Title
Neural network based representation learning and modeling for speech and speaker recognition

Author
Guo, Jinxi

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Neural network based representation learning and modeling for speech and speaker recognition

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Jinxi Guo

2019

ABSTRACT OF THE DISSERTATION

Neural network based representation learning
and modeling for speech and speaker recognition

by

Jinxi Guo
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2019
Professor Abeer A. H. Alwan, Chair

Deep learning and neural network research has grown significantly in the fields of automatic speech recognition (ASR) and speaker recognition. Compared to traditional methods, deep learning-based approaches are more powerful in learning representation from data and building complex models. In this dissertation, we focus on representation learning and modeling using neural network-based approaches for speech and speaker recognition.

In the first part of the dissertation, we present two novel neural network-based methods to learn speaker-specific and phoneme-invariant features for short-utterance speaker verification. We first propose to learn a spectral feature mapping from each speech signal to the corresponding subglottal acoustic signal which has less phoneme variation, using deep neural networks (DNNs). The estimated subglottal features show better speaker-separation ability and provide complementary information when combined with traditional speech features on speaker verification tasks. Additional, we propose another DNN-based mapping model, which maps the speaker representation extracted from short utterances to the speaker representation extracted from long utterances of the same speaker. Two non-linear regression models using an autoencoder are proposed to learn this mapping, and they both improve speaker verification performance significantly.

In the second part of the dissertation, we design several new neural network models which take raw speech features (either complex Discrete Fourier Transform (DFT) features or raw

waveforms) as input, and perform the feature extraction and phone classification jointly. We first propose a unified deep Highway (HW) network with a time-delayed bottleneck layer (TDB), in the middle, for feature extraction. The TDB-HW networks with complex DFT features as input provide significantly lower error rates compared with hand-designed spectrum features on large-scale keyword spotting tasks. Next, we present a 1-D Convolutional Neural Network (CNN) model, which takes raw waveforms as input and uses convolutional layers to do hierarchical feature extraction. The proposed 1-D CNN model outperforms standard systems with hand-designed features. In order to further reduce the redundancy of the 1-D CNN model, we propose a filter sampling and combination (FSC) technique, which can reduce the model size by 70% and still improve the performance on ASR tasks.

In the third part of dissertation, we propose two novel neural-network models for sequence modeling. We first propose an attention mechanism for acoustic sequence modeling. The attention mechanism can automatically predict the importance of each time step and select the most important information from sequences. Secondly, we present a sequence-to-sequence based spelling correction model for end-to-end ASR. The proposed correction model can effectively correct errors made by the ASR systems.

The dissertation of Jinxi Guo is approved.

Alan J. Laub

Yingnian Wu

Christina Panagio Fragouli

Abeer A. H. Alwan, Committee Chair

University of California, Los Angeles

2019

To my family.

TABLE OF CONTENTS

1 Introduction **1**

1.1 Overview and motivation 1

1.2 Deep neural networks 2

1.2.1 DNN architecture and optimization methods 2

1.2.2 Convolutional Neural Networks 4

1.2.3 Recurrent Neural Networks 4

1.3 Speech processing and feature extraction 6

1.4 Speaker Verification 6

1.4.1 I-vector/PLDA system 6

1.4.2 Speech corpora 9

1.4.3 Evaluation metrics 10

1.5 Automatic Speech Recognition 11

1.5.1 DNN-HMM based speech recognition system 11

1.5.2 End-to-end speech recognition system 15

1.5.3 Speech corpora 17

1.5.4 Evaluation metrics 18

1.6 Dissertation Outline 18

2 Learning speaker representations from short utterances **19**

2.1 Introduction 19

2.2 Related work 20

2.3 Learning speaker-specific and phoneme-invariant subglottal acoustic features 20

2.3.1 Subglottal acoustic features 20

2.3.2 Proposed estimation method 21
2.3.3 Estimation experiments 22
2.3.4 Speaker verification experiments 25
2.4 Learning non-linear mapping from short-utterance to long-utterance i-vectors 28
2.4.1 The effect of utterance durations on i-vectors 28
2.4.2 DNN-based i-vector mapping 29
2.4.3 Experimental set-up 35
2.4.4 Evaluation of proposed i-vector mapping methods 38
2.4.5 Speaker verification experiments 40
2.5 Conclusion 50

3 Joint feature learning and acoustic modeling for automatic speech recognition 51

3.1 Introduction 51
3.2 Related work 51
3.3 Feature learning in the frequency domain 52
3.3.1 Baseline Wake-word Detection System 52
3.3.2 DFT-Input Highway networks 54
3.3.3 Experiments and results 57
3.4 Feature learning from raw waveforms 62
3.4.1 CNN-based acoustic modeling using raw waveforms 62
3.4.2 Filters learned from raw waveforms 63
3.4.3 Filter sampling and combination CNN 65
3.4.4 Experiments and results 67
3.5 Conclusion 72

4 Sequence modeling for acoustic and language models 73

4.1 Introduction 73

4.2 Related work 74

4.3 Learning attention mechanism for acoustic modeling 74

4.3.1 Acoustic scene classification 74

4.3.2 Neural network architectures 75

4.3.3 Attention mechanisms for sequence modeling 76

4.3.4 Evaluation set-up 79

4.3.5 Experimental results 80

4.3.6 Analysis of learned attention weights 84

4.4 Learning a spelling correction model for end-to-end speech recognition 85

4.4.1 Motivation 85

4.4.2 Baseline LAS model 86

4.4.3 Approaches of utilizing text-only data 87

4.4.4 Spelling correction model 87

4.4.5 Experimental setup 90

4.4.6 Experimental results 93

4.4.7 Error analysis 96

4.5 Conclusion 97

5 Summary and future work 99

5.1 Summary 99

5.2 Future work 101

References 102

LIST OF FIGURES

1.1 An LSTM block. At time step t, C_t and C_{t-1} represent the current and previous cell states, h_t and h_{t-1} represent the current and previous hidden states, f_t represents forget gate, i_t represents input gate, and o_t represents output gate. . . 5

1.2 A standard ASR system. 11

1.3 Components of the LAS model. 16

2.1 Spectrograms of three vowels by a female speaker to compare within-speaker variability of microphone speech (top panel) and subglottal acoustics (bottom panel). Note that the subglottal acoustics don't vary much. Data are sampled from the recordings of a female speaker in the WashU-UCLA corpus. 21

2.2 Histogram of the correlation coefficient of the actual and estimated subglottal Mel-filterbank coefficients for each frame in the validation dataset. 23

2.3 Block diagram of the proposed framework. 26

2.4 Distribution of active speech length of 40000 long utterances in SRE and SWB datasets. 29

2.5 DNN_1: two-stage training of i-vector mapping. Left schema corresponds to the first-stage pre-training. A short-utterance i-vector w_s and a corresponding long-utterance i-vector w_l are first concatenated into z. Then z is fed into an encoder $f(.)$ to generate the joint embedding h. h is passed to the decoder $g(.)$ to generate the reconstructed $\hat{z}$, which is expected to be a concatenation of a reconstructed $\hat{w}_s$ and $\hat{w}_l$. Right schema corresponds to the second-stage fine-tuning. The pre-trained weights in the first stage is used to initialize the supervised regression model from w_s to w_l. After training, the estimated i-vector $\hat{w}_l$ is used for evaluation. 31

2.6 Residual block. An input x is first passed into two hidden layers to get $F(x)$ and it also goes through a short-cut connection, which skips the hidden layers. The final output of the residual block is a summation of $F(x)$ and x. 32

2.7 DNN_2: single-stage training of i-vector mapping. A short-utterance i-vector w_s is passed to an encoder and the output of the encoder is first used to generate the estimated long-utterance i-vector $\hat{w}_l$ and it is also fed into a decoder to generate the reconstructed short-utterance i-vector $\hat{w}_s$. The two tasks are optimized jointly. 32

2.8 I-vector mapping with additional phoneme information. A short-utterance i-vector w_s is concatenated with a phoneme vector p to generate the estimated long-utterance i-vectors $\hat{w}_l$. 34

2.9 EER as a function of reconstruction loss α for DNN_2. 44

2.10 DET curves for the mapping results of I-vector_GMM and I-vector_DNN systems under 10 s-10 s conditions of NIST SRE10 database. Left figure corresponds to female speakers and right one corresponds to male speakers. 48

2.11 Distribution of active speech length of truncated short utterances in the SITW database. 49

3.1 HMM-based Keyword Spotting. 53

3.2 Baseline WW DNN with the LFBE feature. 54

3.3 Whole WW Highway DNN with the DFT input. 56

3.4 DET curves of LFBE DNN, LFBE HW, DFT TDB-DNN, DFT TDB-HW using different amounts of training data 59

3.5 AUCs calculated from Figure 3.4. 59

3.6 DET curves of LFBE HW, DFT TDB-HW, Audio TDB-HW, LPS TDB-HW using different amounts of training data 61

3.7 AUCs calculated from Figure 3.6 61

3.8 Filters learned from each conv layer (order from top to bottom). Each row represents a layer and 6 different filters from that layer are shown. 64

3.9 Widthwise filter sampling in space Φ. 65

3.10 Depthwise filter sampling in space Φ. 65

4.1 The CLDNN framework 76
4.2 Standard BLSTM layer (left), attention-based BLSTM layer (right) 77
4.3 The mel-filterbank features (bottom) with time-aligned attention scores (top) for the sample segment recorded in a cafe/restaurant 83
4.4 The mel-filterbank features (bottom) with time-aligned attention scores (top) for the sample segment recorded in a park 83
4.5 Spelling Correction model architecture. 89
4.6 Example attention weights from the SC model. 94

