Secondly, the features for mask predictions are different. In our approach, we use the commonly adopted 24D FBank features. However, in [167], an ensemble of different features are concatenated to form the static input coefficients, which include 13D RASTA filtered perceptual linear predictive coefficients, 13D MFCCs, 15D amplitude modulation spectrogram, and 6D pitch-based features. Generally speaking, our approach makes it much easier for an existing DNN-HMM system to incorporate the spectral masking.

With the masks generated from the mask estimator, either a state-dependent one or a DNN-based one, the noisy speech power spectrum values are directly scaled. This way of applying masks is referred to as soft masking, which allows the circumvention of the extra step of determining proper binarization thresholds and potential errors when converting real-valued masks to binary ones. From the masked power spectrum, a new set of FBank features that are more invariant to noise, could be extracted accordingly. Using these noise-invariant features, the existing DNN AM can be retrained to yield a more robust phoneme posterior prediction.

4.3.3 Linear Input Network Adaptation

In speech recognitions, besides the noise corruption, large mismatches between the training and testing data are usually unavoidable due to the inherent variability of noise. Performance degradation is expected when the system is used in unknown noise conditions. This has also been observed for the DNN-based ASR systems in [141, 170, 171] and in our initial investigation on the DNN AM's noise robustness (Table 3.1 and Table 3.2). We hence propose to adopt a Linear Input Network (LIN) adaptation technique [140, 172, 173] to address the mismatch issue. Firstly, the mismatch problem affects the DNN-based ME. Erroneous mask estimations dramatically degrade the system performance, as observed in [164, 165, 166, 167]. However, it is not possible to directly estimate the LINs for the mask estimation DNNs during testing because it requires the IBM supervision labels. The computation of IBMs further requires par-

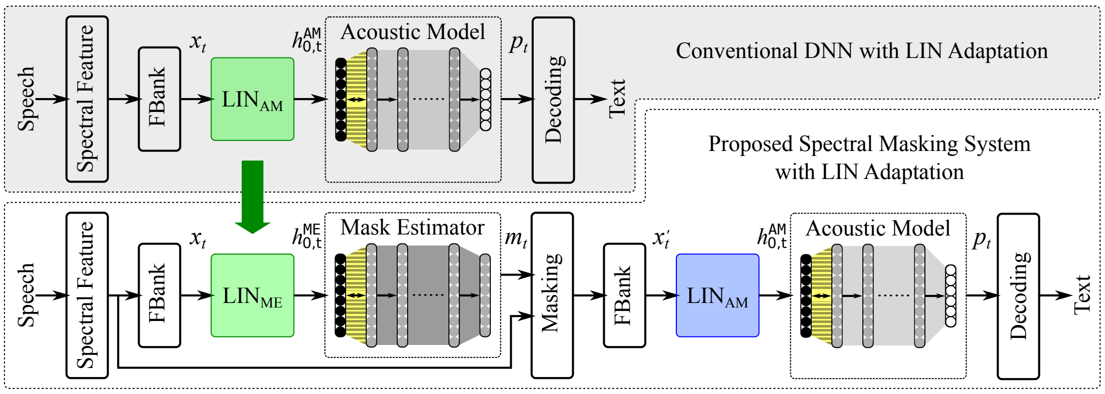


Figure 4.9: System architecture comparisons between the conventional DNN based acoustic model (the lightly shaded upper part) and the proposed spectral masking system (the unshaded lower part). The linear input network (LIN) adaptation transformations for the mask estimator and the acoustic model are represented as LIN$_{\text{ME}}$ and LIN$_{\text{AM}}$ respectively.

allel clean and noisy speech data, which is impossible to obtain during testing. Two approaches are proposed to solve this problem: the RBM-based LIN adaptation and the LIN sharing method. Secondly, the mismatch also happens in the masked feature domain. Although masking aims to remove noise such that the features are more similar to clean speech, it is usually unable to achieve this objective because of mask estimation errors (Figure 4.7f *vs.* Figure 4.7a). Moreover, even ideally masked features (Figure 4.7c) are different from clean speech (Figure 4.7a). Retraining the AM with masked features is compulsory. However, the different mask estimation accuracies of the ME DNN in the training and testing data may also cause potential mismatches between the masked features. Adopting additional adaptation transforms for the AM DNN is necessary and beneficial. Our final spectral masking system with a LIN adaptation is depicted in Figure 4.9. For comparison, a conventional DNN-HMM system with LIN adaptation is also illustrated in Figure 4.9. In this section we first review the LIN adaptation for the AM DNN and then present the proposed ME DNN adaptation.

Acoustic Model Adaptation

The LIN adaptation represents the training and testing mismatch with a weight matrix $\boldsymbol{T}^{\texttt{LIN}}$ and a bias vector $\boldsymbol{b}^{\texttt{LIN}}$. Instead of directly forwarding the observation to the DNNs, the LIN transformed one is used:

$$\boldsymbol{h}'_{0,t} = g_{\texttt{LIN}}(\boldsymbol{h}_{0,t}) = \boldsymbol{T}^{\texttt{LIN}}\,\boldsymbol{h}_{0,t} + \boldsymbol{b}^{\texttt{LIN}}. \tag{4.24}$$

It effectively adds an additional input layer to the original model without nonlinearity, which is why it is referred to as the LIN transform. The estimation of LIN transforms is based on the EBP and hence follows exactly the same procedure as the AM DNN training:

$$\boldsymbol{T}^{\texttt{LIN}}(\tau+1) = \boldsymbol{T}^{\texttt{LIN}}(\tau) + \Delta\boldsymbol{T}^{\texttt{LIN}}(\tau), \tag{4.25}$$

$$\boldsymbol{b}^{\texttt{LIN}}(\tau+1) = \boldsymbol{b}^{\texttt{LIN}}(\tau) + \Delta\boldsymbol{b}^{\texttt{LIN}}(\tau) \tag{4.26}$$

and

$$\Delta\boldsymbol{T}^{\texttt{LIN}}(\tau) = \alpha\Delta\boldsymbol{T}^{\texttt{LIN}}(\tau-1) + \eta * \frac{\partial\mathcal{E}}{\partial\boldsymbol{T}^{\texttt{LIN}}(\tau)}, \tag{4.27}$$

$$\Delta\boldsymbol{b}^{\texttt{LIN}}(\tau) = \alpha\Delta\boldsymbol{b}^{\texttt{LIN}}(\tau-1) + \eta * \frac{\partial\mathcal{E}}{\partial\boldsymbol{b}^{\texttt{LIN}}(\tau)} \tag{4.28}$$

where τ is the update iteration index, α is the momentum coefficient and η is the learning rate. Commonly, we start with $\boldsymbol{T}^{\texttt{LIN}}(0) = \boldsymbol{I}$ and $\boldsymbol{b}^{\texttt{LIN}}(0) = \boldsymbol{0}$. Supervision labels are required for the gradient computation. For unsupervised AM adaptation, we could use the recognition hypotheses. One potential problem is that the hypothesis errors may impede adaptation gain.

Mask Estimator Adaptation

Unlike the AM, no proper supervision labels could be used for ME adaptations. To solve this problem, we propose to use an RBM as the input layer for the DNN, which is referred to as the RBM-DNN. The RBM layer could be deemed as a statistical feature extractor that transforms the spectral acoustic features to hidden representations. With this generatively-trained RBM, we could estimate the LIN transform using CD [109]. The RBM energy function with the LIN is:

$$E(g_{\texttt{LIN}}(\boldsymbol{h}_{0,t}), \boldsymbol{h}_{1,t}) = -\,\boldsymbol{h}_{1,t}^T\,\boldsymbol{W}_1\,g_{\texttt{LIN}}(\boldsymbol{h}_{0,t}) - \boldsymbol{b}_1^T\,\boldsymbol{h}_{1,t} - \boldsymbol{a}_1^T\,g_{\texttt{LIN}}(\boldsymbol{h}_{0,t}) \tag{4.29}$$

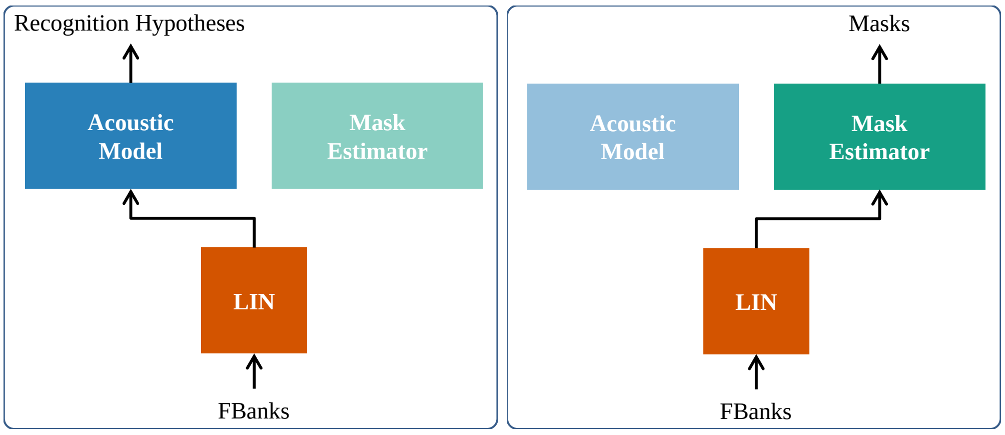
where $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{a}_1$ are the RBM parameters and the subscript 1 implies it is the first layer of the RBM-DNN. $\boldsymbol{a}_1$ is the input bias. The update of the LIN parameters by optimizing the testing data log likelihood using CD is:

$$\Delta\boldsymbol{T}^{\texttt{LIN}}(\tau) = \alpha\Delta\boldsymbol{T}^{\texttt{LIN}}(\tau-1) + \eta * (\langle\boldsymbol{h}_{0,t}\,\boldsymbol{h}_{1,t}^T\,\boldsymbol{W}_1\rangle_{data} - \langle\boldsymbol{h}_{0,t}\,\boldsymbol{h}_{1,t}^T\,\boldsymbol{W}_1\rangle_{model}), \tag{4.30}$$

$$\Delta\boldsymbol{b}^{\texttt{LIN}}(\tau) = \alpha\Delta\boldsymbol{b}^{\texttt{LIN}}(\tau-1) + \eta * (\langle\boldsymbol{W}_1^T\,\boldsymbol{h}_{1,t}\rangle_{data} - \langle\boldsymbol{W}_1^T\,\boldsymbol{h}_{1,t}\rangle_{model}). \tag{4.31}$$

One major concern about the RBM-based generative LIN estimation is that the limited adaptation data may be insufficient to move the existing model parameters from the current local optimum to a better one.

Another approach to adapting the ME is to borrow transforms those are esti-



(a) Training the LIN with AMs. (b) Applying the LIN for MEs in testing.

Figure 4.10: Mask estimator adaptation using LINs borrowed from acoustic models.

mated with a different objective. This is motivated from the success of borrowing the fMLLR transform from the GMM-HMM system to the shallow [140] and deep [174, 175, 176, 177] neural network acoustic models. Although there is no strict formulation to guarantee the validity of this kind of transform sharing, it is commonly deemed as a generic feature transformation that reduces mismatches. Unlike the system in [167, 168], the use of the same inputs between our AM and ME allows the exchange of feature transforms. We hence investigate the borrowing of adaptation transforms from the AM DNN to the ME DNN (Figure 4.10). We estimate the LIN transforms by back-propagating the recognition errors through the AM (Figure 4.10a) and during testing, this LIN is directly applied to the ME (Figure 4.10b). In addition, we also adopt the RBM-DNN for acoustic modeling in our system and further constrain the AM and the ME to share the same RBM input layer. Empirically, we have shown that the LINs estimated for the AM RBM-DNN perform much better for the ME RBM-DNN than those estimated for the pure DNN-based AM. One possible explanation is that the shared RBM serves as a regularization term to ensure that the LIN is suitable to generate shared hidden representations for both tasks.

Structure Constraints for LIN

The use of long-span acoustic features in DNNs is important to their superior performance, but it also causes a large increase in the number of adaptation parameters in the LIN transforms. For example, for the conventional MFCC-based GMM system, the fMLLR adaptation transform has around 1.5k parameters; while for a 11-frame input window DNN, the LIN transform has 184.5k parameters. With the same amount of limited enrollment data, the estimation of maximum likelihood based fMLLR is undoubtedly more reliable than that of the discriminative LIN. For a window of $(2w + 1)$

frames input, *i.e.* $\boldsymbol{h}_{0,t} = \begin{bmatrix} \boldsymbol{o'}^{\top}_{t-w} & \cdots & \boldsymbol{o'}^{\top}_{t} & \cdots & \boldsymbol{o'}^{\top}_{t+w} \end{bmatrix}^{\top}$, the LIN transform also has a similar block structure:

$$\boldsymbol{T}^{\texttt{LIN}} = \begin{bmatrix} \boldsymbol{T}_{-w,-w} & \cdots & \boldsymbol{T}_{-w,0} & \cdots & \boldsymbol{T}_{-w,w} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{T}_{0,-w} & \cdots & \boldsymbol{T}_{0,0} & \cdots & \boldsymbol{T}_{0,w} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{T}_{w,-w} & \cdots & \boldsymbol{T}_{w,0} & \cdots & \boldsymbol{T}_{w,w} \end{bmatrix}, \tag{4.32}$$

where each $\boldsymbol{T}_{i,j}$ is a transform similar to a fMLLR. $\boldsymbol{T}_{i,i}$ models the intra-frame correlation and $\boldsymbol{T}_{i,j}$ with $i \neq j$ models the inter-frame correlation between frame i and j. To reduce the number of parameters in LIN, we begin with removing all the inter-frame correlations by constraining $\boldsymbol{T}_{i,j} = \boldsymbol{0}$ for all $i \neq j$. This kind of LIN is referred to as the block diagonal LIN - "LIN(blk)". Furthermore, we could constrain all the intra-frame correlations to use the same transform. This is referred to as the shared block diagonal LIN, *i.e.* the "LIN(shd)". It has a comparable number of parameters to the fMLLR. In [168], a diagonal LIN has been adopted, which will be referred to as the "LIN(dig)". With this strong constraint, the LIN is effectively estimating the MVN statistics and is only used in utterance-based adaptations. The estimation of these variations is the same as the basic LINs, except for the additional structure constraints that have to be applied after each iteration of parameter update.

4.3.4 Discussions

In this study, we design a spectral masking system based on the existing studies. Firstly, the conventional spectral masking system is simplified to use only the FBank features as inputs and the feature reconstruction module after masking is discarded. After removing those components, the performance of the mask estimator become even more crucial to the success of the masking approach. Experimental justification will be presented in Section 6.2.3 and Section 6.2.4. To estimate high-quality masks, two mask estimation algorithms are further proposed, which both utilize the powerful DNNs. Although masking is effective in removing noise corruptions, the training and testing mismatch is another major factor that dramatically degrades ASR performance. The use of statistical models, including DNNs, renders the mismatch problem more crucial. To address the mismatch problem, we adopt the simple Linear Input Network (LIN) adaptation into our spectral masking system. Since the estimation of the LINs for the mask estimation DNNs requires stereo data, the LINs estimated for the acoustic model DNNs are used to adapt the mask estimators during testing. For the borrowing of the LINs to work well, the first layers of both the DNNs are constrained to share the same parameters, which are learned during the pre-training stage. Besides improving the reliability of transformation sharing, RBM-DNN has also been found to give a better performance compared to the pure DNN-based acoustic models on noisy speech. By combining the spectral masking for noise removal and the linear input network adaptation for mismatch reduction, we achieve the best average WER performance on both the Aurora-2 and Aurora-4 tasks. The detailed experiments could be found in Section 6.2.3 and Section 6.2.4.

4.4 Summary

In this chapter, we start with the investigation into the noise robustness of the normalization feature representations (equation (3.28)). The use of utterance-based MVN is always desirable. However, when the speech segment for recognition is short, the estimation of utterance-based MVN statistics is unreliable. To address this problem, a VTS-MVN technique that utilizes the global MVN as a priori and estimates environment parameters with a conventional maximum likelihood based GMM-HMM system. Based on the frequency of the environment parameter updates, the VTS-MVN could be easily adjusted to balance between reliability and effectiveness. Following that, we investigated the effectiveness of adjusting the window length for the context-expanded representation (equation (3.29)). Benefiting from DNNs' superior modeling capabilities, the use of longer contextual inputs do help improve DNN AM's generalization on unseen noise. However, the increase of the input dimensions leads to an exponential increase of input variations. Increases in DNN capacity are necessary for improved performance, which usually causes over-fitting problems. A DSTC algorithm is hence proposed, to build high capacity DNNs with relatively less weight parameters.

Although the VTS-MVN and DSTC have shown effectiveness in improving DNNs' noise robustness, they do have strong assumptions which may limit their effectiveness on scenarios that invalidate the assumptions. The VTS-MVN works well for short utterances. For long sentences, the simple utterance-based MVN is sufficient (*cf.* Section 6.2.1). The DSTC is effective only when the over-fitted DNN has an over-fitting problem on the training data (*cf.* Section 6.2.2). To further address the general noise corruption problem explicitly, we mimic the separation-prior-to-recognition process of human speech perception and develop a DNN-based spectral masking system. It differs from conventional masking systems that use large ensembles of various kinds of features. Instead, the same FBank feature is used for both mask estimation and acoustic modeling. The use of the powerful DNNs for acoustic modeling enables the direct use of the masked partial features without doing any feature reconstruction. The spectral masking has been experimentally verified to be effective in noise reduction but the DNN-based mask estimator cannot generalize well to unseen noise conditions (*cf.* Section 6.2.3 and Section 6.2.4). To address the degradation caused by mismatches, the LIN adaptation is incorporated into our spectral masking system. Two adaptation algorithms, namely the RBM-based generative adaptation and the transform sharing, are proposed to adapt the mask estimator, as the lack of supervision labels renders the direct adaptation of the ME impossible. The final spectral masking with the LIN adaptation system has been shown to yield the best performance in the literature on both the Aurora-2 and the Aurora-4 tasks as the time of writing. Detailed performance comparisons among different techniques developed in the literature and justified on these two corpora could be found in Section 6.4.

Chapter 5

Noise-Robust Hidden Representation Learning

The success of masking the noise variations in the input feature representations for the DNN AM further intrigues the interest in understanding the noise robustness of the automatically learned hidden representations in DNNs. With a deep layered structure, there are usually many more levels of hidden representations compared to the single input representation. Every hidden layer takes in the representation generated by the layers below and extracts a relatively higher level of abstraction for the layers above. Within this layered hierarchy, the lower layers are found to capture more local and feature-dependent information, while the higher layers capture more task-specific discriminations. In this chapter we extend our study of masking out noise variations into the hidden representations. Unlike spectral masking, it is hard to find a physical explanation for the hidden activations. The definition of the spectral IBM is not applicable in the hidden representation domain. Instead of distinguishing between speech and noise, we define an Ideal Hidden-activation Mask (IHM) that identifies the noise-invariant hidden units. With this IHM, we are capable of improving the DNN AM's performance without additional adaptations. This suggests that the IHM is more robust to estimation errors in the masks, compared to the spectral masking. The current IHM follows exactly the same system structure as the spectral masking that employs an additional DNN as the mask estimator. By further analyzing the effects of masking the hidden activations, it is found that the masking could be simulated by attenuating the sigmoid activation function with a bias shift. This shifting offset can be further decomposed into the product of a code vector and a transformation matrix, which are optimized to minimize the differences between the generated hidden activations of noisy speech and corresponding clean speech. The code vector is hence referred to as the noise code due to this noise variation minimization objective.