LIST OF TABLES

2.1 J-ratio, a measure of class separation for different feature values. Features were extracted from isolated vowel recordings of speech and subglottal acoustics, for all the 50 male and female adult speakers in the WashU-UCLA corpus. 24

2.2 EERs for the MFCC baseline system and the fused system on the NIST SRE 08 truncated 10sec-10sec and 5sec-5sec evaluation tasks. The relative improvements in EERs are also shown. 27

2.3 Mean variance of long and short utterances (from the SRE and Switchboard datasets) 29

2.4 Datasets used for developing I-vector_GMM and I-vector_DNN systems 35

2.5 Square Euclidean distance (D_{sl}) between short and long utterance i-vector pairs from SRE10 before and after mapping. 39

2.6 J-ratio for short-utterance i-vectors from SRE10 before and after mapping. . . . 39

2.7 Baseline results for I-vector_GMM and I-vector_DNN systems under full-length and short-length utterances conditions reported in terms of EER, Relative Improvement (Rel Imp), and minDCF on NIST SRE10 database. 41

2.8 Results for baseline (I-vector_DNN), matched-length PLDA training, LDA dimension reduction, DNN direct mapping and proposed DNN mapping in the 10 s-10 s condition of NIST SRE10 database. 42

2.9 DNN-based mapping results using DNNs with different depths in the 10 s-10 s condition of NIST SRE10 database. 45

2.10 DNN-based mapping results with additional phoneme information in the 10 s-10 s condition of NIST SRE10 database. 46

2.11 DNN-based mapping results with different utterance durations on NIST SRE10 database. 47

2.12 Results for I-vector_GMM and I-vector_DNN systems in the 10 s-10 s conditions of NIST SRE10 database. 48

2.13 DNN-based mapping results on SITW using arbitrary durations of short utterances. 49

3.1 Proposed 1-D CNN structure (the last column shows the number of parameters for each layer). 63

3.2 Baseline comparison: different features and neural network structures. 69

3.3 Filter sampling results for raw waveform CNNs. 'c^*' indicates that the convolutional layers have half the number of filters in each layer compared with the baseline CNN. 'cw' and 'cd' represent compressing the parameters in the convolution layers using widthwise and depthwise filter sampling, respectively. 'fw' represents performing widthwise filter sampling in the fully connected layers. '/4' means reducing the number of parameters by a factor of 4. 70

3.4 Filter sampling and combination results for raw waveform CNNs. lin-MxN means doing linear combination using MxN different scalars. 71

4.1 Classification accuracy (%) of CNNs and CLDNNs 80

4.2 Classification accuracy (%) of CBLDNNs and different attention models 81

4.3 Classification accuracy (%) of CBLDNNs, attention model and 3 combined models 82

4.4 Word error rates (WERs) on LibriSpeech "clean" sets comparing different techniques for incorporating text-only training data. Numbers in parentheses indicate the number of input hypotheses considered by the corresponding model. 93

4.5 Oracle WER before and after applying the SC model. 95

4.6 Word error rates (WERs) on LibriSpeech "clean" sets comparing different techniques for incorporating text-only training data. Numbers in parentheses indicate the number of input hypotheses considered by the corresponding model. 95

4.7 WER comparison on a real audio and TTS dev sets. 96

4.8 Word error rates (WERs) on LibriSpeech "clean" sets comparing different techniques for incorporating text-only training data. Numbers in parentheses indicate the number of input hypotheses considered by the corresponding model. 96

4.9 LAS + SC + LM rescore Wins. LAS + LM rescore (in bold) 97

ACKNOWLEDGMENTS

This dissertation would not have been possible without the support of many people. I would like to first thank my advisor Prof. Abeer Alwan, for being a great mentor over the years and providing me with gracious support. Her vision for speech processing motivated me to start my PhD research. She has given me great freedom to pursue the research that I would like to work on. I am also indebted to Prof. Yingnian Wu for his insightful suggestions to my research. I would like to also thank the rest of my committee member, Prof. Alan Laub and Prof. Christina Fragouli, for their invaluable advice and comments on my dissertation.

I am also grateful to have worked with Dr. Ning Xu at Snap. Ning's sound knowledge and nice personality made my internship and our further collaborations a great pleasure for me. Many thanks to Dr. Kenichi Kumatani, I have enjoyed and benefited a lot from various discussions with him during the internship at Amazon Alexa team. Sincere appreciation also goes to Dr. Tara Sainath and Dr. Ron Weiss for their supervision during my summer internship at Google. Their mentorship has immensely impacted my thought process both as a person and as a researcher.

My heartfelt thanks also go to my former and current labmates at SPAPL. Many thanks to Jom, Gang, Harish, Lee, and Anirudh for helping me find my bearings initially. Without their help, I would not have been able to start my life and work easily at UCLA. I am also grateful to Soo, Amber, Gary, Vijay, Kaan, Rohit, Ruochen, Hitesh and Angli for many enjoyable discussions and collaborations. Many thanks to Kailun, Yang, Kaiyuan, Usha, Deepak for the collaborations directly related to this dissertation. My work would not have been possible without their support.

I also extend my gratitude to the administrative staff in the department (especially Deeona and Ryo) for their patience and tremendous help throughout my entire PhD period. Life would have been a lot tougher without them. I am truly blessed with my wonderful friends in Swipe group. I am thankful for their company, support and many enjoyable chats.

Lastly, but very importantly, a million thanks to my family members for their unconditional love and enormous support. My father Hongfu Guo and my mother Shumei Du have made a lot of sacrifices to ensure that I could focus on my research and career goals. A big thank you to Xiaoxi for her love, understanding and company.

VITA

2009-2013	B.E., Electronics and Information Engineering, Xi'an Jiaotong University, China.
2013-2015	M.S., Electrical Engineering, University of California, Los Angeles, USA.
2013–2019	Graduate Research Assistant, Electrical and Computer Engineering Department, UCLA.
2014–2017	Teaching Assistant, Electrical and Computer Engineering Department, UCLA.
2015	Research Internship, Qualcomm.
2016	Research Internship, Snap Inc.
2017	Research Internship, Amazon.
2018	Research Internship, Google.

PUBLICATIONS

Jinxi Guo, Tara Sainath and Ron Weiss, "A spelling correction model for end-to-end speech recognition," ICASSP, 2019.

Jinxi Guo, Ning Xu, Kailun Qian, Yang Shi, Kaiyuan Qian, Yingnian Wu and Abeer Alwan, "Deep Neural Network based i-Vector mapping for Speaker Verification using Short Utterances," Speech Communication, 2018.

Jinxi Guo, Ning Xu, Xin Chen, Yang Shi, Kaiyuan Qian, Yingnian Wu and Abeer Alwan, "Filter sampling and combination CNN (FSC-CNN): a compact CNN model for small-footprint ASR acoustic modeling using raw waveforms," Interspeech, 2018.

Jinxi Guo, Kenichi Kumatani, Ming Sun, Minhua Wu, Anirudh Raju, Nikko Strom and Arindam Mandal, "Time-delayed bottleneck Highway Networks using a DFT feature for Robust Keyword Spotting," ICASSP, 2018.

Jinxi Guo, Ruochen Yang, Harish Arsikere and Abeer Alwan, "Robust speaker identification via fusion of Subglottal Resonances and Cepstral Features," Journal of Acoustic Society of American, 2017.

Jinxi Guo, Usha Nookala and Abeer Alwan, "CNN-based joint mapping of short and long utterance i-vectors for speaker verification using short utterances," Interspeech, 2017.

Jinxi Guo, Ning Xu, Li-Jia Li and Abeer Alwan, "Attention based CLDNNs for short-duration acoustic scene classification," Interspeech, 2017.

Jinxi Guo, Gary Yeung, Deepak Muralidharan, Harish Arsikere, Amber Afshan and Abeer Alwan, "Speaker verification using short utterances with DNN-based estimation of subglottal acoustic features," Interspeech, 2016.

Jinxi Guo, Rohit Paturi, Gary Yeung, Steven M Lulich, Harish Arsikere and Abeer Alwan, "Age-dependent height estimation and speaker normalization for children's speech using the first three subglottal resonances," Interspeech, 2015.

Jinxi Guo, Angli Liu, Harish Arsikere, Abeer Alwan and Steven M Lulich, "The relationship between the second subglottal resonance and vowel class, standing height, trunk length, and F0 variation for Mandarin speakers," Interspeech, 2014.

CHAPTER 1

Introduction

1.1 Overview and motivation

Automatic Speech Recognition (ASR) and Speaker Recognition (SV) are two very important applications. There have been extensive studies conducted of both tasks over the past decade. However, there are still challenging problems.

For SV, state-of-the-art text-independent systems exhibit satisfactory performance with adequately long speech data (e.g. more than 30 s), but the performance degrades rapidly when only limited data are available [KVD11]. The degraded performance is due to the large phonetic (context) variation between different short utterances. The requirement of significant amounts of speech for training or evaluation, especially with large intersession variability has limited the potential of SV's widespread implementations in practice. To address this issue, a range of techniques has been studied on different aspects of this problem [PSS17, DP18]. However, learning a speaker-specific and phoneme-invariant representation from short utterances is still very challenging.

For ASR, constructing an appropriate feature representation and designing an appropriate phone classifier for these features have often been treated as separate problems in the speech recognition community. One drawback of this approach is that the designed features might not be best for the classification objective at hand. In order to solve this problem, Deep Neural Networks (DNNs), and their variants, can be used to perform feature extraction jointly with classification. However, for a long time, the most popular features to train DNNs remain the log-mel features. The mel filter bank is inspired by auditory evidence of how humans perceive speech signals. Such a filter bank may not always be the best filter

bank in a statistical modeling framework where the end goal is word error rate. To address this issue, there have been various attempts [BR15, HWW15, SWS15] to use a simpler feature representation (e.g. waveforms) with neural networks to learn feature representation jointly with the rest of the network. However, only a few studies have shown improvement over the log-mel trained model.

Moreover, sequence modeling is one of the key problems for speech modeling. Traditional systems use Hidden Markov Models (HMMs) to model speech sequences. In recent years, Recurrent Neural Networks (RNNs), and their variants (e.g. Long Short Term Memory (LSTM) networks), have shown superior performance on various speech sequence modeling tasks compared with HMMs. However, the basic RNN models still need to be modified in order to perform modeling on advanced tasks, such as information selection and sequence-to-sequence modeling (e.g. end-to-end ASR).

In this dissertation, in order to address the aforementioned problems and challenges in ASR and SV, several DNN based approaches are proposed and discussed. Compared to traditional methods, deep learning-based approaches are more powerful in learning representation from data and building complex models. Therefore, we present novel neural network-based approaches to perform representation learning and modeling.

In this chapter, we will review some background knowledge for deep neural networks, speech processing, speaker verification and automatic speech recognition systems.

1.2 Deep neural networks

1.2.1 DNN architecture and optimization methods

A deep neural netowrk (DNN) is a conventional multilayer perceptron (MLP) with many (often more than two) hidden layers. In all $L-1$ hidden layers:

$$\mathbf{v}^l = f(\mathbf{W}^l \mathbf{v}^{l-1} + \mathbf{b}^l), 1 \leq l \leq L-1, \tag{1.1}$$

where $\mathbf{v}^l$, $\mathbf{W}^l$, $\mathbf{b}^l$ are the activation vector, the weight matrix and the bias vector, respectively at layer l. $\mathbf{W}^l$ is a $N_l \times N_{l-1}$ matrix, where N_l represents the number of neurons at layer l. In many applications, the sigmoid function, the hyperbolic tangent function or the rectified linear unit (ReLU) function is used as the activation function.

The output layer L needs to be chosen based on the tasks in hand [YD16]. For a regression task, a linear layer is typically used to generate the output vector $\mathbf{v}^L$ with dimension N_L:

$$\mathbf{v}^L = \mathbf{W}^L \mathbf{v}^{L-1} + \mathbf{b}^L \tag{1.2}$$

For multi-class classification tasks, each output neuron represents a class $i \in \{1, ..., C\}$, where $C = N_L$ is the number of classes. The value of the ith output neuron v_i^L represents the probability $P_{dnn}(i|o)$ that the observation vector o belongs to class i. To be a valid multinomial probability distribution, the output vector $\mathbf{v}^L$ should satisfy the requirements that $v_i^L > 0$ and $\sum_{i=1}^{C} v_i^L = 1$. This can be done by normalizing the excitation with a softmax function:

$$v_i^L = P_{dnn}(i|\mathbf{o}) = \text{softmax}_i(\mathbf{z}^L) = \frac{e^{z_i^L}}{\sum_{j=1}^{C} e^{z_j^L}} \tag{1.3}$$

The model parameters $\{\mathbf{W}, \mathbf{b}\}$ in a DNN are unknown and need to be estimated from training samples for each task. In order to perform parameter estimation, a training criterion and a learning algorithm need be specified.

The training criterion should be highly correlated to the final goal of the task. There are two popular empirical training criteria in DNN model learning. For regression task, the mean square error (MSE) criterion is typically used:

$$J_{MSE}(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) = \frac{1}{2}(\mathbf{v}^L - \mathbf{y})^T(\mathbf{v}^L - \mathbf{y}) \tag{1.4}$$

where $\{\mathbf{W}, \mathbf{b}\}$ are the model parameters, $\mathbf{o}$ is the observation, and $\mathbf{y}$ is the corresponding output vector.

For classification tasks, $\mathbf{y}$ is a probability distribution and the cross-entropy (CE) criterion is often used:

$$J_{CE}(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) = -\sum_{i=1}^{C} y_i \log v_i^L \tag{1.5}$$

Learning neural network parameters is usually performed using back propagation with stochastic gradient descent (SGD) and momentum. For SGD, the true gradient is estimated by the gradient of a small subset of the training examples, called a mini-batch.

1.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) [LBB98] are specialized kind of neural networks used to process data that has a known grid-like topology. There are typically three processing stages for CNNs: convolution stage, detector stage and pooling stage.

The convolution stage is the core part. Unlike fully-connected layers, convolutional layers take into account the input topology, and introduce highly restricted connections to model local information. CNNs use filters (kernels) to carry out the convolution operation over the input, and generate a set of linear activations called feature maps.

In the detector stage, each linear activation generated from convolution stage, is run through a nonlinear activation function, such as ReLU function. In the pooling stage, a pooling function is used to replace the detection-stage output at a certain location with a summary statistic of the nearby outputs. For example, the max pooling operation reports the maximum output within a rectangular neighborhood. Average pooling is another popular function. Pooling helps to make the representation approximately invariant to small translations of the input.

1.2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) [WZ89] are variants of feed-forward neural networks, which contain feedback loops that feed activations not only to the next layer but also as the input to the current layer at the next time step. This design enables the network to handle

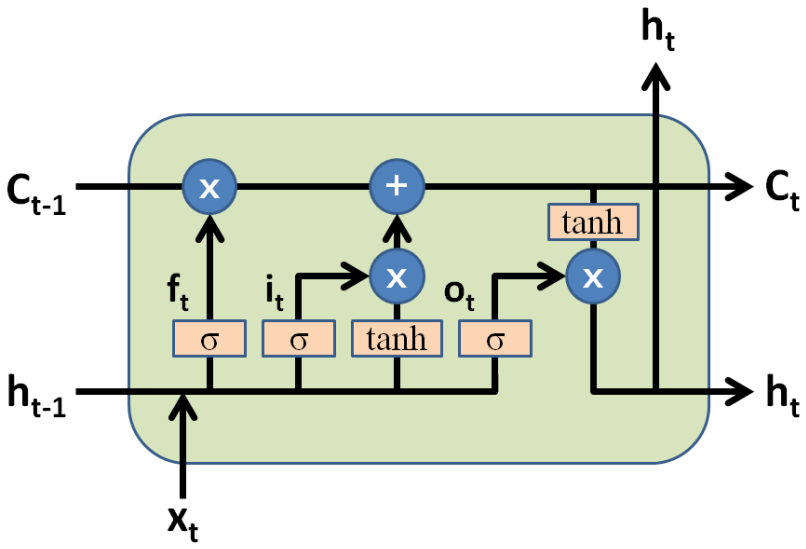


Figure 1.1: An LSTM block. At time step t, C_t and C_{t-1} represent the current and previous cell states, h_t and h_{t-1} represent the current and previous hidden states, f_t represents forget gate, i_t represents input gate, and o_t represents output gate.

variable-length sequences, and more important, RNNs consider all contexts from the past to make a decision about the current frame, which is a desirable property since contextual information plays an important role in sequence modeling.

One challenge in training RNNs is that long-term dependencies cause vanishing gradients, i.e. the magnitude of gradients turns to be very small in long sequences, making recurrent network architectures difficult to optimize. To address this issue, the Long Short Term Memory (LSTM) block was proposed in [HS97] to replace the traditional RNN block.

1.2.3.1 LSTM RNNs

A diagram of an LSTM block is shown in Figure 1.1. A key modification in the LSTM is the memory cell, which maintains history information through time. Nonlinear gates modeled by sigmoid function are introduced to control the information flow. Forget gate f_t controls the information from previous cell state C_{t-1}. Input gate i_t controls the new information generated by non-linear tanh layer. Two information are combined to create an updated cell state C_t. Finally, output gate o_t is used to control the final information outputted to hidden

state h_t.

1.3 Speech processing and feature extraction

The raw representation of speech data is a continuous waveform. To effectively perform recognition, speech waveform is first preprocessed using a feature extractor. A standard feature extractor processes segments of speech waveform every 10 ms using a sliding window of 25 ms. These segments are then converted into feature vectors by using different feature extraction methods. Examples of speech features are Mel-filterbank features, Mel-frequency cepstral coefficients (MFCCs) [DM80] and perceptual linear predictions (PLP) [Her90].

1.4 Speaker Verification

Speaker Verification (SV) is to verify, given a voice sample and an associated claim, if the talker is indeed the one he or she claims to be. SV has found wide applications in telephone-based financial transactions, information retrieval from speech databases, voice-based user authentication, etc. For SV, the speech input can be either totally unconstrained (text independent) or constrained to be a known phrase (text dependent). This dissertation considers the more challenging text-independent case. The success of SV systems depends on extracting and modeling speaker-dependent characteristics from speech signals which can effectively distinguish one talker from another.

1.4.1 I-vector/PLDA system

The state-of-the-art text-independent speaker verification system is based on the i-vector framework [DKD10]. In these systems, a universal background model (UBM) is used to collect sufficient statistics for i-vector extraction, and a probabilistic linear discriminant analysis (PLDA) backend is adopted to obtain the similarity scores between i-vectors. There are two different ways to model a UBM: using unsupervised-trained Gaussian mixture models (GMMs) or using a deep neural network (DNN) trained as a phoneme classifier. Therefore, we

will introduce both the I-vector_GMM and I-vector_DNN systems as well as PLDA modeling.