5.1 Hidden-Activation Masking

The spectral masking approach, especially augmented with the LIN adaptation, has shown some impressive results in improving DNNs' noise robustness (*cf.* Section 4.3.1, Section 6.2.3 and Section 6.2.4). It is adapted from humans' separation-prior-to-recognition speech perception process [154, 155]. Masks are adopted to separate speech from noise in the power spectral domain, where the noise and speech energy are assumed to interact in an additive manner. An Ideal Binary Mask (IBM) [161, 178] is commonly used to identify each unit in the power spectral representation of the noisy signals as speech dominant or noise dominant. The IBMs are theoretically defined to be binary, but due to the potential estimation errors and the need to select proper binarization thresholds, the estimated masks are usually used in a soft masking manner. That is to say, the estimated real-valued IBMs are directly used without binarization to scale the noisy spectral energy rather than having to do a binary selection. The scaled spectral energies are deemed as estimations of clean speech energies. These real-valued masks could be interpreted as the expected probabilities of the corresponding time-frequency components being speech dominant. More intuitively, they are the estimated speech ratio in the noisy components. As the use of real-valued masks rather than the binary ones are becoming more and more popular, directly adopting a soft mask through-out the training and testing may render improved performance. Recently, an Ideal Ratio Mask (IRM) in the same power spectral domain has been developed and shown to be more effective than the IBMs [167]. While continuing the investigation of learning noise-robust representations for DNN AMs in this research, we are interested in extending the idea of masking into DNNs' hidden layers. An Ideal Hidden-activation Mask (IHM) is hence developed. Unlike the spectral masks separating speech dominant and noise dominant units, the IHM evaluates the noise invariance of each hidden unit and discards those generating inconsistent activation levels for speech from different noise conditions.

5.1.1 Assumptions

The DNN's hidden representations are firstly learned in an unsupervised manner to capture the underlying data distribution of the input features and then discriminatively fine-tuned to optimize a specific task. These multiple levels of automatically generated representations have been found to gradually capture various aspects of the data and collectively yield the superior modeling capability of DNNs. It is hence important to justify the necessity of employing the masking technique into these hidden representations before conducting further investigations.

One of the conditions for adopting the masking technique is the existence of variations caused by noise. Different noise in the speech signals may cause variations in the corresponding feature representation. In the human-engineered power spectral domain, these variations are clearly observable through a simple visual comparison between the clean and noisy speech power spectrum (Figure 3.7). However, the visual inspection fails to give us sufficient discrimination between the clean and noisy hidden representations as they are optimized for machines rather than humans. The performance degradation on both the Aurora-2 (Table 3.1) and the Aurora-4 (Table 3.2) may give us some hints about the possible noise variations in DNNs' hidden representations. A more direct comparison would be helpful. Based on the probabilistic interpretation of these hidden representations, a mathematical justification using the KL-divergence is carried out. For each hidden unit, the activation value is in the range between 0 and 1 as a result of the sigmoid nonlinearity. It could be deemed as the probability of this unit being activated by the given input signal. For the tth input observation, the KL-divergence at the lth hidden layer is computed as

$$\mathcal{D}^{(\mathtt{KL})}(\boldsymbol{h}_{l,t}^{(\mathtt{clean})}||\boldsymbol{h}_{l,t}^{(\mathtt{noisy})}) = \sum_{f} \ln\left(\frac{h_{l,t,f}^{(\mathtt{clean})}}{h_{l,t,f}^{(\mathtt{noisy})}}\right) h_{l,t,f}^{(\mathtt{clean})} \tag{5.1}$$

where $\boldsymbol{h}_{l,t}^{(\mathtt{clean})}$ and $\boldsymbol{h}_{l,t}^{(\mathtt{noisy})}$ are the clean and noisy hidden activation vectors for the lth hidden layer at the time frame t respectively. For a given test set, this KL-divergence is averaged across all the feature frames. The KL-divergences for all the six hidden layers of the baseline DNN on the Aurora-4 task between the clean speech (test set 01) and the other 13 speech sets with different noise (test set 02 $\sim$ 14) are depicted in Figure 5.1.

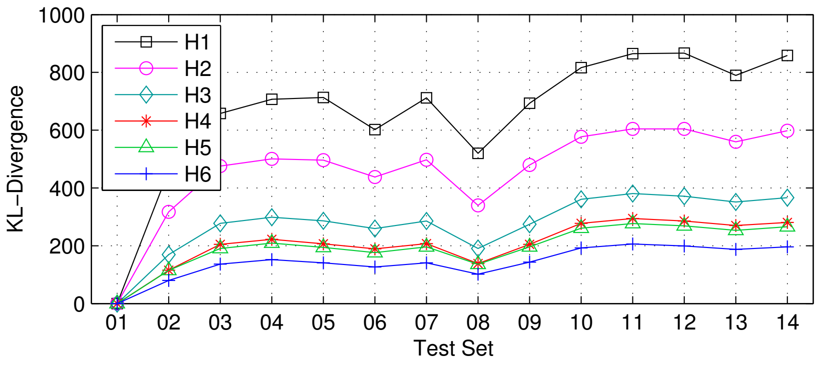


Figure 5.1: The average KL-divergence between noisy and clean hidden representations at different hidden layers of the baseline DNN on Aurora-4.

From this comparison using KL-divergence, more noise variations are seen in the lower layer hidden representations than the higher ones. This is consistent to the observations other researchers have found that the lower layers are more dependent on the input features. It also suggests that for DNNs, a sufficient number of hidden layers is compulsory to achieve its better variation-modeling capabilities. For our six hidden layer DNN, although much smaller, the last hidden layer, *i.e.* the 6th hidden layer - H6, still has noise variations that generates noticeable KL-divergence values. Hence we are more confident of improving the DNN AM's noise robustness by adopting proper masks to reduce the noise variations in their hidden representations.

Besides the assumption of the existence of noise variations, another major concern is whether there is redundancy in the DNN's hidden representations. Generally speaking, the masking technique will discard parts of the information in existing representations. The masking technique helps only if information redundancy exists. Otherwise, the degradation caused by information loss may overwhelm the gains brought by masking. The simplest way to justify this assumption would be to evaluate the masks directly, which is deferred to the next section after we present the definition of our ideal hidden-activation mask.

5.1.2 Ideal Hidden-Activation Mask

Speech data arises from the rich interaction of many sources. These factors interact in a complex way that complicates the recognition task. If we could identify and isolate these factors, we would largely ease the learning problem. The powerful advantage of DNNs over GMMs in modeling large acoustic variations also comes from DNNs' high-level abstraction capabilities in identifying the underlying factors. With the guidance of task-specific supervisions at the final output layer, the distributed hidden representations at each layer try to encode only the underlying speech-dependent factors and discard noise factors in its input features. Using many layers' nonlinear transformations, DNNs could encode a rather complex relationship between the original acoustic features and the target classification labels. However, due to the commonly adopted gradient-based learning, the supervision strength decreases through many layers' back-propagation and the confusion increases. The layers near inputs are believed to maintain more redundancies to avoid missing any potential clues. When the testing data is similar to the training data, these redundant feature detectors have similar active levels as those that have been seen during training and hence will not cause any problems. But when there are noise variations, they may become unexpectedly active and lead to possible performance degradation. We thus propose to mask away the unreliable feature detectors in DNN layers for improved noise robustness.

Due to the lack of intuitive relationships between the hidden units and the target classification labels, we use parallel speech data to guide the learning of noise-invariant hidden detectors. By comparing the hidden activations generated from noisy and corresponding clean speech, we can identify activations that are consistent between them.

The corresponding feature detectors (*i.e.* the hidden units) will then be marked as noise-invariant. This mask is named as the Ideal Hidden-activation Mask (IHM). By applying this IHM, the hidden representations will become less noise-prone and the following DNN layers may easily yield correct predictions. The mathematical formulation of the IHM is defined as follows:

$$m_{l,t,f}^{(\texttt{IHM})} = \begin{cases} 1 & \text{if } q_{l,t,f} > \kappa \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

$$q_{l,t,f} = \exp\{-\lambda * (h_{l,t,f}^{(\texttt{clean})} - h_{l,t,f}^{(\texttt{noisy})})^2\} \tag{5.3}$$

where $q_{l,t,f}$ denotes the similarity between the DNN's clean hidden activation, $h_{l,t,f}^{(\texttt{clean})}$, and the DNN's noisy hidden activation, $h_{l,t,f}^{(\texttt{noisy})}$, of the fth hidden unit in the lth hidden layer at the tth time frame. The parameter λ controls the shape of the similarity curve and κ is the threshold deciding whether a detector is noise-invariant. By default, we use the setting of $\lambda = 1.0$ and $\kappa = 0.5$. The similarity curves (equation (5.3)) with different λ values are plotted in Figure 5.2.

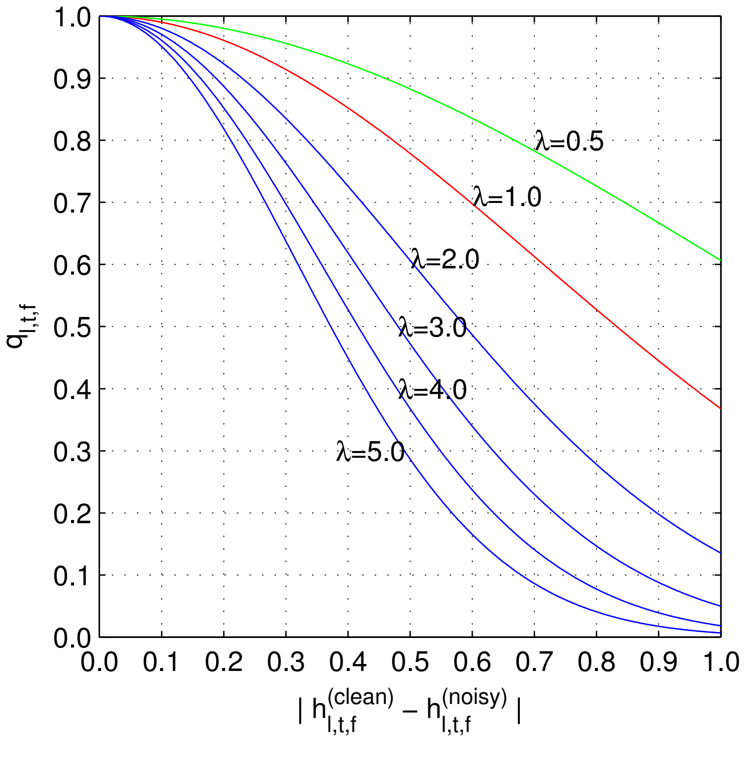


Figure 5.2: The similarity function for the IHM.

To validate the redundancy assumption, we apply the IHM with the default setup to all six hidden layers of the baseline DNN on the Aurora-4 task. The final recognition performance per test set is illustrated in Figure 5.3. Applying the IHM at any of the hidden layers improves the recognition performance on all the noisy test sets. The largest gain is obtained by applying the IHM at the first hidden layer (H1), which may be due to the existence of large noise variations. Except for the H1, the performance difference gained by applying the IHM among the other hidden layers is relatively small. It suggests that applying the IHM to reduce noise variations at earlier stages is better. In the following explorations, we will focus mainly on the first hidden layer.

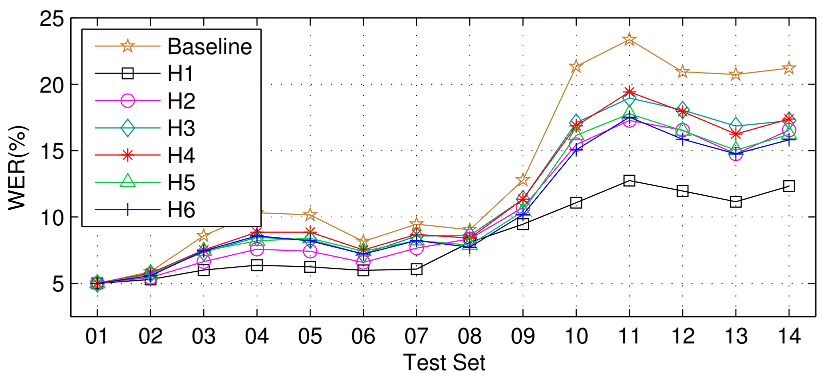


Figure 5.3: WER(%) performance of applying the default IHM ($\lambda = 1.0$ and $\kappa = 0.5$) at different hidden layers of the baseline DNN on Aurora-4.

Due to the existence of the two parameters, λ and κ, in the definition of the IHM (equations (5.2) and (5.3)), it may require additional efforts to determine the appropriate values for them in practice. We thus investigate the sensitivity of the IHM to different values of these two parameters. Firstly, the results of using different λ with $\kappa = 0.5$ are presented in Figure 5.4. The performance difference is relatively small, except for $\lambda = 0.5$, which is equivalent to the baseline. This is because from Figure 5.2, the configuration $\lambda = 0.5$ and $\kappa = 0.5$ simply generates all-one masks. Among all the values we have investigated, $\lambda = 2.0$ is slightly better; we thus use this value for the following investigations.

Next, we vary the threshold parameter κ while fixing $\lambda = 2.0$. When $\kappa = 0.1$, nothing will be masked away (Figure 5.2). When $\kappa = 0.9$, we mask away 37.0% of the active H1 hidden activations (those with values above 0.001) on average. We could still obtain a relative 29.0% average WER improvement on all the 14 test sets. While changing the κ from 0.2 to 0.6 (Figure 5.5), the WER performance only has small variations and reaches the minimum at $\kappa = 0.4$, which yields the average WER of 8.2% and the IHM average discarding ratio is 18.9%. The per test set IHM activation discarding ratios for $\kappa = 0.9$ and $\kappa = 0.4$ are also compared in Figure 5.6.

From the above investigation into the two parameters, λ and κ, the proposed IHM is relatively less sensitive to the specific value used for those two parameters. It is good, as no tedious tuning is required. In our study, we will use $\lambda = 2.0$ and $\kappa = 0.4$ for all

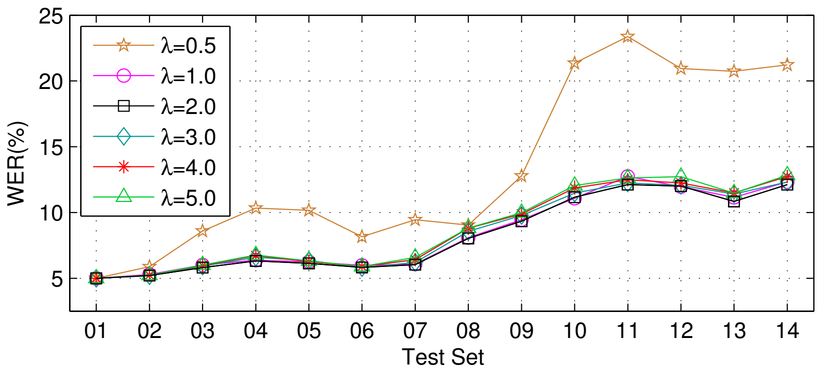


Figure 5.4: WER(%) performance of applying the IHM at the first hidden layer of the baseline DNN with different λ values and fixed $\kappa = 0.5$ on Aurora-4.

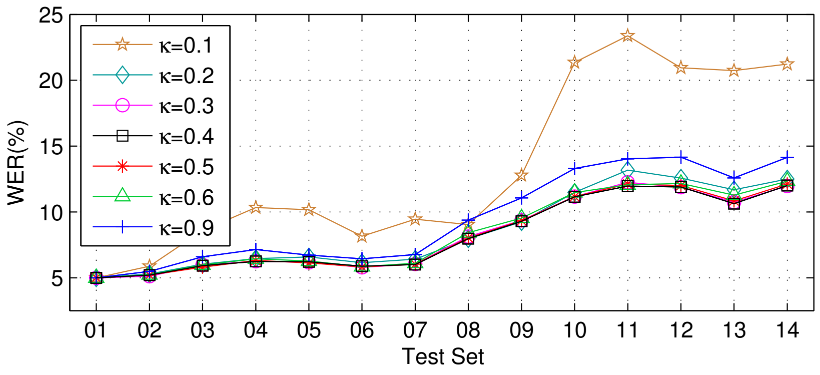


Figure 5.5: WER(%) performance of applying the IHM at the first hidden layer of the baseline DNN with different κ values and fixed $\lambda = 2.0$ on Aurora-4.

the experiments.

In practice, the lack of parallel data for testing utterances requires the estimation of IHMs. In our study, a DNN-based mask estimator is learned, with the training IHMs as supervision targets. During testing, we directly use $m_{l,t,f}^{(\mathtt{IHM})} = q_{l,t,f}$ to alleviate potential errors in the mask estimation.

5.1.3 Comparisons

The IHM is developed by extending the spectral masking approach into the DNN's hidden representations. It is hence important to understand the similarities and differences. For a more complete comparison, a recently proposed IRM [167, 168] is firstly described. The IRM is defined as:

$$m_{t,c}^{(\mathtt{IRM})} = (1 + \exp(-\xi * (r_{t,c}^{(\mathtt{SNR})} - \beta)))^{-1} \tag{5.4}$$

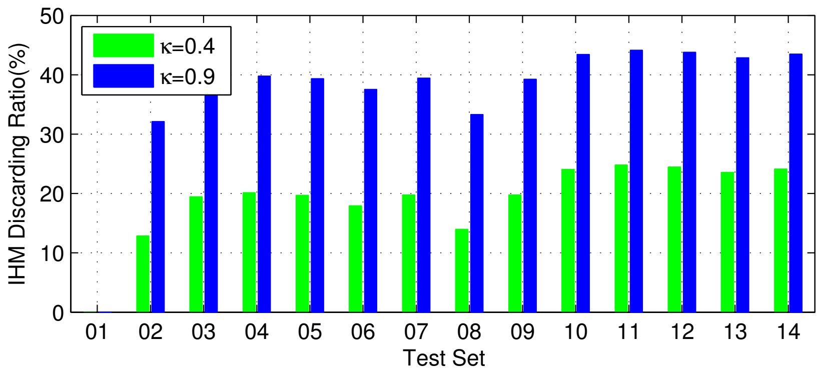


Figure 5.6: The the discarding ratios of active hidden features (> 0.001) by applying the IHM at the first hidden layer of the baseline DNN with different κ values and fixed $\lambda = 2.0$ on Aurora-4.

where $r_{t,f}^{(\mathtt{SNR})}$ is the instantaneous local SNR of the time-frequency unit at the time frame t and the frequency channel c (equation (4.19)). ξ controls the slope of the sigmoid function and β corresponds to the LC. By tuning ξ and β, we can control the range of SNRs to focus on while training the mask estimators. As suggested in [167], $\xi = 0.2$ and $\beta = -6$ dB are adopted.

These three masks, namely the IBM, the IRM and our IHM, require stereo training data, which is indicated by the word "ideal" in their names. This may also be one of the reasons that these masks could further improve DNN AMs' performance. The information captured in these masks by comparing the parallel clean and noisy data is different from that learned by the DNN AM.

The IBM and the IHM are binary masks defined for selection purposes. The IBM selects speech-dominant time-frequency units and discards noise-dominant units, while the IHM chooses hidden feature detectors in the DNN that are invariant to input variations caused by noise. This invariance is measured by the changes in the activation values. If the hidden feature detector generates similar activation values for both the clean speech frame and the same frame under noise conditions, we will say this detector is invariant to noise corruptions. The IRM differs from them by generating scaling ratios. It is defined to be the ratio of the pure speech energy to the mixture of speech and noise. However, in practice, to avoid the potential errors brought by binarizing the masks, even for the IBM and the IHM, we simply apply the estimated real-valued masks in a similar way to the IRM. Hence, the differences between the binary masks and the real-valued masks only exist in the training phase when parallel data is available.

The major difference between the spectral masks, the IBM and the IRM, and the proposed IHM, is the feature domain where masks are applied. The human-designed power spectrum domain is used for the spectral masks. The IHM is applied in the automatically-learned hidden representation domain. Due to the feature domain differences, the spectral masks are dependent on the SNR while the proposed IHM is based on the similarities between the clean and noisy activations. Ultimately, the spectral masks rely on the ratio between the speech signal and the noise, but the Euclidean differences between hidden units determine the IHM values. This may give IHMs more robustness as the change of speech energy only will not change the computed IHM value but do generate different spectral mask values.

The proposed IHM may look similar to the dropout technique [151], but they are different. The masking noise found in the dropout encourages a faster breaking of symmetry by randomly discarding some of the hidden activations. However, the IHM is a deterministic way of identifying the noise-invariant hidden detectors. The IHM may provide some insights in understanding the dropout. Meanwhile, the randomness in the dropout may reduce more variations beside noise-prone activations. It would serve as the guidance for our future development of the IHM.

5.1.4 Discussions

In this section, we extend the idea of masking away noise variations from the input representations to the hidden representations for improving DNN AMs' noise robustness. Unlike the traditional spectral masking techniques such as the IBM and the IRM, the SNR is undefined in those hidden representations. A similarity-based IHM is hence proposed. It operates at the DNN's distributed hidden representation space and removes the hidden feature detectors that generate inconsistent activation levels on noisy and corresponding clean speech. Moreover, the IHM is also found to be more independent towards the various noise types. Detailed experimental verifications on Aurora-4 are presented in Section 6.3.1.

5.2 Noise Code

The use of IHM further confirms the incorporation of additional parallel information improving DNN AMs' noise robustness. However, the current approach adopts a separate DNN for the estimation of masks computed from stereo training data. Although sharing the unsupervised RBM pre-training already reduces the training cost significantly, the use of two DNNs for recognition may still be over taxing. Hence, in this study we investigate alternative ways to achieve the same hidden-activation masking effects but with reduced computational complexities.

5.2.1 IHM and Sigmoid Function

For a given feature vector $\boldsymbol{o}_t$, after feature normalization (equation (3.28)) and context expansion (equation (3.29)), we have the input vector $\boldsymbol{h}_{0,t}$. As found in the previous discussions, the first hidden layer, H1, is the most effective place for applying the IHM. We will also focus on H1 of the DNN AM for this study. The H1 hidden activation vector $\boldsymbol{h}_{1,t}$ is computed as

$$\boldsymbol{h}_{1,t} = \phi(\boldsymbol{W}_1\, \boldsymbol{h}_{0,t} + \boldsymbol{b}_1). \tag{5.5}$$

For simplicity, the frame index subscript t and the layer index 1 of the model parameters are omitted and the hidden activation vector $\boldsymbol{h}_1$ now is

$$\boldsymbol{h}_1 = \phi(\boldsymbol{W}\, \boldsymbol{h}_0 + \boldsymbol{b}). \tag{5.6}$$

With the same input $\boldsymbol{h}_0$, the multi-layer DNN-based mask estimator computes the mask vector $\boldsymbol{m}_1$ for the 1st hidden layer using equations (4.21) and (4.22). The cth component of the masked hidden vector $\hat{\boldsymbol{h}}_1$ is computed as

$$\hat{h}_{1,c} = m_{1,c} * h_{1,c} \tag{5.7}$$

where $m_{1,c}$ and $h_{1,c}$ are the cth component of the mask vector $\boldsymbol{m}_1$ and the original hidden activation vector $\boldsymbol{h}_1$ respectively. When the binary mask vector is used, *i.e.* $m_{1,c}$ can only be 1 or 0, equation (5.7) could be rewritten as

$$\hat{h}_{1,c} = \begin{cases} h_{1,c} = \phi(\boldsymbol{w}_c^\top \boldsymbol{h}_0 + b_c) & \text{if} \quad m_{1,c} = 1 \\ 0 & \text{if} \quad m_{1,c} = 0 \end{cases} \tag{5.8}$$

where $\boldsymbol{w}_c$ is the cth row vector of the weight matrix $\boldsymbol{W}$ and b_c is the cth component of the bias vector $\boldsymbol{b}$.

The standard sigmoid activation function is defined as

$$\phi(x) = \frac{1}{1 + \exp(-x)}, \tag{5.9}$$

which is also plotted as the red curve in Figure 5.7. Adding an extra shifting variable ζ to the input variable x, we have the function $\phi(x + \zeta)$, which can be obtained by shifting the standard sigmoid curve along the x-axis direction by $-\zeta$. The plots for function $\phi(x + \zeta)$ with different ζ values are illustrated in Figure 5.7.

From a different perspective, if we fix the value x, with $\zeta < -6 - x$, we effectively turn off this function by setting all the output values to 0. It could also be represented

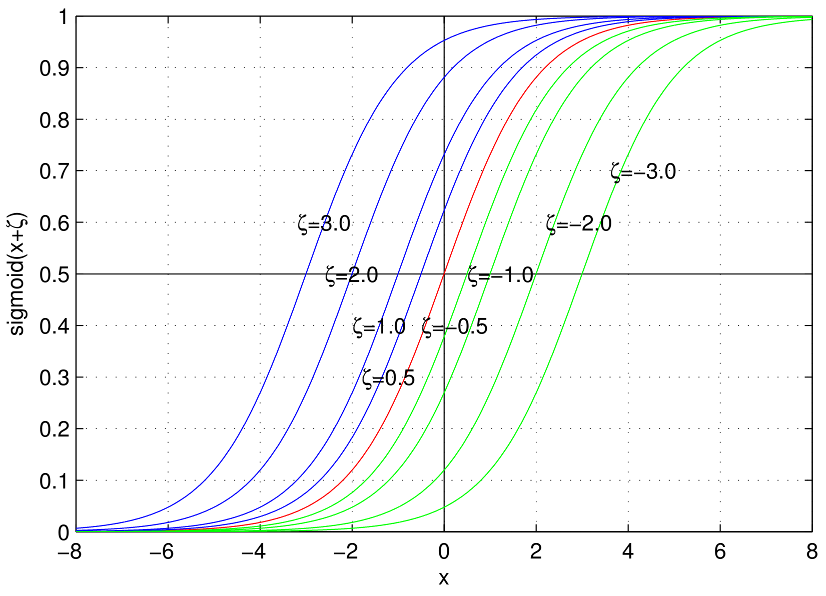


Figure 5.7: Sigmoid functions with different shifting offsets.

as

$$\phi(x+\zeta) = \begin{cases} \phi(x) & \text{if} \quad \zeta = 0 \\ 0 & \text{if} \quad \zeta < -6 - x \end{cases} \tag{5.10}$$

To zero out the function value, ζ depends on the value of x. In practice, we could simply find a value that is smaller than any possible x for ζ.

Comparing equation (5.8) and equation (5.10), we could reformulate the computation of the masked hidden vector as

$$\hat{h}_{1,c} = \phi(x_{1,c} + \zeta_c) \tag{5.11}$$

where $x_{1,c} = \boldsymbol{w}_c^\top \boldsymbol{h}_0 + b_c$ and we use ζ_c to suggest different shifting offsets could be used for different hidden units. This is actually approximating the masking effects at the hidden units by incorporating an additional bias to shift the sigmoid function. From equation (5.7) and equation (5.11), we could further derive the effective mask value $m_{1,c}$ as

$$m_{1,c} = \frac{\phi(\boldsymbol{w}_c^\top \boldsymbol{h}_0 + b_c + \zeta_c)}{\phi(\boldsymbol{w}_c^\top \boldsymbol{h}_0 + b_c)}. \tag{5.12}$$

5.2.2 Learning Algorithm

Based on the current approximation, we are effectively replacing the previous mask vector $\boldsymbol{m}_1$ with the bias offset vector $\boldsymbol{\zeta}$. It has to be noted that although $\boldsymbol{\zeta}$ itself is not dependent on time, the masking effect it causes is still time-dependent which could be seen from equation (5.12). To get rid of the extra DNN-based ME, we hence estimate $\boldsymbol{\zeta}$ within the DNN AM using

$$\zeta = \boldsymbol{T}\boldsymbol{c} \tag{5.13}$$

where $\boldsymbol{c}$ is the code vector which will be appended to the input vector $\boldsymbol{h}_0$ and $\boldsymbol{T}$ is the corresponding code transformation matrix. The same code transform $\boldsymbol{T}$ will be used for all the test utterances, but the code vector is utterance-dependent, which will effectively generate frame dependent masks together with the input vector.

By incorporating this code vector $\boldsymbol{c}$ and transformation $\boldsymbol{T}$, the final hidden activation vector $\boldsymbol{h}_1$ is computed as

$$\begin{aligned}\boldsymbol{h}_1 &= \phi(\begin{bmatrix}\boldsymbol{W} & \boldsymbol{b} & \boldsymbol{T}\end{bmatrix} * \begin{bmatrix}\boldsymbol{h}_0 \\ \boldsymbol{1} \\ \boldsymbol{c}\end{bmatrix}) \\ &= \phi(\boldsymbol{W}\boldsymbol{h}_0 + \boldsymbol{b} + \boldsymbol{T}\boldsymbol{c}) \end{aligned} \tag{5.14}$$

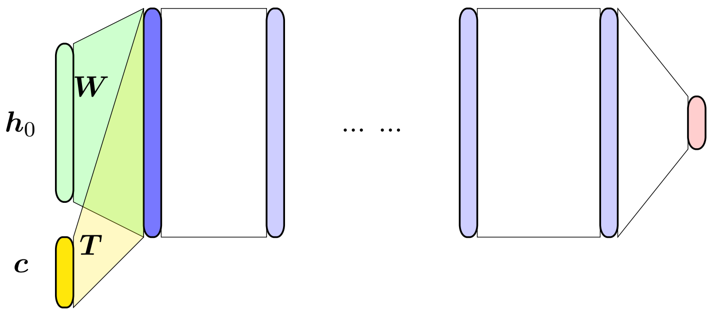


Figure 5.8: The model structure of a DNN with an input noise code vector.

To train such a systems (Figure 5.8), we start with a standard DNN AM without $\boldsymbol{c}$ and $\boldsymbol{T}$. After we obtain the weights $\boldsymbol{W}$ and $\boldsymbol{b}$, we then modify the DNN AM's input layer to support the additional code vector. The code transformation matrix $\boldsymbol{T}$ is randomly initialized and the code vector $\boldsymbol{c}$ is initially set to $\boldsymbol{0}$ to ensure that the learning starts from the baseline DNN. Due to the utterance dependence of the code vector $\boldsymbol{c}$, utterance-based learning rather than the conventional batch-based learning is adopted. For each utterance, the existing code vector is loaded or $\boldsymbol{0}$ is used for the first epoch. Both $\boldsymbol{T}$ and $\boldsymbol{c}$ are updated using EBP. After training, the code vector for this utterance is saved and a new utterance and the corresponding code vector will be loaded. The code transformation matrix $\boldsymbol{T}$ will be updated using all the training data.

The code vector $\boldsymbol{c}$ and the code transformation weight matrix $\boldsymbol{T}$ are introduced to estimate the bias shift vector $\boldsymbol{\zeta}$, with the objective of mimicking the hidden masking technique. Hence the estimation of these parameters are also based on the difference between the hidden activations generated from noisy and corresponding clean speech. The objective function is defined as

$$\mathcal{E} = \frac{1}{2}\sum_{c}(h_{1,t,c}^{(\texttt{noisy})} - h_{1,t,c}^{(\texttt{clean})})^2, \tag{5.15}$$

where $h_{1,t,c}^{(\texttt{noisy})}$ and $h_{1,t,c}^{(\texttt{noisy})}$ are the first hidden layer activations of noisy and corresponding clean speech. From this objective function, the code vector $\boldsymbol{c}$ and the code transformation $\boldsymbol{T}$ are explicitly optimized to reduce the mismatches caused by noise. They are hence addressing noise corruptions. The code vector is effectively coding the noise statistics for a given utterance. This is also why we refer to this approach as the noise code method. However, when testing without the parallel data, the noise code has to be estimated using the DNN AM's erroneous predictions. As all the modifications and computations are in the first hidden layer, the learning of $\boldsymbol{c}$ and $\boldsymbol{T}$ is actually done within a single DNN layer, which greatly reduces the training cost. Due to the changes in the first hidden layer activations caused by the use of noise codes, the remaining layers are re-trained after the estimation of all the noise code vectors is finished.

5.2.3 Comparisons

The major novelty of our noise code approach is the use of noise codes to incorporate additional parallel data information into the DNN AM. This clearly distinguishes our approach from existing ones using similar coding ideas. In [179], a neural network that has terminals for acoustic patterns and speaker parameters as inputs and class labels as outputs is proposed. The network is hence trained to "tune in" the speaker parameters to a particular speaker. Recently, it is ported to noisy speech recognition by appending a noise vector to the input [180]. Instead of learning the code vector from data, it is simply set to the average of feature vectors corresponding to the beginning and ending silences. The code transformation matrix is learned by back-propagating the DNN AM's prediction errors. Due to the over simplified code estimation and the same prediction error guided transformation matrix optimization, the gains obtained are negligible.

In [181, 182], a "speaker code" technique is proposed. Although it addresses speaker differences rather than noise variations, the concept is the same. Their approach operates similarly between the input layer and the first hidden layer. However, an additional adaptation DNN is used and the code vector is appended to each of the layers except for the output layer. All the model parameters including the adaptation DNN, the code vector, and the code transformation matrices corresponding to different layers, are estimated by back-propagating the DNN AM's prediction errors. On their specific task addressing speaker mismatches, improvements have been reported but the use of testing references for the estimation of speaker code vectors is undesirable for practical applications. It is interesting to know how well their technique performs when code vectors are estimated using erroneous recognition hypotheses rather than true references.

5.2.4 Discussions

In this section, we further analyze the ideal hidden-activation mask and find the masking could be simulated by using additional shifts in the sigmoid activation functions of the hidden units. This shifting offset is further decomposed into a product of a code vector and a transformation matrix. Borrowing the similar learning objective of the hidden masking, we estimate these parameters to minimize the mean square error between the generated hidden activations of the noisy speech and the corresponding clean speech. With this objective, the code vector is effectively capturing the noise statistics of each utterance and is referred to as the noise code. Our approach differs from others that adopt the similar idea of code vectors in the objective functions, as all the existing ones use the AM's prediction errors. Our objective utilizes parallel data information that is different from what the DNN AM has seen during training and is more promising in improving its noise robustness.

5.3 Summary

In this chapter, we extend the concept of masking away noise variations into automatically-learned DNN hidden representations. Although these representations already have a much better reliability for the phonetic predictions, an investigation reveals that noise variations and redundancies exist. Following the spectral masks, an Ideal Hidden-activation Mask (IHM) is first proposed to identify hidden feature detectors that are noise-invariant. A DNN-based mask estimator is then optimized with IHMs on the training data as supervision labels. By furthering analyzing the effects of masking the hidden units, it is found that the masking could be approximated using additional bias shifting in the sigmoid activation functions. This bias shifting offset is further represented as a product of a code vector and a transformation matrix. We then propose to estimate them using a similar objective to the masking approaches. This hence makes the code vector a noise specific one, thus the method is referred to as the noise code approach.

Chapter 6

Experiments

In this chapter, we justify the effectiveness of our proposed techniques in improving the noise robustness of the DNN AM. Firstly, the two benchmark datasets, Aurora-2 and Aurora-4, are introduced. The proposed noise-robust input feature representation learning techniques, including the VTS-MVN, the DSTC and the spectral masking, are thoroughly experimented with and discussed. Following that, the evaluation of techniques aiming at reducing noise variations in hidden representations, namely the IHM and the noise code, is presented. Finally, we conclude this chapter with a comparison between the performance we have achieved and those reported in the literature.

6.1 Datasets

6.1.1 The Aurora-2 Corpus

The benchmark noisy speech recognition dataset, Aurora-2 [58], consists of two sets of training data, one for clean training and the other for multi-style training. All the data is sampled at 8kHz. Each of them comprises 8440 utterances, and is equally split into 20 subsets. For the multi-style training data, all the utterances in the same subset share the same noise condition and there are totally 4 different noise scenarios (train, babble, car and exhibition hall) at 5 different SNRs (20dB, 15dB, 10dB, 5dB and clean). All three test sets, `A`, `B` and `C`, are used for evaluation. Set `A` has the same noise types as the multi-style training data and set `B` has four new noise types, namely restaurant, street, airport and train station. For set `C`, there are only two noise scenarios (train and street) but with additional channel distortions. For all three test sets, a total of 6 different SNRs are used for evaluation purposes, which have one additional 0dB compared to the training set. A summary of the Aurora-2 corpus is presented in Table 6.1.

Standard complex back-end GMM-HMM systems are built separately for the clean and multi-style training data using utterance-based CMVN normalized MFCC features

Table 6.1: A summary of the Aurora-2 corpus.

	Train		Test		
	`cleantr`	`multitr`	`A`	`B`	`C`
# of Utterances	8440	8440	28*1001	28*1001	14*1001
Duration (hours)	4.13	4.13	13.81	13.81	6.90
# of Environments	1	20	28	28	14
# of Noise Types	0	4	4	4	2
SRN Range (dB)	-	$5 \sim 20$	$0 \sim 20$	$0 \sim 20$	$0 \sim 20$
# of Speakers	110	110	104	104	104

by maximizing the training data likelihood. The 16-state word-based HMM and the 5-state silence model are adopted, leading to a total of 181 HMM states. These GMM-HMM systems are used to generate the per frame DNN training labels. No language model is used for this task and an equal probability digit-loop is adopted for decoding only. The open source `Kaldi` toolkit [183] is used.

6.1.2 The Aurora-4 Corpus

Aurora-4 [117] is a medium vocabulary noisy speech recognition task based on the WSJ corpus [184]. Each utterance is recorded at 16kHz and down-sampled to 8kHz to simulate the telephone channel. In our experiments, only the original 16kHz data is used. Similarly, two training sets, one with only the clean speech and the other with multi-style speech, are used. Each of the two training sets consists of 7138 utterances. For the multi-style training data, one half of the utterances are recorded using the primary Sennheiser microphone and the other half are recorded using one of 18 different secondary microphones. Both halves include a combination of clean speech and speech corrupted by one of six different noise types (street traffic, train station, car, babble, restaurant, airport) at 10dB $\sim$ 20dB SNRs. The evaluation set is derived from the WSJ0 5K closed vocabulary task which consists of 330 utterances from 8 speakers. This test set is recorded by the primary microphone and a secondary microphone. These two sets are then each corrupted by the same six noise types used in the training at 5dB $\sim$ 15dB SNRs, creating a total of 14 test sets. The types of noise are common across the training and testing data, but the SNRs differ. To ease system comparisons, these 14 test sets are further grouped into four broad sets: clean, noisy, clean with channel distortions, noisy with channel distortions, which are referred to as set `A`, `B`, `C` and `D` respectively. A summary of the Aurora-4 corpus is listed in Table 6.2.

Two context-dependent GMM-HMM systems are trained using maximum likelihood estimation on the two training sets and they have 3358 and 3257 senones respectively. The input features are 39D MFCC features including static, first and second order

Table 6.2: A summary of the Aurora-4 corpus.

	Train		Test			
	`cleantr`	`multitr`	`A`	`B`	`C`	`D`
# of Utterances	7138	7138	330	6*330	330	6*330
Duration (hours)	15.15	15.15	0.67	4.02	0.67	4.02
# of Environments	1	14	1	6	1	6
# of Noise Types	0	6	0	6	1	6
SRN Range (dB)	-	$10 \sim 20$	0	$5 \sim 15$	0	$5 \sim 15$
# of Speakers	83	83	8	8	8	8

delta features. Utterance-based CMVN is performed. These models are used to align the corresponding training data to create senone labels for training the DNN-HMM systems. Decoding is performed with the standard WSJ bi-gram language model. The open source `Kaldi` toolkit [183] is used.

6.2 Noise-Robust Input Representations

6.2.1 VTS-MVN

In this section we justify the effectiveness of our proposed VTS-MVN technique for improving the DNN AM's noise robustness on the Aurora-2 task. The recognition performance is evaluated using the WER criterion. Due to the large number of subsets involved in the evaluation, we report the average WER on each broad set for comparisons.

Clean Training

The VTS-MVN is first evaluated on the clean trained models with MFCC features. The 39D MFCC features consist of 13D cepstral features projected down from 26 FBanks, 13D delta and 13D accelerator parameters. The 0th cepstral coefficient is used instead of the log energy. Both a GMM AM and an 8-hidden-layer (8H) DNN AM are trained. The DNN AM uses a context window of 9 frames and 512D hidden layers. The DNN configuration is chosen based on our initial experiments presented in Section 3.2.2. The global MVN is used for the baseline DNN. The performance of these two baseline systems are tabulated in Table 6.3 in the row "-" and "Global-MVN" respectively. From the results, we can see that both the clean GMM and DNN dramatically degrade in noisy conditions.

A 2048-component GMM is estimated for the feature-based VTS compensation ("VTS(ftr)" in Table 6.3) and 4 iterations of noise estimations are conducted for the model-based VTS compensation ("VTS(mdl)" in Table 6.3). The WERs on all the three

Table 6.3: WER (%) performance of VTS-MVN on clean trained models with MFCC features on Aurora-2.