1.4.1.1 I-vector_GMM system

For an I-vector_GMM system, both speaker model and speaker-independent background model (i.e. UBM) are represented by GMMs. By concatenating the means of these Gaussian mixtures, GMM supervectors could be generated to represent different speakers. The i-vector representation here is based on the total variability modeling concept which assumes that speaker- and channel- dependent variabilities reside in a low-dimensional subspace, represented by the total variability matrix T. Mathematically, the speaker- and channel-dependent GMM supervector s can be modeled as:

$$s = s' + Tw \tag{1.6}$$

where s' is the speaker- and channel-independent supervector generated from UBM, T is a rectangular matrix of low rank and w is a random vector called the i-vector which has a standard normal distribution $\mathcal{N}(0, I)$.

In order to learn the total variability subspace, Baum-Welch statistics are computed for a given utterance, and are defined as:

$$N_c = \sum_t P(c|y_t, \Omega) \tag{1.7}$$

$$F_c = \sum_t P(c|y_t, \Omega) y_t \tag{1.8}$$

where N_c and F_c represents the zeroth and first order statistics, y_t is the feature sample at time index t, Ω represent the UBM of C mixture components, $c = 1, ..., C$ is the Gaussian index and $P(c|y_t, \Omega)$ corresponds to the posterior of mixture component c generating the vector y_t.

1.4.1.2 I-vector_DNN system

As mentioned in the previous subsection, for an I-vector_GMM system, the posterior of mixture component c generating the vector y_t is computed with a GMM acoustic model trained in an unsupervised fashion (i.e. with no phonetic labels).

$$P(c|y_t, \Omega) \Rightarrow P(c|y_t, \Theta) \tag{1.9}$$

However, recently, inspired by the success of DNN acoustic models in automatic speech recognition (ASR), [LSF14] proposed a method which uses DNN senone (cluster of context-dependent triphones) posteriors to replace the GMM posteriors as illustrated in Eq.1.9, which leads to significant improvement in speaker verification. Θ represents the trained DNN model for senone classification.

The senone posterior approach uses ASR features to compute the class soft alignment and the standard speaker verification features for sufficient statistic estimation. Once sufficient statistics are accumulated, the training procedure is the same as in the previous section. In this dissertation, we use a state-of-the-art time delay neural network (TDNN) as in [PPK15] to train the ASR acoustic model.

1.4.1.3 PLDA modeling

PLDA is a generative model of i-vector distributions for speaker verification. In this dissertation, we use a simplified variant of PLDA, termed as G-PLDA [KSO13], which is widely used by researchers. A standard G-PLDA assumes that the i-vector w_i is represented by:

$$w_i = r + Ux + \epsilon_i \tag{1.10}$$

where r is the mean of i-vectors, U defines the between-speaker subspace, and the latent variable x represents the speaker identity and is assumed to have standard normal distribution. The residual term ϵ_i represents the within-speaker variability, which is normally distributed with zero mean and full covariance Σ'.

PLDA based i-vector system scoring is calculated using the log likelihood ratio (LLR) between a target and test i-vectors, denoted as w_{target} and w_{test}. The likelihood ratio can be calculated as follows:

$$LLR = \log \frac{P(w_{target}, w_{test}|H_1)}{P(w_{target}|H_0)P(w_{test}|H_0)} \tag{1.11}$$

where H_1 and H_0 denote the hypothesis that two i-vectors represent the same speaker, and different speakers, respectively.

1.4.2 Speech corpora

Four speech corpora are used for the SV experiments in this dissertation. A brief overview of each speech corpus will be described in the following subsections.

1.4.2.1 NIST SRE

The NIST Speaker Recognition Evaluation (SRE) is a series of evaluations designed for text independent speaker recognition research [DPM00]. Dataset we used in this dissertation includes SRE 2004, 2005, 2006, 2008 and 2010. SRE datasets contain thousands of hours of speech collected from several thousands speakers. They also cover a variety of channels and conditions, including telephone and microphone speech.

1.4.2.2 Switchboard-2

Switchboard-2 (SWB-2) data was collected by Linguistic Data Consortium (LDC) in support of speaker recognition projects, and it contains several thousands telephone conversations. SWB-2 Phase II was used in this dissertation.

1.4.2.3 SITW

The Speakers in the Wild (SITW) speaker recognition database [MFC16] contains hand-annotated speech samples from open-source media for the purpose of benchmarking text-

independent speaker recognition technology on single and multi-speaker audio acquired across unconstrained or wild conditions. The database consists of recordings of 299 speakers, with an average of eight different sessions per person. This data contains real noise, reverberation, intraspeaker variability and compression artifacts.

1.4.2.4 WashU-UCLA

WashU-UCLA dataset [ALS15] contains time-synchronized recordings of speech and subglottal acoustics obtained from 25 male and 25 female adult native speakers of American English. Recordings of time-synchronized speech and subglottal acoustics were made with a microphone and an accelerometer, respectively.

1.4.3 Evaluation metrics

For SV, each test utterance and enrolled speaker model comparison is referred to as a trial. As a binary classification task, given a threshold vale, false acceptance rate (R_{FA}) and false rejection rate (R_{FR}) defined in 1.12 and 1.13 can be used to evaluate performance. A detection error tradeoff (DET) curve can be created by plotting R_{FA} versus R_{FR} given different threshold values.

$$R_{FA} = \frac{\text{Number of False Acceptance}}{\text{Number of impostor accesses}} \tag{1.12}$$

$$R_{FR} = \frac{\text{Number of False Rejection}}{\text{Number of target accesses}} \tag{1.13}$$

In order to quantify the performance, two most popular metrics (i.e. equal error rate (EER) and minimum detection cost function (minDCF)) are calculated based on the DET curve. EER corresponds to the point on DET curve where R_{FA} and R_{FR} are equal. The detection cost function is defined in 1.14 as a weighted sum of R_{FA} and R_{FR},

$$DCF = C_{FR} \times P_{target} \times R_{FR} + C_{FA} \times P_{impostor} \times R_{FA} \tag{1.14}$$

where C_{FR} and C_{FA} are cost defined by the evaluation plans, and P_{target} and $P_{impostor}$ are the prior probability of the specified target or impostor speaker. minDCF is the minimum value of DCF.

1.5 Automatic Speech Recognition

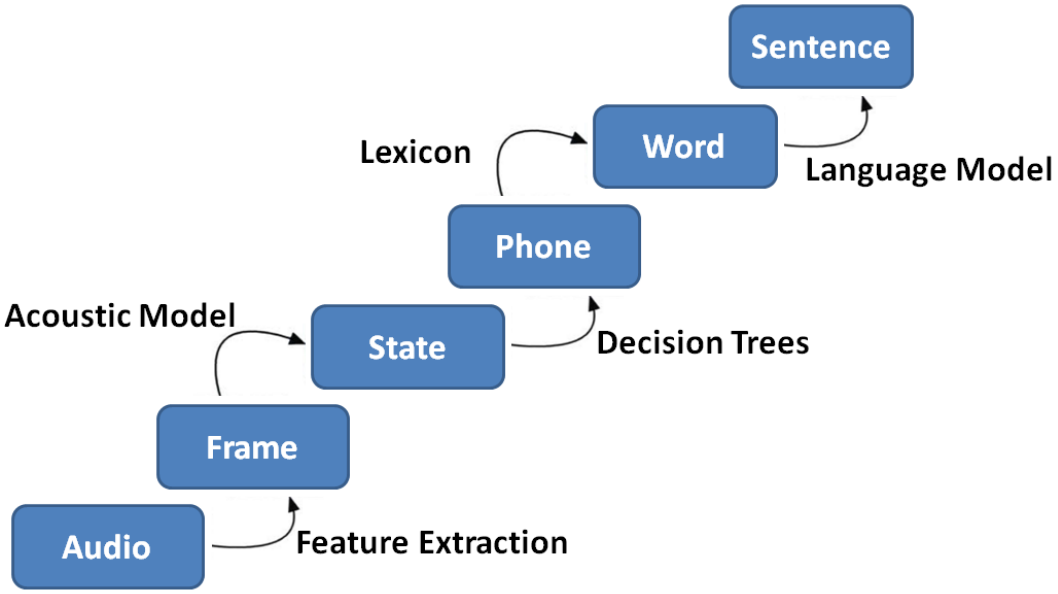


Figure 1.2: A standard ASR system.

1.5.1 DNN-HMM based speech recognition system

Automatic Speech Recognition (ASR) is the process of transcribing human speech into text automatically using machines. Figure 1.2 shows the components of a typical speech recognition system, which we will describe briefly. An input audio recording is first processed using a feature extractor to generate acoustic features, and then the extracted features are used by an Acoustic Model (AM). The acoustic model is a statistical model of the features conditioned on different spoken sound classes. Sound classes are usually represented by states in the Hidden Markov Model (HMM). Acoustic models are typically represented by Gaussian

mixture models (GMMs) or neural network models, e.g. a Deep Neural Network (DNN), Convolutional Neural Network (CNN), or Recurrent Neural Network (RNN).

After acoustic modeling, Decision Trees map sub-word units (the hidden states generated by an HMM) to phoneme sequences. Then, the Lexicon, or Pronunciation Model, maps a sequence of phonemes to a word.

The Language Model (LM) is a statistical model giving the probability of word sequences independent of the acoustics. The standard language models used in ASR are n-grams. N-gram models represent the probability of generating the next word given the previous N-1 words. The log probability from the language model is typically linearly combined with the acoustic model score, and then fed into the decoder.

The decoder combines the probabilities from the AM and LM to search for the best word sequences under the constraints of the pronunciation model. Most decoders use a combination of dynamic programming and beam-searching to generate a subset of plausible candidates, and score them at the same time. Modern decoders are usually implemented using Weighted Finite State Transducers (WFSTs) for efficient searching.