System		Clean	Test Set			
			A	B	C	Avg.
GMM	-	0.4	39.1	40.0	39.1	39.5
	VTS(ftr)	0.4	14.1	13.2	14.8	13.9
	VTS(mdl)	0.4	**8.9**	**8.4**	**9.7**	**8.8**
DNN	Global-MVN	0.3	39.3	40.2	37.3	39.2
	UTT-MVN	0.2	15.2	12.6	14.5	14.0
	VTS(ftr)	0.2	**10.4**	**9.4**	**10.6**	**10.0**
	VTS-MVN	0.2	15.7	13.6	15.8	14.9

test sets are greatly reduced, especially with model-based VTS, from 39.5% to 8.8%. The feature-based VTS is directly applicable to the DNN, which yields an average WER of 10.0%. Using the proposed VTS-MVN with borrowed distortion parameters from the GMM-HMM system reduces the baseline WER from 39.2% to 14.9%. Although it is not as effective as the VTS on GMMs, the single Gaussian-based VTS-MVN does reduce the DNN baseline WER by more than half. One probable explanation is that with thousands of Gaussians in the GMM system, the VTS compensation is more effective. For comparison, we also train a DNN using the utterance-based MVN and the results are listed in the row “UTT-MVN”. Our VTS-MVN performs slightly worse than the simple UTT-MVN. This may be due to the different normalization techniques used for training and testing. Namely, the DNN model is trained with data processed using UTT-MVN; but the testing data is normalized using VTS-MVN.

Multi-style Training

Next we justify its effectiveness on the multi-style trained models. Except for the multi-style training data, all the configurations are the same. The recognition performance of the two baselines are listed in the Table 6.4. The baseline DNN system has a relative 31.5% error reduction over the GMM baseline system, clearly indicating its better acoustic modeling capability. Comparing among the three test sets, the DNN performs much better on data with seen noise (Set `A`) and degrades on Set `B` with unknown noise and on Set `C` with additional channel distortions.

Although the VTS compensation works with the clean speech model assumption, it still performs well for the multi-style models. It may imply that VTS is not restricted to additive noise and channel distortions but also addresses the more general data mismatch problem between training and testing. “VTS(mdl)” consistently outperforms “VTS(ftr)” on the GMM-HMM system; however for the DNN, our simple VTS-MVN is still worse than the “VTS(ftr)”, 7.0% *vs.* 6.7%.

Table 6.4: WER (%) performance of VTS-MVN on multi-style trained models with both MFCC and FBank features on Aurora-2.

System		Clean	Test Set			
			A	B	C	Avg.
GMM MFCC	-	0.6	12.3	10.4	17.9	12.7
	VTS(ftr)	0.5	8.0	7.7	8.0	7.9
	VTS(mdl)	0.5	**7.0**	**6.9**	**7.2**	**7.0**
	NAT(ftr)	0.5	6.7	6.4	7.0	6.6
	NAT(mdl)	0.5	**6.5**	**6.1**	**6.8**	**6.4**
DNN MFCC	Global-MVN	0.4	6.4	8.5	13.7	8.7
	UTT-MVN	0.5	**5.1**	**6.3**	**5.8**	**5.7**
	VTS(ftr)	0.6	6.6	6.9	6.5	6.7
	VTS-MVN	0.3	6.7	6.8	8.3	7.0
	NAT(ftr)	0.7	6.5	7.5	6.8	6.9
	NAT-MVN	0.2	**4.7**	**5.7**	**5.3**	**5.2**
DNN FBank	Global-MVN	0.3	5.7	7.8	12.1	7.8
	UTT-MVN	0.3	**4.6**	**5.7**	**5.6**	**5.2**
	VTS(ftr)	0.7	7.3	7.8	7.9	7.6
	VTS-MVN	0.2	5.9	7.4	6.7	6.6
	NAT(ftr)	0.9	9.9	11.6	11.0	10.8
	NAT-MVN	0.2	**4.2**	**5.7**	**5.3**	**5.0**

After VTS compensation, the GMM renders a similar performance among the three test sets, while the WER of our approach differs greatly. This may be attributed to the fact that the distortion parameters are not directly optimized for the DNN. We then investigate the adaptive training for both the two systems. For the feature-based NAT, "NAT(ftr)", the canonical models are re-estimated on the pseudo-clean features after the distortion parameter estimation. From our experiments, one full iteration of re-training gives the best recognition performance for both the "NAT(ftr)" and the model-based NAT, "NAT(mdl)" (Table 6.4). After re-training the DNN with the VTS-MVN we could achieve an average WER of 5.2%, which is relatively 18.8% lower than the GMM NAT's 6.4% and 8.8% lower than the DNN utterance-based MVN's 5.7%. This suggests the superior modeling capability of DNNs could relieve the limitation of the single Gaussian based VTS-MVN. For the feature-based NAT, slight degradation over the "VTS(ftr)" has been observed, 6.9% *vs.* 6.7%. However the DNN frame accuracy on the training data does improve a lot. It may be explained by imperfect feature compensation that may discard potentially useful information and also bring in unwanted distortions.

With DNN AMs, FBank features have been found to outperform MFCCs [21]. We hence further evaluate our VTS-MVN using the 40D FBank features together with the log energy, delta and accelerator parameters. Nine contextual frames are employed and a total of 8 hidden layers are trained. Due to the much higher input feature dimension, 123D *vs.* 39D, the size of each hidden layer is set to 1024 instead of 512, which is used for MFCC features. Using FBank features gives us the best baseline DNN ("Global-MVN" in Table 6.4). Due to the correlations among each FBank feature dimensions, which are not well modeled by the diagonal GMMs, the feature-based VTS compensation performs worse and degrades greatly in the "NAT(ftr)". With our NAT-MVN, a WER of 5.0% is achieved, which is a relatively 21.9% error reduction from the GMM-based NAT model.

Aurora-4

Aurora-2 is a simple task, we are hence interested in how this technique performs on a relative complex task, Aurora-4. Similarly, we first apply the VTS-MVN on the clean trained DNN using 39D MFCC features. Different from the Aurora-2 setup, 11 adjacent frames are used as the input and 2048 hidden units are used for each hidden layer due to the increased task complexity. We keep adding hidden layers until a degradation is observed. From Figure 6.1, 6 hidden layers give the best performance, which is hence used as our baseline DNN for the Aurora-4 clean training task.

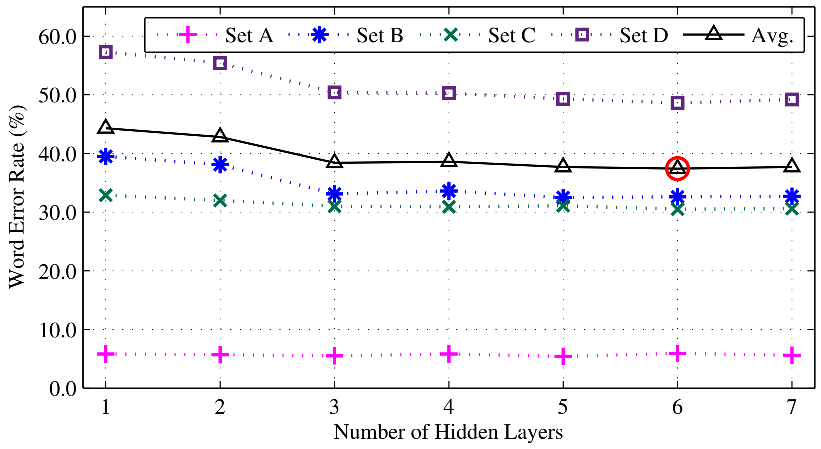


Figure 6.1: WER(%) performance of DNNs with different number of hidden layers using MFCC features on the Aurora-4 clean training task.

For comparisons, we also apply the model-based VTS compensation to the conventional GMM system. Experimental results in Table 6.5 show that our VTS-MVN improves the clean trained baseline DNN. But it is not as effective as the conventional VTS on GMMs.

Next, a conventional GMM using 39D MFCC features and a w9-2048D-6H DNN using 72D FBank features are trained with the multi-style training data of Aurora-4. Model-based VTS compensations are experimented with and results are listed in

Table 6.6. For the multi-style trained DNN, the simple UTT-MVN yields the best performance. It could probably due to the longer length of utterances in Aurora-4 compared to Aurora-2. The direct utterance-based MVN is already reliable enough for DNNs.

Table 6.5: WER (%) performance of VTS-MVN on clean trained models with MFCC features on Aurora-4.

System		Test Set				
		`A`	`B`	`C`	`D`	Avg.
GMM	-	7.3	41.9	40.8	59.7	47.0
	VTS(mdl)	7.3	**15.8**	**14.7**	**24.0**	**18.6**
DNN	UTT-MVN	5.9	32.6	30.5	48.6	37.4
	VTS-MVN	**5.6**	27.0	17.3	40.3	30.5

Table 6.6: WER (%) performance of VTS-MVN on multi-style trained models on Aurora-4.

System		Test Set				
		`A`	`B`	`C`	`D`	Avg.
GMM MFCC	-	12.4	19.9	28.2	36.1	26.9
	VTS(mdl)	13.5	16.9	19.3	23.9	19.8
DNN FBank	UTT-MVN	5.0	**8.8**	**9.0**	**20.1**	**13.4**
	VTS-MVN	5.8	12.8	32.2	27.7	20.0
	VTS-NAT	**4.8**	9.3	10.7	22.8	14.9

Discussions

The simple utterance-based MVN is effective in removing noise variations for improved robustness. The proposed VTS-MVN could further improve DNNs' performance on short utterances when adaptive training is adopted to address the potential mismatch caused by different normalization used in training (*i.e.* UTT-MVN) and testing (*i.e.* VTS-MVN). The VTS-MVN is more efficient than GMM-based VTS as only one single Gaussian is involved per utterance. On Aurora-2, the NAT-MVN could yield a relatively 18.8% WER reduction over the GMM-based NAT system. Moreover, using FBank features, we could achieve a relatively 21.9% improvement against the GMM NAT system. However, this approach is not as effective as VTS on clean trained speech models. One possible improvement would be to estimate the environment distortion parameters directly from the DNN instead of borrowing them from the GMM. Furthermore, this approach assumes the direct utterance-based MVN estimation is unreliable, which is true for short utterances such as those in Aurora-2. When we apply this technique on Aurora-4, no gain could be obtained. Using the simple utterance-based MVN is sufficient for long utterances.

6.2.2 DSTC

In this section, we study how to effectively model long span of speech signals using DNNs on Aurora-2. With longer input information, a better environment estimation could be obtained automatically in DNN AMs, which leads to improved generalization performance on unseen conditions. Only the multi-style training data is used in this study. The 40D FBank coefficients and the energy term together with their delta and accelerator parameters are adopted. A baseline DNN with a context window of 9 frames (w9) is trained. We keep adding 1024D hidden layers until degradations are observed. From Figure 6.2, the first degradation happens when 7 hidden layers are used. Although using 8 hidden layers is the best configuration among all the DNNs we experimented with, the performance improvement is relatively small compared to the increased number of parameters. We hence decide to use the w9-1024D-6H DNN as our baseline system, which has the average performance of 5.3% WER over all the test sets.

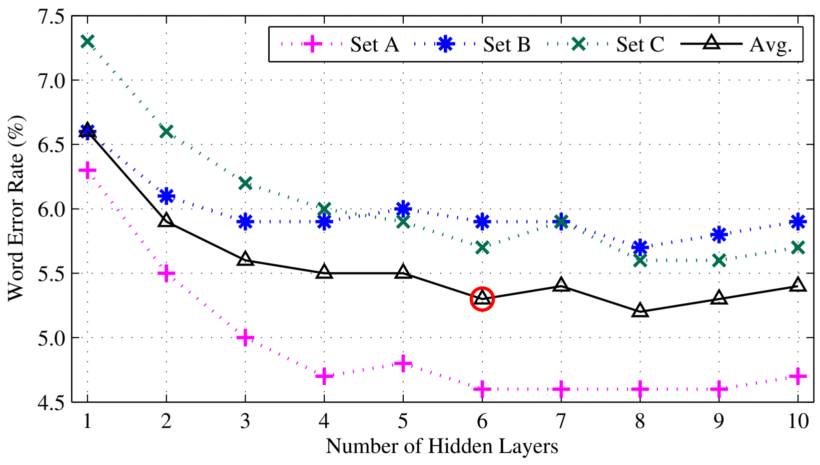


Figure 6.2: WER(%) performance of DNNs with different number of hidden layers using 40D FBank features on the Aurora-2 multi-style training task.

Structure Comparisons

NNs are inherently capable of modeling acoustic contexts. However, the training of shallow NNs can easily get over-fitted if the data is not sufficient, which is shown by the dramatic increase in WERs (Table 6.7) of the 1H shallow NNs (Figure 4.3a) when doubling the contexts from 4 frames on each side to 8 frames, *i.e.* from w9 to w17 input. All the w17-1H NNs perform worse than the w9-1H NN.

One way to increase the input context and the model capacity while maintaining a reasonable model size is to explore the potential independence structures in the wide context and model them separately with smaller NNs. The original STC system (Figure 4.3b) is first built with the existing w9-1024D-1H NN. It gives a WER of 7.0%, which is the lowest among all the existing w17 systems but still higher than the w9-1024D-1H NN. We believe the reason is the information loss in the early decisions made by each sub-context NNs where the posteriors are combined. Moreover, the model size is actually not reduced. With our proposed STC system (Figure 4.3c), we can achieve a WER of 6.1% which is a relative 7.6% improvement over the w9-1024D-1H NN.

Another way is to use many layers of nonlinear processing. As seen from the WERs of different 6H DNNs (Figure 4.4a) in Table 6.7, they are much more robust than shallow NNs and can further improve performance, but have the requirement of increasing the model capacity by using higher dimensional hidden layers. It can be attributed to the increased input variations caused by the expansion of the input context. The w17-3072D-6H DNN has the lowest average WER of 5.1% with 54.2 million parameters compared to the w9-1024D-6H DNN's 5.3% WER and 6.6 million parameters.

Table 6.7: WER (%) performance of multi-style trained NNs with different structures on Aurora-2.

System	Model Capacity	WER (%)			
		A	B	C	Avg.
1H NN (Figure 4.3a)	w9-1024D	**6.3**	**6.6**	**7.3**	**6.6**
	w17-1024D	6.8	7.1	8.3	7.2
	w17-2048D	7.1	7.9	8.8	7.8
	w17-3072D	7.5	8.9	9.7	8.5
1H STC (Figure 4.3b)	w17-2048D	6.6	6.8	8.0	7.0
1H STC (Figure 4.3c)	w17-2048D	**5.8**	**6.0**	**6.8**	**6.1**
6H DNN (Figure 4.4a)	w9-1024D	**4.6**	5.9	5.7	5.3
	w17-1024D	4.9	5.9	5.9	5.5
	w17-2048D	**4.6**	5.6	5.5	5.2
	w17-3072D	4.7	**5.3**	**5.3**	**5.1**
6H DSTC (Figure 4.4b & 4.4c)	w17-2048D M1	4.9	6.4	6.6	5.8
	w17-2048D M2	4.8	6.8	6.4	5.9
	w17-2048D M3	4.5	6.8	6.4	5.8
	w17-2048D M4	4.6	6.6	6.3	5.7
	w17-2048D M5	**4.4**	6.2	6.1	5.5
	w17-2048D M6	**4.4**	**5.6**	**5.6**	**5.1**

Next we use the w9-1024D-6H DNN as partial context DNNs to build our DSTC systems merging at different hidden layers (Figure 4.4b), which are effectively w17-2048D-6H systems. "Ml" indicates the merging occurs at the lth hidden layer. From

Table 6.7, it can be seen that the DSTC system merging at the last hidden layer performs the best. It indicates the importance of learning a better partial context feature representation over focusing too much on modeling the correlations. This DSTC that merges at the last hidden layer (Figure 4.4c) gives a WER of 5.1% with only 13.1 million parameters.

More Partial Context Blocks

To further explore the potential of the DSTC technique, a much wider context of 33 frames is used, which corresponds to 0.33 seconds of speech. Similarly, we build our DSTC systems by reusing the existing DNNs. With the w17-2048D-6H DNN, we could split the w33 input window into left and right partial contexts, *i.e.* 2 blocks. While using the w9-1024D-6H DNN, the input window is split into 4 blocks. For comparison purposes, we also build a single DNN modeling the whole w33 input directly, which can be seen as a 1-block system. From results in Table 6.8, we could achieve a WER of 4.8% with four w9-1024D-6H DNNs to model the 33 acoustic context frames. Although the improvement over the 2-block system is small, the 74.0% relative parameter reduction over the single DNN system is attractive. One probable explanation is that the DNN's model capability is more related to the number of hidden units rather than the number of connections. Furthermore, with eight w9-1024D-6H DNNs we build a w65-8192D-6H DSTC which gives the best 4.4% WER with 52.5 million parameters. It has a relative 12.0% WER reduction and 48.2% parameter reduction over the best single DNN, *i.e.* the "1-block" w33-4096D-6H system.

Table 6.8: WER (%) performance of DSTC systems with different number of partial contexts on Aurora-2.

System	Model Capacity	Model Size (million)	WER (%)			
			A	B	C	Avg.
1-block	w33-4096D	101.3	4.8	**5.1**	5.3	5.0
2-block	w33-4096D	51.3	4.4	**5.1**	5.2	**4.8**
4-block	w33-4096D	26.3	**4.2**	5.2	**5.1**	**4.8**
6-block	w49-6144D	39.4	**4.0**	4.9	4.9	4.5
8-block	w65-8192D	52.5	**4.0**	**4.7**	**4.8**	**4.4**

Additionally, to validate the independence assumption, we modify the 4-block DSTC system to overlap each block with its previous one by half of the partial context window. It gives an effective w34-6144D-6H DSTC system and achieves the same 4.8% WER as the 4-block system, which indicates that the explicit modeling of the partial context dependencies is unnecessary. This may be because the inherent sliding window processing of the hybrid system has already captured the dependencies.

Aurora-4

To further understand the effectiveness of the proposed DSTC system, we test it on the Aurora-4 multi-style training task. The same w11-2048D-6H DNN used in Section 6.2.1 is adopted as our baseline DNN. Only the best DSTC configuration found in previous experiments on Aurora-2 is evaluated and the results are tabulated in Table 6.9. With the 3-block DSTC system, we could obtain only slightly improvement over the baseline DNN. It can be attributed to the higher task complexity of Aurora-4 compared to Aurora-2. The increase of the context window not only provides more information for the target class, but also brings in even more noise variations.

Table 6.9: WER (%) performance of DSTC systems with different number of partial contexts on Aurora-4.

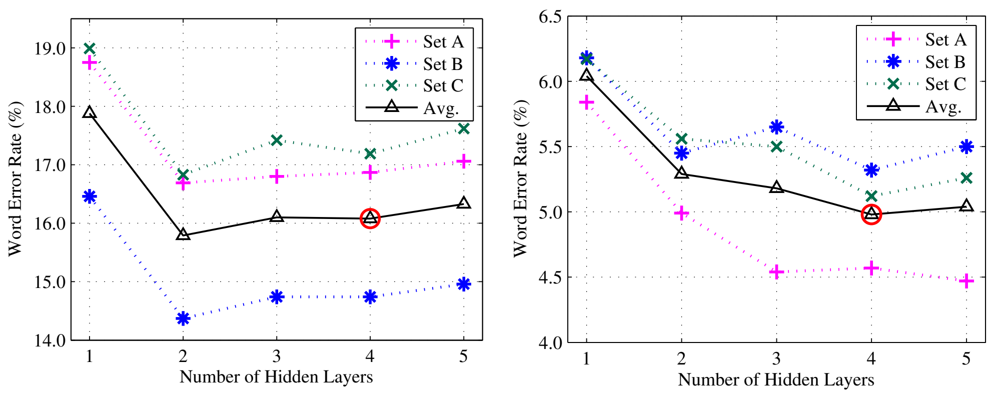
System	Model Capacity	Model Size (million)	WER (%)				
			`A`	`B`	`C`	`D`	Avg.
Baseline	w11-2048D	29.3	**5.0**	**8.8**	9.0	20.1	13.4
2-block	w22-4096D	35.9	5.2	9.0	**8.9**	20.3	13.6
3-block	w33-6144D	42.6	5.1	8.9	9.3	**19.6**	**13.2**
4-block	w44-8192D	49.3	5.1	9.2	9.4	20.2	13.6

Discussions

In this section, we have justified the effectiveness of our proposed Deep Split Temporal Context (DSTC) system to improve the generalization capability of DNNs for noise robust ASRs. The DSTC system uses multiple smaller DNNs to robustly model a long span of acoustic contexts of speech signals. Those partial context DNNs share the unsupervised pre-training phase which largely reduces the DSTC system training cost. Due to the independent modeling of each partial context, the whole DSTC system has fewer model parameters than a DNN with the same hidden capacity. On the Aurora-2 multi-style training task, our DSTC system outperforms the best single DNN by 12.0% WER (4.4% *vs.* 5.0%), as well as a 48.2% model parameter reduction. However, when we apply this technique on the Aurora-4 task, only small gains could be achieved. One probable explanation is that the DSTC assumes the expansion of the context window will bring more discriminative information for the target class than the distraction noise variations. However, on Aurora-4, the discrimination among thousands of states is much more challenging than the 181 states used for the Aurora-2 task. The dramatic increase of the input variations caused by the increased input context window size can only make the problem even harder.

6.2.3 Spectral Masking on Aurora-2

In this part, we investigate the biologically motivated spectral masking approach for DNN-based noise robust speech recognition. Unlike the VTS-MVN and the DSTC techniques, the spectral masking directly addresses noise variations in the power spectrum domain. In this study, the 24D FBank features, together with the delta and the accelerator parameters, are used as input feature representations. Utterance-based MVN is adopted for simplicity. A consecutive 11 frames of the acoustic features are concatenated as the input to the DNNs. Baseline DNN AMs on the Aurora-2 task have four 2048D hidden layers, which is decided based on results in Figure 6.3. IBMs are computed from the parallel training data using equation (4.18) with $LC = 0$. Due to the computation of IBMs, both the clean and multi-style training data are required. Except for that, all the model trainings only employ either the clean or multi-style training data.



(a) Clean training. (b) Multi-style training.

Figure 6.3: WER(%) performance of DNNs with different number of hidden layers using 24D FBank features on Aurora-2.

Mask Estimations

We first justify the effectiveness of spectral masking with ideal masks, including the IBM and the ideal state-dependent mask, and then compare the two mask estimation approaches we have proposed, namely the state-dependent and the DNN-based mask estimations. The results of applying these masks to the clean trained DNN AM are tabulated into the upper half of Table 6.10. The average 16.1% WER of the baseline is far from humans' expectations. Applying IBMs, we could obtain a 3.7% WER, clearly indicating the potential of spectral masking for improving DNNs' noise robustness. To first justify the effectiveness of the estimated IBM bases, we use the true reference to generate the state-level alignment. Based on that alignment, a set of ideal posterior vectors are constructed by setting the correct label state to have value 1 and all the others to 0 for each feature frame. With these ideal posteriors ("Ideal" in Table 6.10), the state-dependent IBM bases could reduce the WER to less than 1.0%. Although the ideal posteriors may bring additional information for recognition, the less than 1.0% WER does imply its potential. However, when the posteriors from the baseline are used for the state-dependent mask estimation ("State"), a rather small improvement could be obtained (from 16.1% to 14.6%). The quality of the posteriors is thus crucial to the effectiveness of this state-dependent mask. Using a DNN-based estimator, a 7.2% WER could be achieved. The gap between the IBM's performance and our estimators' performance is still quite large. Besides the accuracy of the mask estimators, another probable reason is the mismatch between the clean trained model and the masked features. In our study, the masked partial features are directly used without reconstruction, which may cause a mismatch problem. From the early visual inspection (Figure 4.7), the masked features are expected to be different from the clean ones.

Table 6.10: WER (%) performance of different masks for both the clean trained and multi-style trained DNN AMs on Aurora-2.

Style	Masking		WER (%)			
	Train	Test	A	B	C	Avg.
clean	-	-	16.9	14.7	17.2	16.1
		IBM	3.6	3.4	4.3	3.7
		Ideal	0.7	0.8	0.8	0.8
		State	15.0	13.6	15.7	14.6
		DNN	**5.9**	**8.4**	**7.3**	**7.2**
multi	-	-	**4.6**	**5.3**	**5.1**	**5.0**
		IBM	4.0	4.0	4.5	4.1
		Ideal	0.4	0.5	0.5	0.5
		State	5.1	6.8	6.3	6.0
		DNN	5.3	9.0	6.9	7.1
	IBM	IBM	1.1	1.0	1.2	1.1
	State	State	5.2	7.1	6.5	6.2
	DNN	DNN	**4.1**	**6.3**	**5.4**	**5.2**

One possible way of addressing the mismatch problem is the use of multi-style training data. In "multi" part of Table 6.10, without any masking, the DNN baseline already has a 5.0% WER. It suggests the importance of data samples from target environments in achieving good generalization capability for DNNs. Ideal masks could further reduce WERs. With estimated masks, we could achieve better performance than the clean system but worse performance than the "multi" baseline. This suggests that the variations of multi-style data improve DNNs' robustness to masked features but there are still mismatches. Retraining the DNN with masked multi-style data would be the best choice. From the last three lines of results in Table 6.10, the IBM could yield around 1% WER for all three test sets. The DNN estimator improves from 7.1% to 5.2%, however, it is still a little worse than the baseline's 5.0%. This has to be attributed to the quality of the estimated masks.

From this study, the ideal masks suggest great potential but the estimated masks cannot even outperform the multi-style baseline DNN. Comparing the two mask estimations, the DNN-based one is consistently better. We will hence focus our study on the DNN-based mask estimator for the multi-style DNN. Comparing the baseline "multi" DNN and the one retrained on features filtered by DNN predicted masks, we could observe a rather different WER breakdown on each test set. The DNN mask reduces the WER on set `A` from 4.6% to 4.1%, indicating its effectiveness for known noise. However, for unseen noise in set `B`, the performance degrades from 5.3% to 6.3%, implying that the DNN mask estimator does not generalize well to unseen noise. This also suggests that using no mask is more preferable than using unreliable ones for the hybrid DNN AM. Similarly, the performance degrades on set `C` due to the unseen noise and additional channel distortions. This indicates that the spectral masking is effective but reliable masks have to be estimated.

RBM-DNN *vs.* DNN

To address the mismatch problem in the DNN ME through adaptation, a RBM-DNN is proposed. Prior to adapting the ME, it is interesting to understand the effect of using an RBM input layer. We hence experiment with different combinations of generative ("gen") and discriminative ("dis") depths in the AM DNN. We use the term "generative" only to indicate the layers are trained in a generative manner. Experimental results are tabulated in Table 6.11 with the first row as the baseline DNN system. With the same number of hidden layers, the RBM-DNN with only 1 RBM performs the best. It has lower WERs on all the 3 sets than the standard DNN. While keeping the same number of discriminative layers, adding only one RBM input layer is the best. As the RBM-DNN is both faster (1 less layer for fine-tuning) and more robust than the DNN, we hence take the RBM-DNN with 1 generatively trained RBM layer and 3 discriminatively tuned DNN layers as our new baseline.

Spectral Masking using RBM-DNN

Next, we revise the spectral masking system to use the RBM-DNN instead of the standard DNN for both the mask estimation and the acoustic modeling. Moreover, this RBM input layer is shared between these two RBM-DNNs. For easy reference, we denote the baseline RBM-DNN AM as system "$\mathcal{A}$". The masked features are firstly decoded with the baseline RBM-DNN AM (*i.e.* system "$\mathcal{B}$" in Table 6.12). The

Table 6.11: WER (%) performance of different RBM-DNN configurations on Aurora-2.

# of Hidden Layers			Test Set			Avg.
Total	gen	dis	A	B	C	
4	0	4	4.6	5.3	5.1	5.0
4	1	3	**4.5**	**5.1**	**5.0**	**4.9**
	2	2	4.9	5.3	5.2	5.1
	3	1	5.7	5.6	5.8	5.7
	4	0	7.4	6.8	7.6	7.2
5	0	5	4.5	5.5	5.3	5.0
	1	**4**	4.7	**5.1**	5.1	**4.9**
6	0	6	4.6	5.4	5.2	5.0
	2	**4**	4.7	5.2	5.1	5.0
7	0	7	4.5	5.5	5.2	5.0
	3	**4**	5.1	5.4	5.4	5.3

mismatches between the noisy features and the masked partial features lead to an increase in the WER, from 4.9% to 6.9%. After retraining the RBM-DNN AM with the masked training data, *i.e.* system "$\mathcal{C}$" in Table 6.12, the performance is improved to 5.2% WER. However, it is still worse than system "$\mathcal{A}$". From the detailed WER reductions of system "$\mathcal{C}$" compared to "$\mathcal{A}$" in Figure 6.4, masking helps in reducing WERs in matched conditions, and degrades in all the unknown conditions. For most of the matched noise types, the masking system "$\mathcal{C}$" has larger improvements on lower WERs. For speech-like babble noise (A2 in Figure 6.4), our masking system also fails.

Table 6.12: WER (%) performance of RBM-DNN based spectral masking system on Aurora-2.

System	Masking		Test Set			Avg.
	Train	Test	A	B	C	
$\mathcal{A}$	×	×	4.5	**5.1**	**5.0**	**4.9**
$\mathcal{B}$	×	✓	5.2	8.6	6.9	6.9
$\mathcal{C}$	✓	✓	**3.9**	6.3	5.4	5.2

Acoustic Model Adaptation with LINs

To address the mismatch problem, we first investigate the effectiveness of LINs for AMs. An initial decoding is required to generate adaptation hypotheses. One LIN is estimated for each noise condition. For Aurora-2, each test set contains 1001 utterances from the same 104 speakers. It counts up to around half an hour of speech data. All the utterances in one test set share the same noise condition and SNR. Hence, the estimated LIN transforms are noise- and SNR-dependent and speaker-independent. The LINs are initialized to be identity and estimated using EBP. To avoid over-fitting, 10% of the

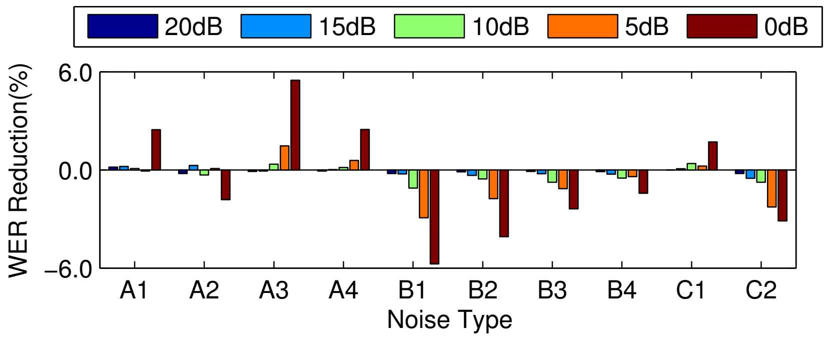


Figure 6.4: WER reductions of system "$\mathcal{C}$" from system "$\mathcal{A}$" on Aurora-2.

adaptation data is used for cross-validation. Experimental results in Table 6.13 show that the LIN adaptation could slightly improve the baseline system "$\mathcal{A}$" on both set `B` and set `C`, but slight degradation on set `A` has been observed (from 4.5% to 4.6%). From Figure 6.5, LINs could hardly give improvements on matched conditions as the AM RBM-DNN has already captured those variations from the training data. LINs are sensitive to the hypothesis errors as they degraded dramatically for speech at 0dB in set `A2` and `A3`. On unseen noise types, LINs are effective in improving performance by minimizing mismatches.

Table 6.13: WER (%) performance of RBM-DNN AM adaptation with LINs on Aurora-2.

System	LIN	Test Set			Avg.
		`A`	`B`	`C`	
$\mathcal{A}$	×	**4.5**	5.1	5.0	4.9
$\mathcal{A}$+LIN	✓	4.6	**5.0**	**4.9**	**4.8**

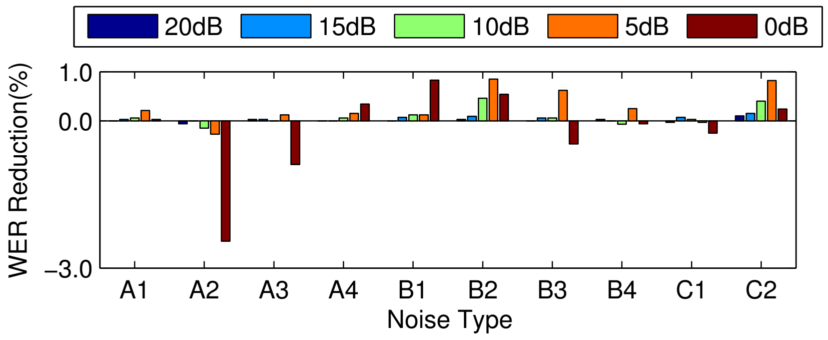


Figure 6.5: WER reductions of system "$\mathcal{A}$+LIN" from system "$\mathcal{A}$" on Aurora-2.

Mask Estimator Adaptation with Generative LINs

To adapt the ME, we firstly justify the effectiveness of the generatively trained LIN. From Table 6.14, the LIN increases the mask estimation errors in terms of MSE. But it does not affect the recognition performance much. One explanation is that the LINs are estimated for data reconstruction rather than mask or phoneme predictions. The LIN may only have captured some generic mismatches, which are not useful to both the mask estimation and the phoneme recognition.

Table 6.14: WER (%) and MSE performance of ME adaptation using generative LINs on Aurora-2.

Task	LIN	Test Set			Avg.
		`A`	`B`	`C`	
MSE	×	**5.49**	**6.95**	**5.97**	**6.17**
	✓	5.52	7.04	6.01	6.23
WER(%)	×	**3.9**	**6.3**	**5.4**	**5.2**
	✓	4.0	**6.3**	**5.4**	**5.2**

Mask Estimator Adaptation using LIN Sharing

Next, we investigate the applicability of borrowing transforms from AMs to MEs. Both the proposed RBM-DNN and the standard DNN are studied. LINs are estimated using the corresponding AMs and then directly applied to MEs. Both mask estimation and speech recognition performance are reported in Table 6.15. For mask estimation, sharing LINs for the DNN-based system slightly degrades performance. For our RBM-DNN, the LIN improves mask estimation on both set `B` and `C`. Slight degradation on set `A` is observed. To justify how these changes in masks affect recognition performance, we decode the masked features generated from those adapted MEs. For the DNN system, the mask estimation degradation increases recognition errors. Conversely, large improvements have been seen for our ME RBM-DNN by borrowing LINs. The average 5.2% WER of the spectral masking system "$\mathcal{C}$" (Table 6.12) is reduced to 4.9% (the last row in Table 6.15, which will be referred to as system "$\mathcal{D}$"). Although the adapted ME do not perform well on set `A`, the degradation is smaller than the improvements on set `B` and `C`. From the detailed WER reductions in Figure 6.6, the degradation on set `A` mainly happens on the lowest SNR. It is probably due to errors in the hypotheses for LIN estimations. The shared LINs do address the mismatch problem by giving clear improvements on all the cases in set `B` and `C`. Hence we may say that our RBM-DNN is not only more robust than the standard DNN, but also more reliable in sharing transforms.

Table 6.15: WER (%) and MSE performance of ME adaptation using LIN sharing on Aurora-2.

Task	Model	LIN	Test Set			Avg.
			A	B	C	
MSE	DNN	×	5.58	6.97	6.05	6.23
		✓	5.67	6.95	6.02	6.25
	RBM-DNN	×	**5.49**	6.95	5.97	6.17
		✓	5.58	**6.89**	**5.88**	**6.16**
WER(%)	DNN	×	4.6	**5.3**	5.1	5.0
		✓	4.3	6.0	5.3	5.2
	RBM-DNN	×	**3.9**	6.3	5.4	5.2
		✓	4.1	5.7	**5.0**	**4.9**

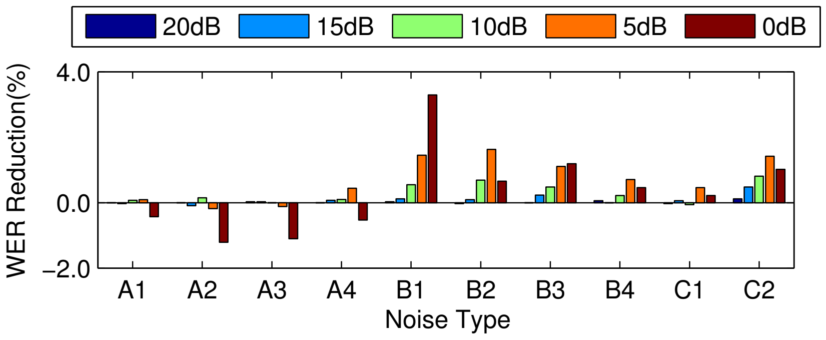


Figure 6.6: WER reductions of system "$\mathcal{D}$" from system "$\mathcal{C}$" on Aurora-2.

AM Adaptation with Spectral Masking

To address the potential mismatches in the masked feature domain, we further adapt the AM of the system "$\mathcal{D}$" with another set of LINs. It will be referred to as system "$\mathcal{D}$+LIN". The results are listed in Table 6.16 and the detailed WER reductions are illustrated in Figure 6.7. The best average 4.7% WER is achieved which is also better than the system "$\mathcal{A}$+LIN". Most of the gains come from lower SNRs. Comparing system "$\mathcal{A}$+LIN" (Figure 6.5) and system "$\mathcal{D}$+LIN" (Figure 6.7), the use of masking enables a more effective LIN adaptation, especially for low SNR noisy speech.

Table 6.16: WER (%) performance of spectral masking with LIN adaptations on Aurora-2.

System	LIN		Test Set			Avg.
	ME	AM	A	B	C	
$\mathcal{C}$	×	×	**3.9**	6.3	5.4	5.2
$\mathcal{D}$	✓	×	4.1	5.7	5.0	4.9
$\mathcal{D}$+LIN	✓	✓	4.1	**5.3**	**4.8**	**4.7**

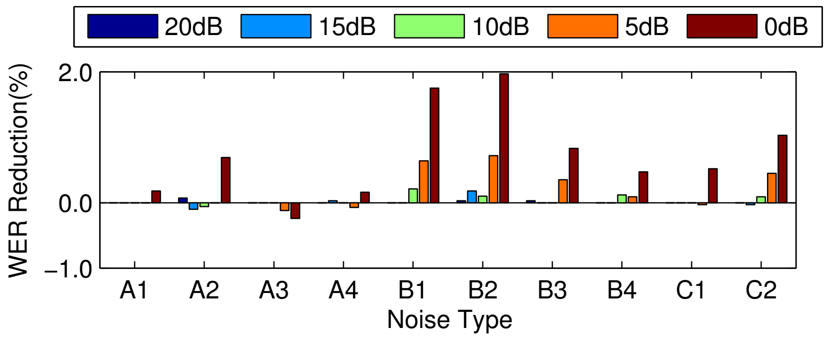


Figure 6.7: WER reductions of system "$\mathcal{D}$+LIN" from system "$\mathcal{D}$" on Aurora-2.

Constraining the LIN Transforms

Adaptation with LIN has improved system performance consistently. But the errors in the supervision hypotheses for LIN estimations have always been causing degradation on set `A`. To improve the adaptation robustness against supervision errors, we propose to reduce the number of parameters by constraining the LINs. This leads to the block diagonal LIN, "LIN(blk)", and the shared block diagonal LIN, "LIN(shd)". They are firstly evaluated in the system "$\mathcal{A}$+LIN". Results in Table 6.17 show that the constraints fail to result in any improvement. Despite this, we still borrow those transforms for our ME. Results in Table 6.17 for the system "$\mathcal{D}$" show that these constraints improve our masking system. The gains may come from the ME's high sensitivity to mismatches. For the ME, each sigmoid output is independent of the others, while for the AM, shifts in the final prediction could probably be normalized away due to the softmax nonlinearity. Moreover, by further adapting the AM in our masking system with constrained LINs, the best average WER 4.6% is achieved (the system "$\mathcal{D}$+LIN" in Table 6.17).

Table 6.17: WER (%) performance of LINs with different structure constraints on Aurora-2.