We now describe the AM and LM parts in a standard state-of-the-art DNN-HMM speech recognition system with formal mathematical notation.

1.5.1.1 Acoustic model

Given the observation sequence (feature frames), $\mathbf{X}$, extracted from a speech waveform, we want to find the best word sequence $\hat{\mathbf{W}}$ that maximizes the posterior probability $p(\mathbf{W}|\mathbf{X})$:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}}\, p(\mathbf{W}|\mathbf{X}) \tag{1.15}$$

We can decompose this probability into two terms using Bayes's Rule, an acoustic model, $p_{AM}(\mathbf{X}|\mathbf{W})$, and a language model, $p_{LM}(\mathbf{W})$:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}}\, p_{AM}(\mathbf{X}|\mathbf{W}) p_{LM}(\mathbf{W}) \tag{1.16}$$

In the traditional Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) paradigm, the output probabilities are generated by the GMM, and the sequential property of speech is modeled by the HMM. The hidden states, $\mathbf{S}$, in the HMM typically represent a subword or phonetic segmentation of a word. Therefore we would change $p_{AM}(\mathbf{X}|\mathbf{W})$ in Eq.1.16 to:

$$\begin{aligned} p_{AM}(\mathbf{X}|\mathbf{W}) &= \sum_{\mathbf{S}} p(\mathbf{X}, \mathbf{S}|\mathbf{W}) \\ &= \sum_{\mathbf{S}} p(\mathbf{X}|\mathbf{S}, \mathbf{W}) p(\mathbf{S}|\mathbf{W}) \\ &= \sum_{\mathbf{S}} \prod_{t} p(x_t|s_t) p(\mathbf{S}|\mathbf{W}) \end{aligned} \tag{1.17}$$

where x_t and s_t are the observation and hidden state at time t, respectively. The first term $p(x_t|s_t)$ in Eq.1.17 is modeled by GMMs, which evaluate the likelihood of a speech observation x_t being generated by a hidden state s_t.

In practice, HMM hidden states are typically modeled by 3-state triphones. A triphone is a phone with a left and right context. Each triphone is usually modeled by 3 left-to-right states to handle transient acoustic dynamics. Systems that model triphones are usually referred to as having context dependent (CD) models, while systems that model just single phonemes without any context use context independent (CI) models.

The parameters of the GMM-HMM acoustic model can be estimated using Maximum Likelihood Estimation (MLE) and the Forward-Backward algorithm. Details can be found in [RJR93].

A DNN-HMM system usually starts with a baseline GMM-HMM speech recognizer that computes frame-level output target labels. This is usually done by force aligning the transcription with the input speech by the GMM-HMM recognizer. Then, a DNN is used to model the posterior probability of an acoustic frame x_t being in state s_t:

$$p(s_t|x_t) = \mathrm{DNN}(x_t) \tag{1.18}$$

Since DNNs produce posteriors but the HMM requires the likelihood $p(x_t|s_t)$ during the decoding process (as shown in Eq.1.17), we need to convert the DNN outputs to likelihoods:

$$p(x_t|s_t) = \frac{p(s_t|x_t)p(x_t)}{p(s_t)} \tag{1.19}$$

$p(s_t)$ is the prior probability of state s_t estimated from the training set. $p(x_t)$ is independent of the word sequence and thus can be ignored. In this approach, HMMs are still used to model transition probability and perform sequence modeling, and therefore, DNN-HMM is usually called the hybrid model.

1.5.1.2 Language model

A language model, $p_{LM}(\mathbf{W})$ in Eq. 1.16, can be decomposed as:

$$p_{LM}(\mathbf{W}) = p(w_1, w_2, ..., w_m) = p(w_1)p(w_2|w_1) \cdots p(w_m|w_1, ..., w_{m-1}) \tag{1.20}$$

Each of these conditional probabilities could be estimated by checking counts of word sequences $(w_1, w_2, ..., w_i)$ and $(w_1, w_2, ..., w_{i-1})$ in the training corpus:

$$p(w_i|w_1, ..., w_{i-1}) = \frac{c(w_1, w_2, ..., w_i)}{c(w_1, w_2, ..., w_{i-1})} \tag{1.21}$$

As mentioned before, the classic technique to model LM for ASR are n-grams. An n-gram language model has the Markovian assumption, conditioning on the previous $n-1$ word, i.e.:

$$p(w_i|w_1, ..., w_{i-1}) = p(w_i|w_{i-(n-1)}, ..., w_{i-1}) \tag{1.22}$$

In real applications, 2- to 5-word history is typically used for an n-gram model, and, therefore, it will lose long-range context dependency. Recently, researchers have considered using RNNs for language modeling as well. However, due to the computational cost, LMs based on RNNs are typically used to re-score the N-best lists after the beam search is completed. LMs are usually trained independently from the AMs on a large amount of text data.

1.5.2 End-to-end speech recognition system

In the previous subsection, we described DNN-HMM hybrid ASR systems, which are composed of several individual components: acoustic models, pronunciation models, and language models. Each module is trained separately with different criteria, which may not be optimal for the overall task. Therefore, several end-to-end ASR models have been proposed in the last few years.

Connectionist Temporal Classification (CTC) model [GFG06] is one such end-to-end model, which can directly transform a variable acoustic sequence into English characters. The model can be optimized using a CTC loss. CTC models have been shown to learn pronunciations model directly. However, CTC models still suffer from conditional independence assumptions and must rely on explicit language models during decoding.

In the Chapter 4 of this dissertation, we focus on attention-based sequence-to-sequence models for end-to-end ASR, which are able to fold separate models of a conventional ASR system into a single neural network, and not be restricted by the independence assumptions of HMM and CTC models. Listen, Attend and Spell (LAS) [CJL16] is one of such models, and it offers improvements over other sequence-to-sequence models as shown in previous work [PRS17].

Let $\mathbf{x} = (x_1, ..., x_T)$ be the input sequence of audio frames, and $\mathbf{y} = (y_1, ..., y_S)$ be the output sequence of text units (such as characters, words or subwords). The LAS model predicts each output y_i using a conditional distribution over the previously emitted output $y_{<i}$ and the input acoustic sequence $\mathbf{x}$. The LAS model consists of three sub-modules as shown in Figure 1.3: an encoder which is analogous to a conventional acoustic model, an attender that does alignment, and a decoder that is analogous to the language model.

The encoder (the Listen function) transforms the original feature sequence $\mathbf{x}$ into a high level representation $\mathbf{h}$. The attender and decoder (the AttendandSpell function) take $\mathbf{h}$ as input and produce a distribution over text unit sequences:

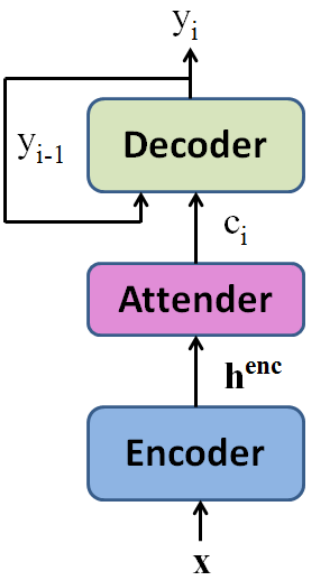


Figure 1.3: Components of the LAS model.

$$\mathbf{h} = \text{Listen}(\mathbf{x}) \tag{1.23}$$

$$p(\mathbf{y}|\mathbf{x}) = \text{AttendAndSpell}(\mathbf{h}) \tag{1.24}$$

The Listen function can be LSTM networks or stacked CNN and LSTM networks. The AttendAndSpell function is an attention-based transducer:

$$s_i = \text{DecoderRNN}(y_{i-1}, c_{i-1}, s_{i-1}) \tag{1.25}$$

$$c_i = \text{AttentionContext}(s_i, \mathbf{h}) \tag{1.26}$$

$$p(y_i|\mathbf{x}, \mathbf{y}_{<i}) = \text{TokenDistribution}(s_i, c_i) \tag{1.27}$$

The DecoderRNN function produces a transducer state s_i as a function of the previously emitted token y_{i-1}, the previous attention context c_{i-1} , and the previous transducer state s_{i-1}. Usually a unidirectional LSTM network is used for DecoderRNN.

The AttentionContext function generates context c_i with a fully-connected attention network. It first predicts the attention scores for current transducer state s_i with each of the hidden vector in $\mathbf{h}$. The attention scores act as alignments between output text unit and input frames. The attention context c_i is then created as a weighted linear sum of $\mathbf{h}$ using the attention scores. The TokenDistribution function is a fully-connect nework with softmax outputs modeling the text unit distribution $p(y_i|\mathbf{x}, \mathbf{y}_{<i})$.

1.5.3 Speech corpora

Three speech corpora are used for the ASR experiments in this dissertation. A brief overview of each speech corpus will be described in the following subsections.

1.5.3.1 Fisher English

Fisher English dataset [CMW04] is a conversational telephone speech dataset for large vocabulary continuous speech recognition (LVCSR). It contains time-aligned transcript data for 11699 complete conversations, each lasting up to 10 mins (in total around 2000 hours).

1.5.3.2 Wall Street Journal

The Wall Street Journal (WSJ) [PB92] speech corpus contains read speech of articles drawn from the Wall Street Journal text corpus. The corpus has rich diversity of voice quality and dialect. The speech data was recorded using microphones with 16kHz sampling rate. In this dissertation, 80 hours' speech data is used.

1.5.3.3 LibriSpeech

The LibriSpeech corpus [PCP15] is an ASR corpus based on public domain audio books. The corpus is derived from LibriVox's audio books, and contains around 1000 hours of read speech sampled at 16 kHz.

1.5.4 Evaluation metrics

In order to compare ASR hypothesis and ground-truth transcription, word error rate (WER) is introduced to evaluate recognition accuracy. WER measures the rate of word errors in ASR hypothesis, which is computed as:

$$WER = \frac{S + D + I}{N} \tag{1.28}$$

where S, D and I are the number of substitutions, deletions and insertions, respectively. N is the number of words in the reference (transcription). Computation of WER requires aligning each hypothesized word sequence with the reference using dynamic programming.

1.6 Dissertation Outline

The rest of this dissertation is organized as follows:

Chapter 2 presents two novel neural-network based representation learning approaches for short-utterance speaker verification. Results on large SV databases are presented and the effectiveness of speaker representation learning using the proposed approaches is analyzed.

Chapter 3 investigates joint feature extraction and acoustic modeling using unified neural-network models. Several novel neural-network architectures are presented, where complex DFT or raw waveforms constitute the input. Experimental results are presented for a large-scale keyword spotting task and a large-vocabulary ASR task.

Chapter 4 proposes two novel neural-network models for sequence modeling. An acoustic-sequence model is presented for an acoustic scene classification task, and then a sequence-to-sequence based text-error correction model is proposed for an end-to-end ASR system.

Chapter 5 summarizes the key concepts and results of this dissertation, and provides suggestions for future work.

CHAPTER 2

Learning speaker representations from short utterances

2.1 Introduction

As mentioned in Chapter 1, the i-vector based framework has defined the state-of-the-art for text-independent speaker recognition systems. The i-vector/ PLDA systems perform well if long (e.g. more than 30 s) enrollment and test utterances are available, but the performance degrades rapidly when only limited data are available [KVD11]. However, the requirement of significant amounts of speech for evaluation, has limited the potential of its widespread practical implementations. A speaker verification system, in the real world, is constrained by the amount of speech data.

To address this issue, in this chapter, we present two novel neural-network based representation learning approaches for short-utterance speaker verification. The fundamental idea is to alleviate possible phoneme mismatch in text-independent short utterance situatitons.

We first propose a method to use a deep neural network to estimate subglottal features from speech signals by leveraging the speaker specificity and stationarity of subglottal acoustics, which are largely phoneme independent. This work was published in [GYM16]. We then propose an i-vector mapping approach using deep neural networks, which maps the short utterance i-vector to its long version. Full-length (more than 30sec-length utterance) i-vectors have much smaller variations compared with i-vectors extracted from short utterances. The work was partially published in [GNA17, GXQ18].

2.2 Related work

There has been a number of methods to learn speaker representations from short utterance. Recently, several approaches have been proposed which use deep neural networks to learn speaker embeddings from short utterances. In [SGP17], the authors use a neural network, which is trained to discriminate between a large number of speakers, to generate fixed-dimensional speaker embedding, and the speaker embedding are used for PLDA scoring. In [ZK17], the authors propose an end-to-end system which directly learns a speaker discriminative embedding using a triplet loss function and an Inception Net [SLJ15]. Both methods show improvement over GMM-based i-vector systems. A few recent papers have also focused on i-vector mapping, which maps the short utterance i-vector to its long version. In [KMA18], the authors proposed a probabilistic approach, in which a GMM-based joint model between long and short utterance i-vectors was trained, and a minimum mean square error (MMSE) estimator was applied to transform a short i-vector to its long version. However, the proposed mapping function is actually a weighted sum of linear functions, which may not be complex enough to model this mapping.

2.3 Learning speaker-specific and phoneme-invariant subglottal acoustic features

2.3.1 Subglottal acoustic features

Our previous research indicates that subglottal acoustics (capturing the acoustics of the trachea-bronchial airways) are speaker specific and their spectral characteristics are much less variable than the spectral characteristics of speech waveforms [Guo15, GYA17, GPY15]. Subglottal acoustic data were recorded by a noninvasive accelerometer attached to the skin of the neck below the thyroid cartilage. The recordings constitute the WashU-UCLA database [ALS15]. The database consists of 35 monosyllables (14 hVd and 21 CVd words, where V includes all the American English monophthongs and diphthongs) in a phonetically neutral carrier phrase (I said a __ again), with 10 repetitions of each word by each speaker. The corpus

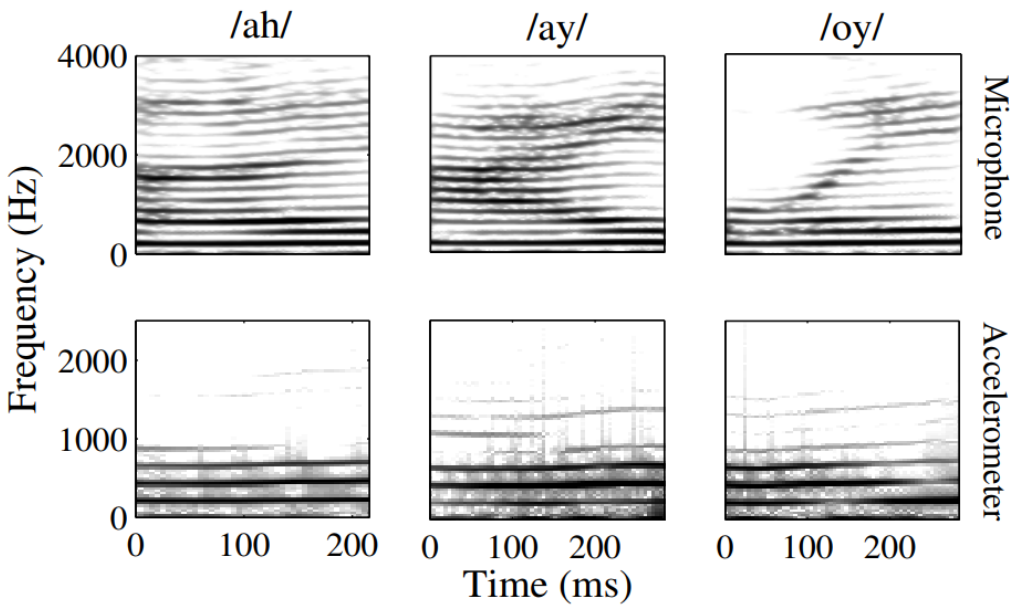


Figure 2.1: Spectrograms of three vowels by a female speaker to compare within-speaker variability of microphone speech (top panel) and subglottal acoustics (bottom panel). Note that the subglottal acoustics don't vary much. Data are sampled from the recordings of a female speaker in the WashU-UCLA corpus.

has simultaneous microphone and (subglottal) accelerometer recordings of 25 adult male and 25 adult female speakers of American English, and in total 17500 individual microphone (and accelerometer) waveforms. Figure 2.1 exemplifies the stationarity of subglottal acoustics using spectrograms of vowels and their corresponding recordings of subglottal acoustics. The stationary nature of subglottal acoustics can be particularly beneficial when the amount of speech data is limited.

2.3.2 Proposed estimation method

Estimating subglottal features using speech signals is challenging. DNNs have been shown to be effective for feature mapping of speech signals [HHB15]. We adopt DNNs here for subglottal feature estimation and evaluate the technique on the WashU-UCLA corpus (which contains time-synchronized recordings of speech and subglottal acoustics). We train a DNN regression model to learn the spectral feature mapping from speech to subglottal acoustics. The objective function for optimization is based on the mean square error. Eq. 2.1 is the

cost function for each training batch:

$$L_r(x_k, y_k; \theta_r) = \frac{1}{N} \sum_{k=1}^{N} \|y_k - f(x_k)\|^2 \tag{2.1}$$

where x_k and y_k are the input speech feature and the corresponding subglottal acoustic feature, respectively, and θ_r denotes the regression parameters to be learned during training. The trained DNN regression model $f(.)$ provides a non-linear mapping from a more variable speech spectral domain to the less variable subglottal spectral domain (viewed in some sense as a many-to-one mapping).

2.3.3 Estimation experiments

2.3.3.1 Feature extraction and DNN mapping setup

To avoid redundancy and keep the phonetic balance in the data that is used to train the DNN regression model, only the vowel segments of the monosyllables in the database are isolated and used. Another reason why we only extract the vowel segments is that the accelerometer signals show little information for consonants. Since we only have the DNN mapping for vowels, we need a way to deal with non-vowel segments while estimating subglottal acoustic features for the speaker verification experiment. Section 2.3.4 explains the specific method used for that.