System	LIN			Test Set			Avg.
	ME	AM	Type	`A`	`B`	`C`	
$\mathcal{A}$+LIN	–	✓	full	4.6	**5.0**	4.9	4.8
			blk	4.8	5.1	4.9	4.9
			shd	4.9	5.1	4.9	5.0
$\mathcal{D}$	✓	×	full	4.1	5.7	5.0	4.9
			blk	**3.9**	5.5	4.9	4.7
			shd	**3.9**	5.6	4.9	4.8
$\mathcal{D}$+LIN	✓	✓	full	4.1	5.3	4.8	4.7
			blk	**3.9**	5.2	**4.7**	**4.6**
			shd	**3.9**	5.2	**4.7**	**4.6**

Posterior Interpolation

Comparing the WER breakdowns of the conventional system "$\mathcal{A}$+LIN(blk)", *i.e.* without masking, and the proposed masking system "$\mathcal{D}$+LIN(blk)" in Table 6.17, some performance complementariness is observable. System "$\mathcal{A}$+LIN(blk)" performs the best on set `B`, while system "$\mathcal{D}$+LIN(blk)" has the best performance on set `A` and `C`. In this experiment, we simply average the posteriors generated from these two systems and an average 4.3% WER is achieved. No further gain could be achieved by tuning the posterior interpolation weight from 0.0 to 1.0 by 0.1 each time.

6.2.4 Spectral Masking on Aurora-4

In view of the success on Aurora-2, we further justify the effectiveness of the spectral masking approach on a larger dataset, Aurora-4. The IBMs are computed using parallel training data with the threshold of $LC = -6$dB. Baseline DNNs are trained using 24D FBank features together with the delta and accelerator parameters. Utterance-based MVN is adopted. A context window of 11 adjacent frames is used as the DNN input. The DNN with 6 2048D hidden layers yields the best 13.8% WER (*cf.* Section 6.2.1). With two additional iterations of re-alignment and re-training, we could further reduce the average WER to 13.4%. No further improvement could be obtained by doing more iterations.

Mask Estimations

Different masks are first evaluated on the clean trained DNN-HMM system. As the state-dependent mask estimator does not perform well on Aurora-2, we hence experiment only with the DNN-based one. Results are reported in Table 6.18. Unlike on Aurora-2, the IBM only yields slight improvement (from 29.2% to 26.2%) and is outperformed by the DNN-based mask estimation. One probable explanation is that binary masks may introduce more variations than soft masks used in our DNN-based masking. Next the multi-style trained system "multi" is used to evaluate these masks. Comparing the "mutli" baseline to the "clean" one, it performs better under noisy conditions but degrades the performance on clean data, which may be due to the high complexity of the Aurora-4 task (more than 3000 senones *vs.* 181 states on Aurora-2 for discrimination). Directly decoding masked features gives slightly worse performance, especially for the IBM. It could be attributed to the fact that masked features are more similar to clean features rather than noisy ones, which leads to large mismatches between the model and the feature. To address this problem, retraining the DNN with masked features yields improved performance. However, for the DNN-based mask estimator, the retrained system has a WER of 13.6% and is still higher than the multi-style baseline's

13.4%, which is similar to what we have observed on Aurora-2.

Table 6.18: WER (%) performance of different masking algorithms on Aurora-4.

Style	Masking		WER (%)				
	Train	Test	A	B	C	D	Avg.
clean	-	-	**4.1**	22.7	21.7	41.1	29.2
		IBM	4.1	20.0	21.7	36.9	26.2
		DNN	**4.1**	**14.3**	**21.1**	**34.8**	**22.8**
multi	-	-	5.0	**8.8**	9.0	**20.1**	**13.4**
		IBM	5.0	19.0	9.0	24.1	19.5
		DNN	5.1	12.4	10.4	26.0	17.6
	IBM	IBM	4.9	6.5	8.0	12.2	8.9
	DNN	DNN	**4.7**	9.3	**8.4**	20.3	13.6

RBM-DNN *vs.* DNN

Similarly, we justify the effect of using RBM-DNN *vs.* DNN on Aurora-4 first. The results are listed in Table 6.19. The two RBM-DNN systems that have one RBM front-end perform the best, 13.2% and 13.1%. This further verifies that adopting the generatively trained RBM front-end is helpful but having too many RBMs also degrades the performance. The RBM-DNN with 1 RBM layer and 6 discriminatively tuned DNN layers is then used as our new baseline system on Aurora-4.

Table 6.19: WER (%) performance of different RBM-DNN setups on Aurora-4.

# of Hidden Layers			Test Set				Avg.
Total	gen	dis	A	B	C	D	
6	0	**6**	5.0	8.8	9.0	20.1	13.4
	1	5	**4.9**	**8.6**	9.1	19.8	13.2
	2	4	5.6	9.0	10.3	20.1	13.6
	3	3	6.5	9.7	11.5	21.1	14.5
7	0	7	5.0	8.8	**8.8**	20.1	13.3
	1	**6**	5.1	**8.6**	9.6	**19.4**	**13.1**
8	0	8	**4.9**	8.7	8.9	20.3	13.4
	2	**6**	5.4	8.9	9.7	19.8	13.4

Acoustic Model Adaptation

First, we evaluate the performance of different LIN adaptations on this AM RBM-DNN. One LIN transform is estimated for each test set using EBP with recognition hypotheses. Each set has 330 utterances, corresponding to 40 minutes of speech. 10% of these are used for cross validation to avoid over-fitting. The utterances come from 8 different speakers. A slight difference from Aurora-2 is that they have different SNRs. Hence, the estimated test set dependent LINs capture only the noise mismatches and are speaker- and SNR-independent. Results in Table 6.20 show that all the LINs are effective in reducing WERs on all the test sets, even including the clean set `A`. This could be attributed to the multi-style training data. Compared to Aurora-2, the acoustic modeling complexity is much higher for this task. The AM cannot maintain both a superior clean performance and a better generalization on noisy speech. Degradation on the clean speech of the multi-style model is hence expected. The LIN transforms seem capable of fixing this problem. The largest relative improvement, 14.6% (from 9.6% to 8.2%) is obtained on set `C`. It clearly suggests the effectiveness of LINs in addressing the channel mismatch. Although for the best LIN(shd), the absolute gain on set `D` (from 19.4% to 18.2%) is much larger than that on set `B` (from 8.6% to 7.9%). The relative improvement is almost the same, 8.3% on set `D` *vs.* 8.2% on set `B`.

Table 6.20: WER (%) performance of AM adaptation with different LINs on Aurora-4.

System	LIN	Test Set				Avg.
		`A`	`B`	`C`	`D`	
$\mathcal{A}$	×	5.1	8.6	9.6	19.4	13.1
$\mathcal{A}$+LIN	full	4.8	8.1	**8.2**	18.7	12.4
	blk	**4.6**	8.0	8.3	18.3	12.2
	shd	**4.6**	**7.9**	**8.2**	**18.2**	**12.1**

Spectral Masking with LIN Adaptation

In this experiment, we justify our proposed spectral masking system on Aurora-4. Firstly the direct use of ME RBM-DNN degrades the performance from 13.1% (system "$\mathcal{A}$" in Table 6.20) to 13.5% (system "$\mathcal{C}$" in Table 6.21). However, our masking system does give improvements on set `A` (from 5.1% to 4.7%) and `C` (from 9.6% to 8.7%). It is interesting to see improvements on speech with only channel distortions as the masking is defined to remove additive noise. One probable explanation is that the scaling of the component-wise soft-masking is effectively doing a mean and variance normalization in the power spectrum domain. For set `B` and `D`, unreliable mask estimation is probably the reason for degradation. Borrowing LIN transforms from the AM to the ME, *i.e.* in system "$\mathcal{D}$", could only bring the performance back to the baseline performance (13.1%). But the different WER breakdowns may imply a difference between them. It is also similar to what we have observed on Aurora-2. By further adapting the AM of system "$\mathcal{D}$", which leads to system "$\mathcal{D}$+LIN", we finally achieve the best average WER of 11.8% (LIN(blk)) with spectral masking. Comparing the "$\mathcal{A}$+LIN" in Table 6.20 and our "$\mathcal{D}$+LIN", the WER reductions are relatively small. However, the differences in the generated hypotheses are statistically significant [185]. For LIN and LIN(blk), the p-values are all smaller than 0.001 and for LIN(shd), it is 0.018. These suggest large differences in the recognition hypotheses between these two systems despite the similar average WER performance.

Table 6.21: WER (%) performance of spectral masking with different LIN adaptations on Aurora-4.

System	LIN			Test Set				Avg.
	ME	AM	Type	A	B	C	D	
$\mathcal{C}$	×	×	-	4.7	9.2	8.7	20.2	13.5
$\mathcal{D}$	✓	×	full	4.7	9.0	8.5	20.0	13.4
			blk	4.6	9.1	8.5	19.8	13.3
			shd	4.6	8.9	8.5	19.6	13.1
$\mathcal{D}$+LIN	✓	✓	full	4.7	8.1	**7.3**	18.1	12.1
			blk	4.5	**7.9**	7.5	**17.7**	**11.8**
			shd	**4.4**	8.0	7.6	**17.7**	11.9

Posterior Interpolation

To explore the differences between the system "$\mathcal{A}$+LIN" and "$\mathcal{D}$+LIN", we simply average the two sets of posteriors. Only the block-diagonal version of LIN is experimented with, and an average WER of 11.4% is achieved. The performance gains on almost all the test sets clearly indicate the complementariness between these two systems. Adjusting the interpolation weight from 0.0 to 1.0 by 0.1 could not give any further improvement. To the best of our knowledge, this 11.4% WER is currently the best reported performance on Aurora-4.

Utterance-based adaptation

Till now, we estimate the LIN from a set of adaptation data for all the experiments. Relaxing this condition is desirable for real world applications. In this experiment we justify the effectiveness of our proposed spectral masking system in an utterance-based adaptation scenario. One LIN will be estimated for each test utterance. The learning is exactly the same as previous ones except for the fact that no cross validation is used. The LIN is hence trained until no further training frame accuracy improvement could be achieved. Due to having rather limited data, only the LIN(shd) is evaluated. Results in Table 6.22 show that we can adapt system "$\mathcal{A}$" from WER of 13.1% to 13.0% with LIN(shd). To further reduce the model parameters, we keep only the diagonal elements of the LIN transform [168], which is referred to as "dig". With it, a slightly better AM adaptation performance (12.9%) could be achieved. Using LIN(dig) in our proposed

"$\mathcal{D}$+LIN" system, we could reduce the average WER to 12.3%. Similarly, the posterior averaging further reduces it to 12.1%. Compared to using only the AM adaptation ("$\mathcal{A}$+LIN"), the masking system is much more effective.

Table 6.22: WER (%) performance of utterance-based LIN adaptation on Aurora-4.

System	LIN Type	Test Set				Avg.
		A	B	C	D	
$\mathcal{A}$+LIN	shd	5.0	8.6	9.6	19.4	13.0
$\mathcal{A}$+LIN	dig	5.1	8.5	9.3	19.2	12.9
$\mathcal{D}$+LIN		**4.8**	8.2	**8.2**	18.3	12.3
PosterInter		5.1	**8.0**	8.8	**17.9**	**12.1**

6.3 Noise-Robust Hidden Representations

6.3.1 IHM

With the success of spectral masking in reducing noise variations in the spectral feature representations, we become interested in further extending the idea of masking into the DNN's hidden representations. Through the initial justifications on the noise robustness of those hidden representations, we do find the existence of noise variations and information redundancies. In this section, we hence evaluate the proposed hidden masking approach on the Aurora-4 task.

Comparisons with IBM and IRM

We compare our proposed IHM with the existing spectral masking techniques, namely the IBM and the IRM for noisy speech recognition on Aurora-4. Firstly, the ideal masks are only applied to the test sets, and evaluated with the baseline DNN. This is denoted as "E1" in Table 6.23. The real-valued IRM yields lower WERs than both the two binary masks due to the richness of its scaling-based masking. Our IHM largely outperforms the IBM, of which the large degradation comes from the mismatch between the masked features and the training data. To reduce this mismatch we retrain the baseline DNN with ideally masked training data, which is denoted as "E2" in Table 6.23. With retraining, all the three masks have achieved further WER reductions. The IRM still performs the best and our IHM and the IBM have similar WERs. The dramatic change in IBM performance from "E1" to "E2" further confirms the differences between the masked and the original features.

The investigations on ideal masks could tell us the potential of different masks in removing noise corruptions. However, for real applications, the lack of ideal masks poses a great challenge for all the masking techniques. The errors in the estimated

Table 6.23: WER (%) performance of different masks on Aurora-4.

System				Test Set				Avg.
Name	Train	Test	Type	A	B	C	D	
Baseline				5.0	8.8	9.0	20.0	13.4
E1	×	Ideal	IBM	5.1	19.1	19.1	27.1	21.5
			IRM	5.1	**5.7**	**7.4**	**8.3**	**6.9**
			IHM	**5.0**	5.9	8.0	11.2	8.2
E2	Ideal	Ideal	IBM	4.8	5.9	9.4	8.4	7.1
			IRM	**4.4**	**4.5**	**6.4**	**6.0**	**5.3**
			IHM	5.0	5.6	6.9	9.3	7.2
E3	Ideal	Est.	IBM	4.9	12.7	10.3	27.4	18.3
			IRM	**4.5**	10.2	9.3	24.6	15.9
			IHM	5.1	**9.3**	**8.9**	**20.4**	**13.7**
E4	Est.	Est.	IBM	**4.6**	9.3	8.4	21.5	14.1
			IRM	4.7	9.0	**8.2**	21.2	13.9
			IHM	4.9	**8.8**	8.8	**19.7**	**13.2**
E5	Est.	Est.	IBM	**4.7**	**8.3**	8.2	**19.0**	**12.6**
			IRM	4.9	8.4	**8.0**	19.1	12.7
			IHM	4.9	8.5	8.8	19.5	13.0

masks may even outstrip the gains obtained. Finding a mask that is both effective in variation removal and robust to estimation errors is crucial to practical applications. We hence build three 6-hidden layer DNN-based mask estimators respectively. Details about the learning of the mask estimators can be found in [186]. In the "E3" part of Table 6.23, we first evaluate the estimated masks with the ideally masked DNN, *i.e.* the DNN used in "E2". All the masks degrade the performance and our proposed IHM shows the least degradation. It suggests that the errors in the estimated masks are crucial. To address the mismatch between the ideal masks and the estimated masks, we retrain the baseline DNN with the estimated masks instead of the ideal ones. From the "E4" results of Table 6.23, our IHM performs the best and is the only one that improves over the baseline DNN's performance. The improvement is also statistically significant at the level of $p = 0.05$, using the matched pair sentence segment word error method. By further comparing the relative WER reductions of these masks in Fig. 6.8, spectral masking is preferable when the noise is simple (such as the car noise in set 02), but it degrades to a large extent when the additive-noise assumption fails or mismatches exist. The proposed IHM aims to identify the noise-invariant feature detectors and is hence more reliable across different noise types. On some sets (such as 03, 06, 07, 11 and 12), degradation has been observed for all the masks, which may require better mask estimations.

In Section 6.2.3 and Section 6.2.4, although the estimated spectral masks cannot

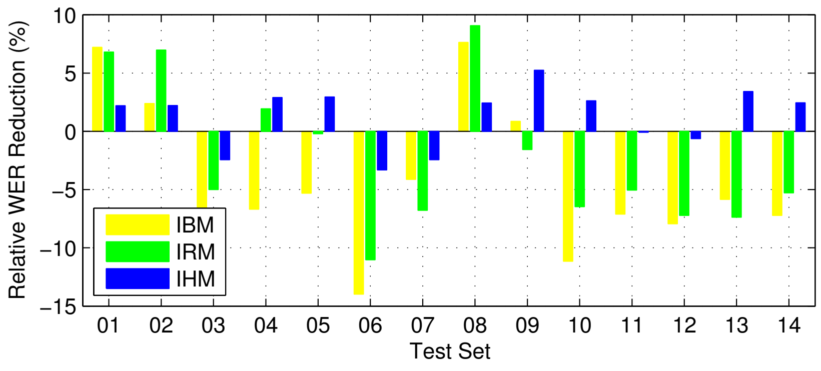


Figure 6.8: A comparison of the estimated IBM, IRM and IHM using relative WER reductions from the baseline system on Aurora-4.

improve the baseline DNN's performance, they do provide complementary information to yield gains by averaging the two sets of posteriors. We thus average the posteriors generated from "E4" and the baseline respectively. The results reported in "E5" of Table 6.23 reconfirm the finding in Section 6.2.3 and Section 6.2.4, and further gains are achieved for all three masks, with the IBM benefiting most from the posterior averaging.

Discussions

From these experiments, the proposed Ideal Hidden-activation Mask (IHM) at the first hidden layer of the DNN acoustic models further improves their noise robustness for speech recognition. Our IHM identifies noise-invariant hidden feature detectors and discards those that are dependent on noise. Experimental results on the Aurora-4 dataset show that the proposed IHM is more robust to the mask estimation errors. Unlike the spectral masks, the IHM has no noise type assumptions and could obtain consistent gains across different test sets.

6.3.2 Noise Code

In this section, we will discuss the further development of our hidden masking technique. Instead of using an additional DNN for the estimation of masks, we proposed to use a noise code approach that is directly integrated within the existing DNN AM. This noise code technique estimates a generic transformation matrix and a environment-dependent noise code, the product of which distributes the bias shifts to each hidden unit. These shifts attenuate the original hidden activation values and have a similar effect as applying soft masks.

Experiment Setup

Similar to previous experiments, we use the 24D FBank features together with the delta and the accelerator parameters as the input feature representation. Utterance-based MVN is adopted for feature normalization and a context window of 11 adjacent frames is used for context expansion. The multi-style trained 6 hidden-layer (6H) DNN has an average WER of 13.4%.

Unlike previous experiments, the noise code $\boldsymbol{c}$ is environment-dependent. We can use the codes estimated on training for testing purpose only if we know the exact training and testing noise condition mapping. Generally speaking this is an impractical assumption. Hence, the code vector $\boldsymbol{c}$ has to be estimated either from some supervised enrollment data or using unsupervised test data. This is similar to the LIN adaptation technique but with much fewer parameters.

Specific to the Aurora-4 corpus, the transformation is estimated from the training data and the code vectors are estimated from the development set (referred to as "dev") which has exactly the same complete set of noise conditions as the testing. Besides this standard setup, we justify the benefit of using more enrollment data from the extra development data (referred to as "dev330") that is commonly not utilized. Additionally, there is indeed a mapping between conditions in the multi-style training data and the testing data, which allows us to estimate code vectors during training (referred to as "train").

Due to the different data-dependency of the two parameters, we could either learn them jointly or alternatively. In the joint training, both of them are updated after each training batch; while for the alternative training, only one set of parameters will be updated at a specific epoch and the training alternates between them. We will refer to these two training manners as "joint" and "alter" respectively.

Effectiveness for Recognition

We firstly use the proposed parallel data based objective for learning the noise code parameters. However, the training fails to converge. One probable reason is that a constant shift is hard to address the large noise variations in the first hidden layer. By reverting to the standard AM DNN training, performance gains could be observed. The results are tabulated in Table 6.24. The code vector used in our experiments has the same dimensionality of 32. The improvements are relatively small. This may be due to the fact that with the DNN AM's multi-layered back propagation, the error signals are rather weak for the learning of effective noise code parameters. Comparing Table 6.23 and Table 6.24, it is possible to see that the noise code approach indeed approximates the effect of the IHM by yielding similar performance.

Our noise code approach is similar to the "speaker code" technique in [181, 182].

Despite the fact that one additional adaptation NN is used, we still justify its effectiveness on our noisy speech recognition task. Similarly, we estimate the one hidden layer adaptation NN and the transformation matrices on the training data. Three sets of code vectors estimated on "train", "dev" and "dev330" are evaluated. None of them could reach the baseline DNN AM's performance. The code vectors estimated on the training data give the averaged 13.7% WER, which has the smallest degradation.

Table 6.24: WER (%) performance of noise codes with different experiment configurations on Aurora-4.

System		Test Set				Avg.
Training	Data	A	B	C	D	
Baseline		5.0	8.8	9.0	20.0	13.4
joint	train	5.2	8.8	**8.9**	20.0	13.4
	dev	**5.1**	**8.7**	**8.9**	19.9	13.3
	dev330	**5.1**	**8.7**	**8.9**	**19.7**	**13.2**
alter	train	5.3	8.8	**8.9**	20.1	13.4
	dev	**5.2**	8.8	**8.9**	19.9	13.3
	dev330	**5.2**	**8.7**	9.0	**19.8**	**13.2**

Discussions

From this study, we verified the idea of approximating the hidden masking effect within the original DNN AM. No extra mask estimation DNNs are required anymore. A noise vector representing the environment statistics and a matrix that transforms it to the bias shifts of the sigmoid hidden activation functions are the only modifications. However, due to the large noise variations in the first hidden layer representations and the simple linear noise correction process of the noise code approach, the performance gains are relatively small. The idea of integrating the effect of the code vector using an extra DNN to address the speaker variations [181, 182] has also been tested for our noisy speech recognition problem. It fails to improve the baseline DNN as well, which suggests that exploring different but helpful information for DNNs could be more effective.

6.4 Summary

In this chapter, we justified the effectiveness of our proposed techniques in improving different representations' noise robustness of the DNN AM. Those techniques include the VTS-MVN for the normalized input representation, the DSTC for the context expanded input representation and the spectral masking with LIN adaptation for the spectral feature representation, the IHM and the noise code for hidden representations.

Table 6.25: Reported average WER(%) performance of multi-style trained systems on Aurora-2.

System	WER(%)
Projection-based fMLLR [187]	8.6
Lasso4 [188]	7.9
MMSE-SPLICE [189]	7.8
VTS [146], CAUG-LM [190]	7.7
CMN [146]	7.1
Extended VTS [191]	7.0
PLP-Tandem [192]	6.9
AFE [146, 193]	6.8
CMVN [146]	6.5
NAT [146]	6.3
VTS + CMVN [194]	6.2
DNN	**5.2**
VTS-MVN (DNN)	**5.0**
RBM-DNN	**4.9**
RBM-DNN+LIN	**4.8**
Masking	**4.7**
ESSEM-MCM [195]	4.6
DSTC	**4.4**
Masking (Posterior Average)	**4.3**

Table 6.26: Reported average WER(%) performance of multi-style trained systems on Aurora-4.

System	WER(%)
VQ-mask [196]	25.8
VTS [197]	17.9
NAT [197]	16.0
SGMM NAT+JUD [198]	15.7
MPE-NAT [199]	15.3
NAT + Derivative Kernels [197]	14.8
NAT + Joint MLLR/VTS [200]	13.4
DNN [24]	**13.4**
IHM, Noise Code	**13.2**
RBM-DNN	**13.1**
Dropout+NAT [24]	12.4
RBM-DNN+LIN	**12.2**
cFDLR [168]	12.1
Masking	**11.8**
Masking (Posterior Average)	**11.4**

As benchmark tasks, many researchers have contributed their efforts in advancing the recognition performance on the Aurora-2 and the Aurora-4 tasks. We hence summarize the various reported results in Table 6.25 and Table 6.26 with our results in bold fonts.

Chapter 7

Conclusions

This thesis has investigated the noise-robust automatic speech recognition problem using Deep Neural Networks (DNNs). Despite the large improvements reported in the literature by adopting DNNs for acoustic modeling, severe degradation has also been observed when they are used under adverse noise conditions. Additionally, many of the existing compensation techniques have been found to be ineffective in DNNs. Based on the DNN's layered representation learning, a specific noise-robust representation learning framework is proposed in this study. The main contributions of this research are the techniques we have developed to address the noise variations in different levels of representations of the DNN AM. More specifically, a Vector Taylor Series - Mean Variance Normalization (VTS-MVN) technique is developed to improve the reliability of estimating utterance-based MVN statistics from short utterances. With this VTS-MVN, the normalized input representation is made more reliable and effective for the DNN AM. After that, the context expanded representation is studied. Longer contexts have been found to be crucial for DNNs to automatically learn the environment statistics. A Deep Split Temporal Context (DSTC) technique is hence developed, to model the long span of speech context information for improved generalization capabilities in unknown noise conditions. Besides these two techniques that improve the reliability of existing representations under noise conditions, a spectral masking technique targeted at directly reducing noise variations has also been developed, first for the input spectral feature representation and then extended to the DNN AM's hidden representations. Finally, the noise code technique has been proposed to mimic the effect of masking without the use of extra mask estimation DNNs. Experimental evaluations have been conducted on the benchmark Aurora-2 and Aurora-4 tasks, and clear performance gains have been achieved. Our system has successfully yielded the best reported performance on both the Aurora-2 and the Aurora-4 datasets at the time of writing when using the spectral masking with LIN adaptation approach.

The following part of this chapter reviews the key findings in more details and concludes this thesis with discussions on potential future directions.

7.1 Summary of Results

The VTS-MVN is a kind of feature normalization technique. In comparison to other techniques, the VTS-MVN is more flexible in balancing the normalization reliability, effectiveness and timeliness. It utilizes the global MVN as the prior MVN estimation when no or not enough target speech information has been observed. Once a reliable target environment estimation is obtained, the VTS-MVN adopts the model-based VTS compensation to update the global MVN toward that specific testing environment. Depending on the update schedule, the VTS-MVN could revert to the global MVN if no update is done, and mimic the utterance-based MVN if the noise statistics are updated per utterance. Experimental results on Aurora-2 verifies the effectiveness of the VTS-MVN. However, the gains over utterance-based MVN is relatively small. Moreover, for long utterances, utterance-based MVN is usually sufficient.

To utilize a longer span of acoustic information, the DSTC technique models the partial contexts independently and a final linear classifier is good enough for phonetic prediction. Effectively the DSTC builds large models in terms of both depth and width with a relatively small amount of parameters by identifying block structures. With these structure constraints, better generalization capabilities have been observed on the Aurora-2 task. However, the DSTC fails to achieve similar improvements on Aurora-4 due to the higher complexity of the task and the difficulty of building huge DNNs that have the same degree of over-fitting on Aurora-4 as on Aurora-2.

The spectral masking technique directly addresses the noise corruption by removing the noise-dominant time-frequency units in the power spectral domain. Masks are used to separate speech and noise information. The estimated spectral masks are effective in reducing noise variations. However, due to the use of DNNs for the mask estimation, generalizations in unseen noise conditions are poor. By further incorporating the Linear Input Network (LIN) adaptation for both the mask estimator and the acoustic model, large error reductions could be achieved. Compared to the conventional spectral masking, the success of our approach lies in the use of direct masking, that gets rid of potential errors brought by the extra reconstruction process and the LIN adaptation that addresses the mismatch problem of statistical mask estimation models.

Finally, by extending the spectral masking into hidden representations, the Ideal Hidden-activation Mask (IHM) is proposed. Through the investigation of IHMs, noise variations are found in all levels of the representations learned automatically by DNNs with lower layers having more. Improved robustness could be achieved by masking away those variations, which also suggests redundancies inside DNNs' hidden representations. Furthermore, by formulating the masking as the effect of attenuating the sigmoid functions' activation levels, the noise code technique has shown its potential in approximating the masking effect without additional DNNs. Although the gains from using these hidden masking techniques are relatively smaller than spectral masking, they have shown better robustness against mask estimation errors.

7.2 Future Work

The focus of this work is on the DNN acoustic model. It has less model assumptions and better variation modeling capabilities than the conventional Gaussian Mixture Model (GMM). Due to the underlying differences, many popular techniques developed for GMM-based systems are not effective for DNNs. One of the common beliefs is that DNNs are capable of learning better predictions automatically from large amounts of data. In our study, for a given dataset, exploring different information could still improve their performance. The masking method is effectively injecting parallel clean and noisy speech difference information into DNNs which may not be explored in the standard learning algorithms. And the noise code method injects the noise factors into the DNN model. However, the current noise codes are optimized within the original DNN learning framework, which may be the reason for its limited effectiveness. A potential direction would be to estimate those noise codes reliably for a different but helpful objective, such as minimizing the clean and noisy representation differences.

Besides the objective, the noise code is currently estimated per noise condition. Even under the same noise condition, variations still exist. For the masking approach, a mask vector will be produced for each feature frame. From the feature transformation perspective, the masks could be treated as frame-dependent diagonal linear transforms. This hence has far greater correction capabilities but also requires much higher accuracy than utterance-dependent or condition-dependent transformations. It may also be the reason for the limited gains obtained by the current noise code method. Estimating much more reliable noise codes with finer granularities could probably lead to improved noise robustness.

In this research, we only focus on the additive noise and channel distortions. In reality, there are many other types of noise, such as reverberation noise, interfering speech and so on. Extending the masking technique into those problems would be promising. However, the challenge remains the same, *i.e.* how to reliably estimate masks under different scenarios.

The masks investigated in this work are all referred to as "ideal" masks because of the use of parallel clean and noisy data. In practice, it is impossible to obtain such data since they are mainly artificially created. Masks encoding similar complementary information as those “ideal” masks, but generated from realistic recordings, would be more desirable. One possible direction is to explore the information differences among speech that has been recorded from microphone arrays. Human beings have two ears to receive and process speech information. Utilizing multiple microphones would be helpful to ASRs. Although this kind of parallel data is more practical to collect, how effective the masks derived from these data needs to be justified first.

Bibliography

[1] K. Davis, R. Biddulph, and S. Balashek, "Automatic recognition of spoken digits," *The Journal of the Acoustical Society of America*, vol. 24, p. 637, 1952. 1

[2] J. Baker, "The DRAGON system – An overview," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 23, no. 1, pp. 24–29, 1975. 1

[3] F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532–556, 1976. 1

[4] W. Macherey, L. Haferkamp, R. Schlüter, and H. Ney, "Investigations on error minimizing training criteria for discriminative training in automatic speech recognition," in *Proc. Interspeech.* ISCA, 2005. 1

[5] E. McDermott, T. J. Hazen, J. Le Roux, A. Nakamura, and S. Katagiri, "Discriminative training for large-vocabulary speech recognition using minimum classification error," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 1, pp. 203–223, 2007. 1

[6] D. Povey, "Discriminative training for large vocabulary speech recognition," *Cambridge University*, vol. 79, 2004. 1

[7] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," in *Proc. ICASSP*, vol. 1. IEEE, 2005, pp. 961–964. 1

[8] A. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition," in *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009. 1, 9, 39

[9] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP.* IEEE, 2013. 1

[10] B. Schrauwen and E. Antonelo, "TIMIT benchmark results," 03 2010. [Online]. Available: http://organic.elis.ugent.be/organic/benchmarks/287 1

[11] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "An application of pretrained deep neural networks to large vocabulary conversational speech recognition," Tech. Rep. 001, Department of Computer Science, University of Toronto, Tech. Rep., 2012. 2, 9

[12] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 4, pp. 357–366, 1980. 4, 16

[13] X. Huang, A. Acero, H. W. Hon *et al.*, *Spoken language processing.* Prentice Hall, 2001, vol. 15. 4, 21

[14] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 34, no. 1, pp. 52–59, 1986. 5

[15] H. Bourlard and N. Morgan, *Connectionist speech recognition: A hybrid approach.* Springer,

1994. 8, 33, 38, 62

[16] Y. Bengio, “Artificial neural networks and their application to sequence recognition,” Ph.D. dissertation, McGill University, 1991. 8

[17] N. Morgan, Q. Zhu, A. Stolcke, K. Sonmez, S. Sivadas, T. Shinozaki, M. Ostendorf, P. Jain, H. Hermansky, D. Ellis *et al.*, “Pushing the envelope-aside [speech recognition],” *Signal Processing Magazine, IEEE*, vol. 22, no. 5, pp. 81–88, 2005. 8

[18] J. Markoff, “Scientists see promise in deep-learning programs,” 11 2012. [Online]. Available: http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-intelligence.html 9

[19] D. McClain, “Once again, machine beats human champion at chess,” 12 2006. [Online]. Available: http://www.nytimes.com/2006/12/05/crosswords/chess/05cnd-chess.html 9

[20] J. Markoff, “Computer wins on ‘jeopardy!’: trivial, it’s not,” 2 2011. [Online]. Available: http://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html 9

[21] A. Mohamed, G. Dahl, and G. Hinton, “Acoustic modeling using deep belief networks,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012. 9, 45, 97

[22] G. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012. 9

[23] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012. 9, 39, 40

[24] D. Yu, M. Seltzer, J. Li, J. Huang, and F. Seide, “Feature learning in deep neural networks - a study on speech recognition tasks,” in *Proc. ICLR*, 2013. 9, 121

[25] A. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. A. Picheny, “Deep belief networks using discriminative features for phone recognition,” in *Proc. ICASSP.* IEEE, 2011, pp. 5060–5063. 9

[26] J. H. Hansen, “Analysis and compensation of speech under stress and noise for environmental robustness in speech recognition,” *Speech communication*, vol. 20, no. 1, pp. 151–173, 1996. 14

[27] J. C. Junqua and Y. Anglade, “Acoustic and perceptual studies of Lombard speech: Application to isolated-words automatic speech recognition,” in *Proc. ICASSP.* IEEE, 1990, pp. 841–844. 14

[28] S. E. Bou Ghazale and J. H. Hansen, “A comparative study of traditional and newly proposed features for recognition of speech under stress,” *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 4, pp. 429–442, 2000. 14

[29] J. C. Junqua, “The Lombard reflex and its role on human listeners and automatic speech recognizers,” *The Journal of the Acoustical Society of America*, vol. 93, p. 510, 1993. 14

[30] S. E. Bou Ghazale and J. H. Hansen, “Duration and spectral based stress token generation for HMM speech recognition under stress,” in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 1. IEEE, 1994, pp. I–413. 14

[31] O. Siohan, Y. Gong, and J. Haton, “A Bayesian approach to phone duration adaptation for Lombard speech recognition,” in *Proc. Eurospeech.* ISCA, 1993. 14

[32] A. Acero, “Acoustical and environmental robustness in automatic speech recognition,” Ph.D. dissertation, Carnegie Mellon University, 1990. 14

[33] M. Gales, “Model-based techniques for noise robust speech recognition,” Ph.D. dissertation, Cambridge University, 1995. 14, 19, 22, 56

[34] P. Moreno, "Speech recognition in noisy environments," Ph.D. dissertation, Carnegie Mellon University, 1996. 14, 23, 27

[35] C. P. Chen, "Noise robustness in automatic speech recognition," Ph.D. dissertation, University of Washington, 2004. 14

[36] H. Liao, "Uncertainty decoding for noise robust speech recognition," Ph.D. dissertation, Cambridge University, 2007. 14, 28, 57

[37] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990. 16

[38] H. Hermansky and N. Morgan, "RASTA processing of speech," *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 4, pp. 578–589, 1994. 16

[39] M. Westphal, "The use of cepstral means in conversational speech recognition," in *Proc. Eurospeech.* ISCA, 1997. 17

[40] S. Molau, F. Hilger, and H. Ney, "Feature space normalization in adverse acoustic conditions," in *Proc. ICASSP*, vol. 1. IEEE, 2003, pp. I–656. 17

[41] S. S. Wang, J. W. Hung, and Y. Tsao, "A study on cepstral sub-band normalization for robust ASR," in *Proc. ISCSLP.* IEEE, 2012, pp. 141–145. 17

[42] C. P. Chen and J. A. Bilmes, "MVA processing of speech features," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 1, pp. 257–270, 2007. 17

[43] F. Hilger and H. Ney, "Quantile based histogram equalization for noise robust large vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 3, pp. 845–854, 2006. 17

[44] ETSI, "Speech processing, transmission and quality aspects (STQ); distributed speech recognition; advanced front-end feature extraction algorithm; compression algorithms," *ETSI ES*, vol. 202, no. 050, p. v1, 2007. 17

[45] P. Lockwood and J. Boudy, "Experiments with a nonlinear spectral subtractor (NSS), hidden Markov models and the projection, for robust speech recognition in cars," *Speech communication*, vol. 11, no. 2, pp. 215–228, 1992. 17

[46] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 27, no. 2, pp. 113–120, 1979. 17

[47] Y. Ephraim, D. Malah, and B. H. Juang, "On the application of hidden Markov models for enhancing noisy speech," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 12, pp. 1846–1856, 1989. 18

[48] Y. Ephraim, "A Bayesian estimation approach for speech enhancement using hidden Markov models," *Signal Processing, IEEE Transactions on*, vol. 40, no. 4, pp. 725–735, 1992. 18

[49] B. Logan and A. Robinson, "Enhancement and recognition of noisy speech within an autoregressive hidden Markov model framework using noise estimates from the noisy signal," in *Proc. ICASSP*, vol. 2. IEEE, 1997, pp. 843–846. 18

[50] C. Seymour and M. Niranjan, "An HMM-based cepstral-domain speech enhancement system," in *Proc. ICSLP*, 1994, pp. 1595–1598. 18

[51] Y. Ephraim, "Statistical-model-based speech enhancement systems," *Proceedings of the IEEE*, vol. 80, no. 10, pp. 1526–1555, 1992. 18

[52] M. Gales, "Predictive model-based compensation schemes for robust speech recognition," *Speech Communication*, vol. 25, no. 1, pp. 49–74, 1998. 18, 22

[53] Y. Gong, "Speech recognition in noisy environments: A survey," *Speech communication*, vol. 16, no. 3, pp. 261–291, 1995. 18

[54] U. Yapanel, J. H. Hansen, R. Sarikaya, and B. Pellom, "Robust digit recognition in noise: An evaluation using the Aurora corpus," in *Proc. Eurospeech.* ISCA, 2001. 18

[55] A. Varga and H. J. 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, no. 3, pp. 247–251, 1993. 18

[56] L. Deng, A. Acero, M. Plumpe, and X. Huang, “Large-vocabulary speech recognition under adverse acoustic environments,” in *Proc. ICSLP*, vol. 3, 2000, pp. 806–809. 18

[57] R. Lippmann, E. Martin, and D. Paul, “Multi-style training for robust isolated-word speech recognition,” in *Proc. ICASSP*, vol. 12. IEEE, 1987, pp. 705–708. 18

[58] H. G. Hirsch and 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. 18, 27, 40, 52, 93

[59] J. L. Gauvain and C. H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 2, pp. 291–298, 1994. 19, 21

[60] M. Gales, “Maximum likelihood linear transformations for HMM-based speech recognition,” *Computer speech and language*, vol. 12, no. 2, 1998. 20, 21

[61] C. J. Leggetter and P. C. Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models,” *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995. 20

[62] M. Gales, *The generation and use of regression class trees for MLLR adaptation.* University of Cambridge, Department of Engineering, 1996. 21

[63] K. Shinoda, “Speaker adaptation with autonomous control using tree structure,” in *Proc. EuroSpeech.* ISCA, 1995. 21

[64] T. Watanabe, K. Shinoda, K. Takagi, and K.-I. Iso, “High speed speech recognition using tree-structured probability density function,” in *Proc. ICASSP*, vol. 1. IEEE, 1995, pp. 556–559. 21

[65] S. Young, G. Evermann, M. Gales, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK book version 3.4.* Cambridge University Engineering Department, 2006. 21

[66] G. Saon, G. Zweig, and M. Padmanabhan, “Linear feature space projections for speaker adaptation,” in *Proc. ICASSP*, vol. 1. IEEE, 2001, pp. 325–328. 21

[67] M. Gales and S. Young, “Robust speech recognition in additive and convolutional noise using parallel model combination,” *Computer Speech & Language*, vol. 9, no. 4, pp. 289–307, 1995. 21, 22

[68] M. Gales and S. Young, “An improved approach to the hidden Markov model decomposition of speech and noise,” in *Proc. ICASSP*, vol. 1. IEEE, 1992, pp. 233–236. 22

[69] A. Varga, R. Moore, J. Bridle, K. Ponting, and M. Russel, “Noise compensation algorithms for use with hidden Markov model based speech recognition,” in *Proc. ICASSP.* IEEE, 1988, pp. 481–484. 23

[70] A. Acero, L. Deng, T. Kristjansson, and J. Zhang, “HMM adaptation using vector Taylor series for noisy speech recognition,” in *Proc. ICSLP*, vol. 3, 2000, pp. 869–872. 23

[71] D. Y. Kim, C. Kwan Un, and N. S. Kim, “Speech recognition in noisy environments using first-order vector Taylor series,” *Speech Communication*, vol. 24, no. 1, pp. 39–49, 1998. 23

[72] J. A. Arrowood, “Using observation uncertainty for robust speech recognition,” Ph.D. dissertation, Georgia Institute of Technology, 2003. 24

[73] J. A. Arrowood and M. A. Clements, “Using observation uncertainty in HMM decoding,” in *Proc. ICSLP*, 2002. 24

[74] Q. Huo and C. H. Lee, "A Bayesian predictive classification approach to robust speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 2, pp. 200–204, 2000. 24