We extract the 40 dimensional log Mel-filterbank coefficients for both speech and accelerometer segments, and use the filterbank feature vectors of the speech segments as input and their corresponding subglottal filterbank feature vectors as output for the DNN model. The input and output features are normalized using the L2 norm of the feature vector. The activation functions of both the hidden layers and the output layer are the tanh functions. Three hidden layers are used and each hidden layer has 256 neurons. We use backpropagation with mini-batch stochastic gradient descent to train the DNN model, and the optimization technique uses adaptive gradient descent along with a momentum term. The THEANO DNN toolkit is used for DNN training [BLP12]. All available vowel segment pairs (17500 in

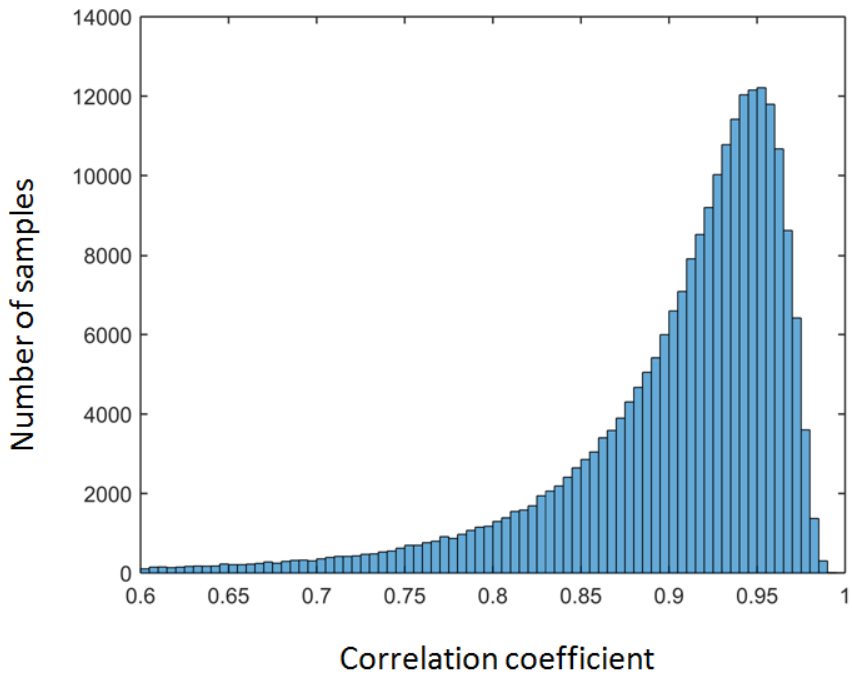


Figure 2.2: Histogram of the correlation coefficient of the actual and estimated subglottal Mel-filterbank coefficients for each frame in the validation dataset.

total) are split into a training set and a validation set. The training set has roughly 80% of the data and the rest is for validation. All signals are down sampled to 8 kHz (from the original sampling rate of 48 kHz), which is consistent with the NIST SRE dataset (used for speaker verification). The log Mel-filterbank coefficients for both speech and subglottal acoustic signals are extracted at 10ms intervals using a 20 ms Hamming window.

2.3.3.2 Evaluation results

To evaluate the performance of the DNN-based estimation model, we use two methods: (1) computing the correlation between actual and estimated log Mel-filterbank coefficients for each frame of subglottal recordings, and (2) comparing the actual and estimated subglottal features with regards to their ability to discriminate between speakers.

Figure 2.2 shows the histogram of the correlation coefficients for all frames in the validation dataset. The average value of the correlation coefficients is 0.9, which indicates the sufficiency of the DNN-based estimation model.

Feature sets	J-Ratio
MFCCs (x_1-x_{20})	4.92
Actual SGCCs (x_1-x_{20})	5.48
Estimated SGCCs (x_1-x_{20})	5.47

Table 2.1: J-ratio, a measure of class separation for different feature values. Features were extracted from isolated vowel recordings of speech and subglottal acoustics, for all the 50 male and female adult speakers in the WashU-UCLA corpus.

To compare the actual and estimated subglottal filterbank features in terms of speaker discriminability, the J-Ratio [Fuk13], which measures class separation, is used. Before calculating the J-Ratio, we compute the DCT of the log Mel-filterbank coefficients, since it will decorrelate the filterbank features and be consistent with MFCC features (used commonly for speaker verification tasks). We refer to the subglottal features after taking the DCT on the log Mel-filterbank coefficients as subglottal cepstral coefficients, which are denoted as SGCCs. The zeroth cepstral coefficient is discarded, since it only captures the energy information. The first 20 coefficients are used for both MFCCs and SGCCs. Given feature vectors for N speakers, the J-Ratio can be computed using Eqs. 2.2-2.4:

$$S_w = \frac{1}{M}\Sigma_{s=1}^{M} R_i \tag{2.2}$$

$$S_b = \frac{1}{M}\Sigma_{s=1}^{M} (x_i - x_o)(x_i - x_o)^T \tag{2.3}$$

$$J = \mathrm{Tr}((S_b + S_w)^{-1} S_b) \tag{2.4}$$

where S_w is the within-class scatter matrix, S_b is the between-class scatter matrix, x_i is the mean feature vector for the i_{th} speaker, x_o is the mean of all x_is, and R_i is the covariance matrix for the i_{th} speaker (a higher J-Ratio means better separation).

Table 2.1 shows the J-Ratio values for different feature sets. The results show that: (1) SGCCs offer better separation compared to MFCCs, which is partly attributable to the

stationarity of subglottal acoustics and the low within-class variance that they represent, and (2) the estimated SGCCs are similar in performance to actual SGCCs, which is due to the effectiveness of the DNN-based feature mapping model.

2.3.4 Speaker verification experiments

2.3.4.1 Task description and experimental settings

We evaluate our features and proposed system on the NIST SRE datasets with state-of-the-art i-vector/PLDA framework. The NIST SRE 2004, 2005, 2006 and Switchboard II datasets are used as the development dataset. Gender-dependent universal background models (UBM) with 2048 Gaussians are trained using a subset of the development dataset, which only has utterances from male speakers. The total variability subspace for the i-vector extractor, channel compensation technique LDA and speaker variability subspace for PLDA are trained using all male speakers from the development dataset. The Kaldi toolkit [PGB11] is used to build the system.

MFCCs using the first 20 coefficients (discarding the zeroth coefficient) with appended first and second order derivatives are extracted from the detected speech segments after voice activity detection. A 20 ms Hamming window, a 10 ms frame shift, and a 23-channel Mel-filterbank are used for baseline MFCC feature extraction. A total variability matrix T of 400 factors is used and the dimension is reduced to 200 using LDA before the PLDA modeling. Length normalization of the ivectors is also used.

For SGCC feature extraction, non-vowel speech frames must be discarded since the DNN feature extractor is trained only on isolated vowels. A normalized autocorrelation peak value of 0.7 is used as a threshold to detect the strongly-voiced vowel frames. A 20 ms Hamming window and a 10 ms frame shift are used to extract 40-channel Mel-filterbank coefficients from voiced frames. Then, the filterbank coefficients are inputted into the trained DNN feature extractor to estimate subglottal features. The first 20 coefficients (excluding the zeroth coefficient) with appended first order derivatives are selected after taking the DCT on the estimated subglottal Mel-filterbank coefficients. A total variability subspace of 150

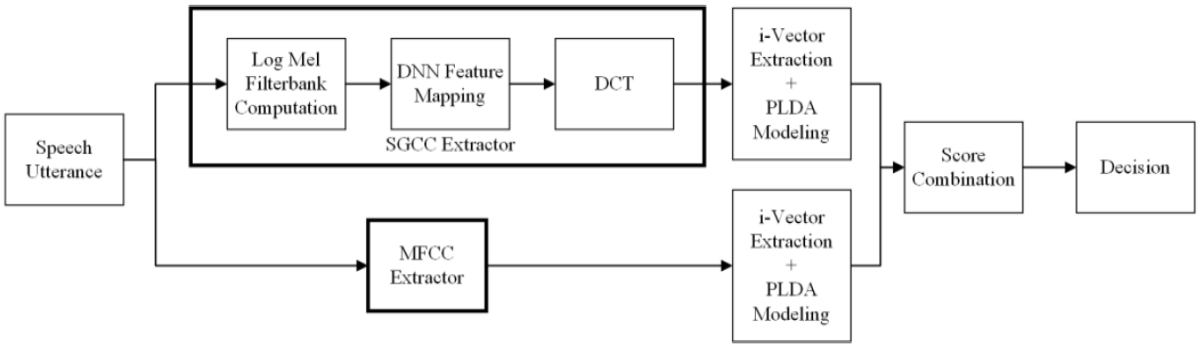


Figure 2.3: Block diagram of the proposed framework.

dimensions is used and the same number of latent components is adopted for PLDA modeling. Length normalization is also done to scale the lengths of each i-vectors to unit length.

The NIST SRE 2008 core task, which has both microphone and telephone speech and channel matched and mismatched conditions, was used for the experiments. The enrollment and testing datasets are truncated to 10 seconds and 5 seconds for each utterance for the short-utterance speaker verification tasks. The core task contains 1993 female and 1270 male speakers. Only the male speakers with 39433 test trials are used here for evaluation. We show the results for conditions C2 (interview speech from the same microphone types for both training and testing), C7 (English telephone speech spoken by both native and non-native U.S. English speakers), and C8 (English telephone speech spoken by native U.S. English speakers). These conditions were chosen because they contain English-only speech.

Given an utterance, MFCCs and SGCCs are computed as described earlier. Each feature set will generate a set of scores for test trials. Scores from the two speaker verification systems were normalized to the range (0, 1) and fused in a linearly-weighted fashion such that the weights sum up to 1. The fused scores are used to make final decisions. The overall block diagram of the framework is presented in Figure 2.3.

2.3.4.2 Results and analysis

While the J-Ratio analysis in Section 2.3.3.2 shows that the estimated SGCCs can provide better speaker separation than MFCCs using the selected vowel segments, initial experiments

Conditions	Feature set	C2	C7	C8
10sec-10sec	MFCCs	8.12	19.51	21.08
	MFCCs+SGCCs	7.20	18.24	19.29
	Relative improvement	11.5%	6.5%	8.5%
5sec-5sec	MFCCs	14.11	27.76	27.83
	MFCCs+SGCCs	12.10	26.21	26.07
	Relative improvement	14.3%	5.6%	6.3%

Table 2.2: EERs for the MFCC baseline system and the fused system on the NIST SRE 08 truncated 10sec-10sec and 5sec-5sec evaluation tasks. The relative improvements in EERs are also shown.

indicate that the SGCC-only system performs worse than the MFCC baseline on the NIST SRE dataset. This discrepancy could be due to (1) acoustic mismatch between the WashU-UCLA corpus and the speaker verification corpora, and (2) using only the strongly-voiced frames for SGCC estimation.