[75] M. Cooke, P. Green, L. Josifovski, and A. Vizinho, "Robust automatic speech recognition with missing and unreliable acoustic data," *Speech communication*, vol. 34, no. 3, pp. 267–285, 2001. 24, 25, 26

[76] B. Raj and R. M. Stern, "Missing-feature approaches in speech recognition," *Signal Processing Magazine*, vol. 22, no. 5, pp. 101–116, 2005. 24, 25, 27, 67

[77] J. N. Holmes, W. J. Holmes, and P. N. Garner, "Using formant frequencies in speech recognition," in *Proc. Eurospeech.* ISCA, 1997. 24

[78] L. Deng, J. Droppo, and A. Acero, "Dynamic compensation of HMM variances using the feature enhancement uncertainty computed from a parametric model of speech distortion," *Speech and Audio Processing, IEEE Transactions on*, vol. 13, no. 3, pp. 412–421, 2005. 24, 25

[79] M. Benitez, J. Segura, A. Torre, J. Ramirez, and A. Rubio, "Including uncertainty of speech observations in robust speech recognition," in *Proc. ICSLP*, 2004. 24, 25, 27

[80] M. Wolfel and F. Faubel, "Considering uncertainty by particle filter enhanced speech features in large vocabulary continuous speech recognition," in *Proc. ICASSP*, vol. 4. IEEE, 2007, pp. IV–1049. 24

[81] V. Stouten, H. Van Hamme, and P. Wambacq, "Accounting for the uncertainty of speech estimates in the context of model-based feature enhancement," in *Proc. ICSLP*, 2004. 25

[82] L. Deng, J. Droppo, and A. Acero, "Exploiting variances in robust feature extraction based on a parametric model of speech distortion," in *Proc. ICSLP*, vol. 4, no. 1, 2002, p. 1. 25

[83] J. Droppo, A. Acero, and L. Deng, "Uncertainty decoding with SPLICE for noise robust speech recognition," in *Proc. ICASSP*, vol. 1. IEEE, 2002, pp. I–57. 25

[84] T. T. Kristjansson, "Speech recognition in adverse environments: A probabilistic approach," Ph.D. dissertation, University of Waterloo, 2002. 25

[85] H. Liao and M. Gales, "Joint uncertainty decoding for noise robust speech recognition," in *Proc. Interspeech.* ISCA, 2005. 25

[86] A. Morris, J. Barker, and H. Bourlard, *From missing data to maybe useful data: Soft data modelling for noise robust ASR*, 2001. 26

[87] J. Barker, M. Cooke, and P. Green, "Robust ASR based on clean speech models: An evaluation of missing data techniques for connected digit recognition in noise," in *Proc. Eurospeech.* ISCA, 2001. 26

[88] B. Raj, M. Seltzer, and R. M. Stern, "Robust speech recognition: The case for restoring missing features," in *Proc. Eurospeech.* ISCA, 2001. 26

[89] H. Van Hamme, "Robust speech recognition using missing feature theory in the cepstral or LDA domain," in *Proc. Eurospeech.* ISCA, 1973. 27

[90] B. Frey, L. Deng, A. Acero, and T. Kristjansson, "ALGONQUIN: Iterating Laplaces method to remove multiple types of acoustic distortion for robust speech recognition," in *Proc. Eurospeech.* ISCA, 2001. 27

[91] J. Segura, M. Benitez, A. De La Torre, S. Dupont, and A. Rubio, "VTS residual noise compensation," in *Proc. ICASSP*, vol. 1. IEEE, 2002, pp. I–409. 27

[92] V. Stouten, J. Duchateau, P. Wambacq *et al.*, "Evaluation of model-based feature enhancement on the Aurora-4 task," in *Proc. Eurospeech.* ISCA, 2003. 27

[93] V. Stouten, P. Wambacq *et al.*, "Model-based feature enhancement with uncertainty decoding for noise robust ASR," *Speech communication*, vol. 48, no. 11, pp. 1502–1514, 2006. 27

[94] V. Stouten, K. Demuynck, P. Wambacq *et al.*, "Robust speech recognition using model-based

feature enhancement," in *Proc. Eurospeech.* ISCA, 2003. 27

[95] T. Robinson and F. Fallside, "A recurrent error propagation network speech recognition system," *Computer Speech & Language*, vol. 5, no. 3, pp. 259–274, 1991. 30

[96] S. Makino, T. Kawabata, and K. Kido, "Recognition of consonant based on the perceptron model," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 8. IEEE, 1983, pp. 738–741. 30

[97] 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*, vol. 37, no. 3, pp. 328–339, 1989. 30

[98] M. Franzini, K. F. Lee, and A. Waibel, "Connectionist Viterbi training: A new hybrid method for continuous speech recognition," in *Proc. ICASSP.* IEEE, 1990, pp. 425–428. 30

[99] R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis.* Wiley New York, 1973, vol. 3. 30

[100] B. Li and K. C. Sim, "Hidden logistic linear regression for support vector machine based phone verification," in *Proc. Interspeech.* ISCA, 2010. 31

[101] D. E. Rumelhart, G. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 1, p. 213, 2002. 32, 71

[102] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989. 33

[103] V. Kuurkova, "Kolmogorov's theorem and multilayer neural networks," *Neural networks*, vol. 5, no. 3, pp. 501–506, 1992. 33

[104] F. Rosenblatt, "Principles of neurodynamics, perceptrons and the theory of brain mechanisms," DTIC Document, Tech. Rep., 1961. 33

[105] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," DTIC Document, Tech. Rep., 1985. 33

[106] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1974. 33

[107] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2010. 33

[108] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010. 33

[109] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006. 33, 34, 37, 66, 74

[110] G. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002. 34, 36

[111] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausble Inference.* Morgan Kaufmann Pub, 1988. 34

[112] G. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade.* Springer, 2012, pp. 599–619. 37, 71

[113] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech.* ISCA, 2011. 37

[114] D. Yu, L. Deng, F. Seide, and G. Li, "Discriminative pretraining of deep neural networks," May 2013, uS Patent App. 13/304,643. 37

[115] J. S. Bridle, "Alpha-Nets: A recurrent neural network architecture with a hidden Markov model interpretation," *Speech Communication*, vol. 9, no. 1, pp. 83–92, 1990. 38

[116] L. T. Niles and H. F. Silverman, "Combining hidden Markov model and neural network classifiers," in *Proc. ICASSP.* IEEE, 1990, pp. 417–420. 38

[117] N. Parihar and J. Picone, “Aurora working group: DSR front end LVCSR evaluation AU/384/02,” *Inst. for Signal and Information Process, Mississippi State University, Tech. Rep*, 2002. 40, 94

[118] J. Chen, J. Benesty, Y. Huang, and E. J. Diethorn, “Fundamentals of noise reduction,” in *Springer Handbook of Speech Processing*, J. Benesty, M. Sondhi, and Y. Huang, Eds. Springer Berlin Heidelberg, 2008, pp. 843–872. 43

[119] P. Scalart and J. Vieira Filho, “Speech enhancement based on a priori signal to noise estimation,” in *Proc. ICASSP*, vol. 2. IEEE, 1996, pp. 629–632. 43

[120] Y. Ephraim and D. Malah, “Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 32, no. 6, pp. 1109–1121, 1984. 43

[121] R. Martin, “Speech enhancement based on minimum mean-square error estimation and super-Gaussian priors,” *Speech and Audio Processing, IEEE Transactions on*, vol. 13, no. 5, pp. 845–856, 2005. 43

[122] J. H. Hansen, V. Radhakrishnan, and K. H. Arehart, “Speech enhancement based on generalized minimum mean square error estimators and masking properties of the auditory system,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 14, no. 6, pp. 2049–2063, 2006. 43

[123] T. Lotter and P. Vary, “Speech enhancement by MAP spectral amplitude estimation using a super-Gaussian speech model,” *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 1110–1126, 2005. 43

[124] S. Suhadi, C. Last, and T. Fingscheidt, “A data-driven approach to a priori SNR estimation,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 186–195, 2011. 43

[125] R. McAulay and M. Malpass, “Speech enhancement using a soft-decision noise suppression filter,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 2, pp. 137–145, 1980. 43

[126] U. Kjems and J. Jensen, “Maximum likelihood based noise covariance matrix estimation for multi-microphone speech enhancement,” *Proc. EUSIPCO*, 2012. 43

[127] Y. C. Su, Y. Tsao, J. E. Wu, and F. R. Jean, “Speech enhancement using generalized maximum a posteriori spectral amplitude estimator,” in *Proc. ICASSP*. IEEE, 2013. 43

[128] N. Andrew, “Learning feature hierarchies and deep learning,” in *ECCV-2010 Tutorial: Feature Learning for Image Classification*, 2010. 45

[129] J. Li, D. Yu, J.-T. Huang, and Y. Gong, “Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM,” in *Proc. SLT*. IEEE, 2012, pp. 131–136. 45

[130] A. Mohamed, G. Hinton, and G. Penn, “Understanding how deep belief networks perform acoustic modelling,” in *Proc. ICASSP*. IEEE, 2012. 45

[131] B. Li and K. C. Sim, “Noise adaptive front-end normalization based on vector Taylor series for deep neural networks in robust speech recognition,” in *Proc. ICASSP*. IEEE, 2013. 45

[132] N. Jaitly and G. Hinton, “Learning a better representation of speech soundwaves using restricted Boltzmann machines,” in *Proc. ICASSP*. IEEE, 2011, pp. 5884–5887. 45

[133] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, 1995. 45

[134] P. Sermanet, S. Chintala, and Y. LeCun, “Convolutional neural networks applied to house numbers digit classification,” in *Proc. ICPR*. IEEE, 2012, pp. 3288–3291. 45

[135] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” *Neural Networks, IEEE Transactions on*, vol. 8, no. 1, pp. 98–113, 1997. 45

[136] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, “Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition,” in *Proc. ICASSP*. IEEE, 2012, pp. 4277–4280. 45

[137] C. Zhang, 08 2013. [Online]. Available: http://freemind.pluskid.org/machine-learning/deep-learning-and-shallow-learning/ 48

[138] Y. Bengio, “Learning deep architectures for AI,” *Foundations and trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009. 48

[139] Y. Bengio, “Deep learning of representations: Looking forward,” *CoRR*, vol. abs/1305.0445, 2013. 48

[140] B. Li and K. C. Sim, “Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems,” in *Proc. Interspeech*. ISCA, 2010. 49, 72, 75

[141] O. Vinyals, S. V. Ravuri, and D. Povey, “Revisiting recurrent neural networks for robust ASR,” in *Proc. ICASSP*. IEEE, 2012. 52, 72

[142] A. L. Maas, Q. V. Le *et al.*, “Recurrent neural networks for noise reduction in robust ASR,” in *Proc. Interspeech*. ISCA, 2012. 52

[143] L. Deng, A. Acero, L. Jiang, J. Droppo, and X. Huang, “High-performance robust speech recognition using stereo training data,” in *Proc. ICASSP*, vol. 1. IEEE, 2001, pp. 301–304. 52

[144] ETSI, “Advanced front-end feature extraction algorithm,” in *Technical Report. ETSI ES 202 050*, 2007. 52

[145] P. Moreno, B. Raj, and R. Stern, “A vector Taylor series approach for environment-independent speech recognition,” in *Proc. ICASSP*, vol. 2. IEEE, 1996, pp. 733–736. 52, 55, 59

[146] O. Kalinli, M. Seltzer, J. Droppo, and A. Acero, “Noise adaptive training for robust automatic speech recognition,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 18, no. 8, pp. 1889–1901, 2010. 52, 57, 121

[147] S. Rennie, P. Fousek, and P. Dognin, “Factorial hidden restricted Boltzmann machines for noise robust speech recognition,” in *Proc. ICASSP*. IEEE, 2012, pp. 4297–4300. 52

[148] C. G. Gross and R. Jung, “Handbook of sensory physiology,” 1993. 61

[149] L. Aitkin, C. Dunlop, and W. Webster, “Click-evoked response patterns of single units in the medial geniculate body of the cat,” *Journal of Neurophysiology*, 1966. 61

[150] J. C. Stevens and J. W. Hall, “Brightness and loudness as functions of stimulus duration,” *Perception & Psychophysics*, vol. 1, no. 5, pp. 319–327, 1966. 61

[151] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv:1207.0580*, 2012. 62, 87

[152] H. Lee, C. Ekanadham, and A. Ng, “Sparse deep belief net model for visual area v2,” in *Proc. NIPS*, 2008. 62

[153] P. Schwarz, P. Matejka, and J. Cernocky, “Hierarchical structures of neural networks for phoneme recognition,” in *Proc. ICASSP*. IEEE, 2006. 62

[154] J. Boldt, “Binary masking & speech intelligibility,” Ph.D. dissertation, Aalborg Universitet, 2011. 66, 80

[155] D. L. Wang, G. J. Brown *et al.*, *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley interscience, 2006. 66, 80

[156] R. Lyon, “A computational model of binaural localization and separation,” in *Proc. ICASSP*. IEEE, 1983. 66

[157] G. J. Brown, “Computational auditory scene analysis: A representational approach,” Ph.D. dissertation, University of Sheffield, 1992. 66

[158] D. L. Wang and G. J. Brown, “Separation of speech from interfering sounds based on oscillatory correlation,” *Neural Networks, IEEE Transactions on*, vol. 10, no. 3, pp. 684–697, 1999. 66

[159] A. Narayanan and D. L. Wang, “The role of binary mask patterns in automatic speech recognition in background noise,” *The Journal of the Acoustical Society of America*, vol. 133, p. 3083, 2013. 66

[160] W. Hartmann, A. Narayanan *et al.*, “A direct masking approach to robust ASR,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 2013. 66, 67, 68

[161] D. L. Wang, “On ideal binary mask as the computational goal of auditory scene analysis,” *Speech separation by humans and machines*, 2005. 67, 80

[162] D. L. Wang, U. Kjems, M. Pedersen, J. Boldt, and T. Lunner, “Speech intelligibility in background noise with ideal binary time-frequency masking,” *The Journal of the Acoustical Society of America*, vol. 125, p. 2336, 2009. 67

[163] W. Hartmann, A. Narayanan *et al.*, “Nothing doing: Re-evaluating missing feature ASR,” *Reconstruction*, 2011. 67

[164] M. Seltzer, B. Raj, and R. M. Stern, “A Bayesian classifier for spectrographic mask estimation for missing feature speech recognition,” *Speech Communication*, 2004. 67, 68, 72

[165] S. Keronen, H. Kallasjoki *et al.*, “Mask estimation and imputation methods for missing data speech recognition in a multisource reverberant environment,” *Computer Speech & Language*, 2012. 67, 68, 72

[166] J. F. Gemmeke, Y. J. Wang *et al.*, “Application of noise robust MDT speech recognition on the SPEECON and speechdat-car databases.” in *Proc. Interspeech*. ISCA, 2009. 67, 68, 72

[167] A. Narayanan and D. L. Wang, “Ideal ratio mask estimation using deep neural networks for robust speech recognition,” in *Proc. ICASSP*. IEEE, 2013. 67, 68, 72, 75, 80, 85, 86

[168] A. Narayanan and D. L. Wang, “Investigation of speech separation as a front-end for noise robust speech recognition,” *OSU-CISRC-6/13-TR14*, 2013. 67, 68, 75, 76, 85, 115, 121

[169] A. Narayanan and D. L. Wang, “Coupling binary masking and robust ASR,” in *Proc. ICASSP*. IEEE, 2013. 70

[170] B. Li, Y. Tsao, and K. C. Sim, “An investigation of spectral restoration algorithms for deep neural networks based noise robust speech recognition,” in *Proc. Interspeech*. ISCA, 2013. 72

[171] B. Li and K. C. Sim, “Noise adaptive front-end normalization based on vector Taylor series for deep neural networks in robust speech recognition,” in *Proc. ICASSP*. IEEE, 2013. 72

[172] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, “Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system,” 1995. 72

[173] V. Abrash, H. Franco, A. Sankar, and M. Cohen, “Connectionist speaker normalization and adaptation,” in *Proc. Eurospeech*. ISCA, 1995. 72

[174] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, “Deep convolutional neural networks for LVCSR,” in *Proc. ICASSP*. IEEE, 2013, pp. 8614–8618. 75

[175] J. Gehring, W. Lee, K. Kilgour, I. Lane, Y. Miao, A. Waibel, and S. V. Campus, “Modular combination of deep neural networks for acoustic modeling,” in *Proc. Interspeech*. ISCA, 2013. 75

[176] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, “Improvements to deep convolutional neural networks for LVCSR,” in *Proc. ASRU*. IEEE, 2013, pp. 315–320. 75

[177] Y. Q. Wang and M. Gales, “TANDEM system adaptation using multiple linear feature transforms,” in *Proc. ICASSP*. IEEE, 2013. 75

[178] U. Kjems, J. Boldt *et al.*, “Role of mask pattern in intelligibility of ideal binary-masked noisy speech,” *The Journal of the Acoustical Society of America*, 2009. 80

[179] J. S. Bridle and S. Cox, “RecNorm: Simultaneous normalisation and classification applied to speech recognition,” in *Proc. NIPS*, 1990, pp. 234–240. 91

[180] M. Seltzer, D. Yu, and Y. Q. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. ICASSP.* IEEE, 2013. 91

[181] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *Proc. ICASSP.* IEEE, 2013. 91, 119, 120

[182] O. Abdel-Hamid and H. Jiang, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Proc. Interspeech.* ISCA, 2013. 91, 119, 120

[183] D. Povey, A. Ghoshal *et al.*, "The Kaldi speech recognition toolkit," in *Proc. ASRU.* IEEE, 2011. 94, 95

[184] J. Garofalo, D. Graff, D. Paul, and D. Pallett, "CSR-I (WSJ0) complete," in *LDC93S6A*, 2007. 94

[185] S. L. Chow, *Statistical significance: Rationale, validity and utility.* Sage, 1996, vol. 1. 115

[186] B. Li and K. C. Sim, "Improving robustness of deep neural networks via spectral masking for automatic speech recognition," in *Proc. ASRU.* IEEE, 2013. 117

[187] G. Saon, H. Huerta, and E. Jan, "Robust digit recognition in noisy environments: The IBM Aurora 2 system," in *Proc. Interspeech.* ISCA, 2001. 121

[188] X. Xiao, J. Li, E. Chng, and H. Li, "Lasso environment model combination for robust speech recognition," in *Proc. ICASSP.* IEEE, 2012. 121

[189] J. Droppo, "Feature compensation," *Techniques for Noise Robustness in Automatic Speech Recognition*, 2012. 121

[190] A. Ragni and M. Gales, "Structured discriminative models for noise robust continuous speech recognition," in *Proc. ICASSP.* IEEE, 2011. 121

[191] R. van Dalen and M. Gales, "Extended VTS for noise-robust speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, 2011. 121

[192] D. Ellis and M. Reyes-Gomez, "Investigations into tandem acoustic modeling for the Aurora task," in *Proc. Eurospeech.* ISCA, 2001. 121

[193] D. Macho, L. Mauuary, B. Noé, Y. Cheng, D. Ealey, D. Jouvet, H. Kelleher, D. Pearce, and F. Saadoun, "Evaluation of a noise-robust DSR front-end on Aurora databases," in *Proc. ICSLP*, 2002. 121

[194] J. Droppo and A. Acero, "Environmental robustness," in *Springer Handbook of Speech Processing.* Springer, 2008. 121

[195] Y. Tsao, J. Li, C. H. Lee, and S. Nakamura, "Soft margin estimation on improving environment structures for ensemble speaker and speaking environment modeling," in *Proc. IUCS*, 2009. 121

[196] M. Van Segbroeck and H. Van Hamme, "Vector-quantization based mask estimation for missing data automatic speech recognition," in *Proc. ICSLP*, 2007. 121

[197] A. Ragni and M. Gales, "Derivative kernels for noise robust ASR," in *Proc. ASRU.* IEEE, 2011, pp. 119–124. 121

[198] L. Lu, A. Ghoshal, and S. Renals, "Noise adaptive training for subspace Gaussian mixture models," in *Proc. Interspeech.* ISCA, 2013. 121

[199] F. Flego and M. Gales, "Discriminative adaptive training with VTS and JUD," in *Proc. ASRU.* IEEE, 2009, pp. 170–175. 121

[200] Y. Q. Wang and M. Gales, "Speaker and noise factorization for robust speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 7, pp. 2149–2158, 2012. 121