Therefore, we further investigate the performance of fused MFCCs+SGCCs systems in Table 2.2, to examine if they are complementary to each other. The fused system gives improvement for all conditions of the short-utterance task. The gains are higher and more significant for the conditions that better match the characteristics of the WashU-UCLA corpus used for DNN training. For example, the combined system yields the biggest improvement under matched microphone speech (C2), with a relative 11.5% EER reduction for the 10sec-10sec task, and 14.3% for the 5sec-5sec task. This may be due to the fact that the DNN mapping model is also trained under matched microphone speech. For English telephone speech, we can see that C8, which contains utterances spoken by Native American English speakers, gives relative better improvement compared with C7. This may also result from the fact that all the speakers in the WashU-UCLA dataset are native US speakers. The weights used for fusion are the same for both 10sec-10sec and 5sec-5sec tasks, which are 0.85 for MFCCs and 0.15 for SGCCs.

2.4 Learning non-linear mapping from short-utterance to long-utterance i-vectors

The previous section investigates the method of learning speaker-specific and phoneme-invariant features at the frame level. In this section, we will explore an approach of learning phoneme-invariant speaker features through utterance-level representations.

2.4.1 The effect of utterance durations on i-vectors

Full-length i-vectors have relatively smaller variations compared with i-vectors extracted from short utterances, because i-vectors of short utterances can vary considerably with changes in phonetic content. In order to show the variation changes between long and short utterance i-vectors, we first calculate the average diagonal covariance (denoted as σ_m) of i-vectors across all utterances of a given speaker m and then calculate the mean (denoted as σ_{mean}) of the covariances over all speakers. σ_m and σ_{mean} are defined in as:

$$\sigma_m = \frac{1}{N}\Sigma_{n=1}^{N} \mathrm{Tr}((w_{mn} - \bar{w_m})(w_{mn} - \bar{w_m})^T) \tag{2.5}$$

$$\sigma_{mean} = \frac{1}{M}\Sigma_{m=1}^{M}\sigma_m \tag{2.6}$$

where $\bar{w_m}$ corresponds to the mean of the i-vectors belonging to speaker m. N represents the total number of utterances for speaker m, Tr(.) represents the trace operation, and M is total number of speakers.

In order to compare the σ_{mean} for long and short utterance i-vectors, we choose around 4000 speakers with multiple long utterances (more than 2 min in duration and 100 s of active speech) from the SRE and Switchboard-2 (SWB-2) datasets (in total around 40000 long utterances) and truncate each long utterances into multiple 5-10 s short utterances. We plot the distribution of active-speech length (utterance length after voice activity detection) across these 40000 long utterances in Figure 2.4. From the figure, we can observe that the

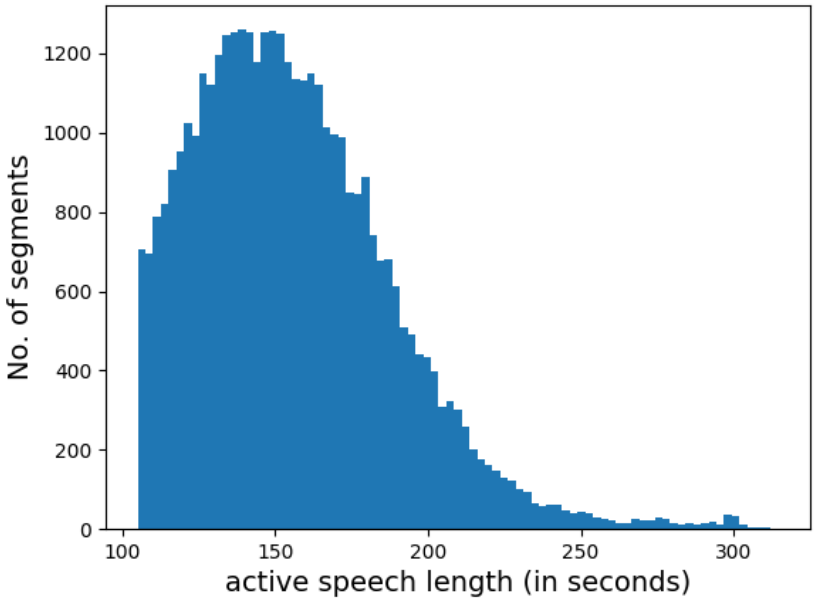


Figure 2.4: Distribution of active speech length of 40000 long utterances in SRE and SWB datasets.

Table 2.3: Mean variance of long and short utterances (from the SRE and Switchboard datasets)

	i-vectors	
	long utterance	short utterance
mean variance(σ_{mean})	283	493

majority of the utterances have active-speech length between 100-250 s. The i-vectors are then extracted for each short and long utterance using the I-vector_DNN system. Table 2.3 shows the mean variance σ_{mean} across all speakers calculated from long and short utterance i-vectors. The mean of the variances in the Table 2.3 indicates that short-utterance i-vectors, as expected, have larger variation compared to those of long-utterance i-vectors.

2.4.2 DNN-based i-vector mapping

In order to alleviate possible phoneme mismatch in text-independent short utterances, we propose several methods to map short-utterance i-vectors to their long versions. This mapping is a many-to-one mapping, from which we want to restore the missing information from

the short-utterance i-vectors and reduce their variance.

In this section, we will introduce and compare several novel DNN-based i-vector mapping methods. Our pilot experiments indicate that, if we train a supervised DNN to learn this mapping directly, similar to the approaches in [BR17] , the improvement is not significant due to over-fitting to the training dataset. In order to solve this problem, we propose two different methods which model the joint representation of short and long utterance i-vectors by using an autoencoder. The decoder reconstructs the original input representation and forces the encoded embedding to learn a hidden space which represents both short and long utterance i-vectors and thus can lead to better generalizations. The first is a two-stage method: use an autoencoder to first train a bottleneck representation of both long and short utterance i-vectors, and then use pre-trained weights to perform a supervised fine-tuning of the model, which maps the short-utterance i-vector to its long version directly. The second is a single-stage method: jointly train the supervised regression model with an autoencoder to reconstruct the short i-vectors. The final loss to optimize is a weighted sum of the supervised regression loss and the reconstruction loss. In the following subsections, we will introduce these two methods in detail.

2.4.2.1 DNN_1 (two-stage method): pre-training and fine-tunning

In order to find a good initialization of the supervised DNN model, we first train a joint representation of both short and long utterance i-vectors using an autoencoder. The autoencoder consists of an encoder and a decoder as illustrated in Figure 2.5. We first concatenate the short i-vector w_s and its long version w_l into z as input. The encoder function $h = f(z)$ learns a hidden representation of input vector z, and the decoder function $\hat{z} = g(h)$ produces a reconstruction. The learning process is described as minimizing the loss function $L(z, g(f(z)))$. The autoencoder learns the joint hidden representation of both short and long i-vectors, which leads to good initialization of the second-stage supervised fine-tuning. In order to learn a more useful representation, we add a restriction on the autoencoder: constrain the hidden representation h to have a relatively small dimension in order to learn the

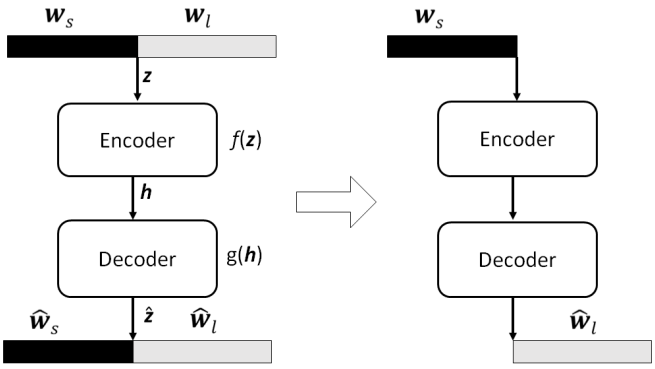


Figure 2.5: DNN_1: two-stage training of i-vector mapping. Left schema corresponds to the first-stage pre-training. A short-utterance i-vector w_s and a corresponding long-utterance i-vector w_l are first concatenated into z. Then z is fed into an encoder $f(.)$ to generate the joint embedding h. h is passed to the decoder $g(.)$ to generate the reconstructed $\hat{z}$, which is expected to be a concatenation of a reconstructed $\hat{w}_s$ and $\hat{w}_l$. Right schema corresponds to the second-stage fine-tuning. The pre-trained weights in the first stage is used to initialize the supervised regression model from w_s to w_l. After training, the estimated i-vector $\hat{w}_l$ is used for evaluation.

most salient features of the training data.

For the encoder function $f(.)$, we adopt options from several fully-connected layers to stacked residual blocks [HZR16], in order to investigate the effect of encoder depth. Each residual block has two fully-connected layers with a short-cut connection as shown in Figure 2.6. By using residual blocks, we are able to train a very deep neural network without adding extra parameters. A deep encoder may help learn better hidden representations. For a decoder function $g(.)$, we use a single fully-connected layer with a linear regression layer, since it is enough to approximate the mapping from the learned hidden representation h to the output vector. For the loss function, we use the mean square error criterion, which is $\|g(f(z)) - z\|^2$.

Once the autoencoder is trained, we use the trained DNN-structure and weights to initialize the supervised mapping. We optimize the loss between the predicted long i-vector and the real long i-vector $\frac{1}{N}\sum_{n=1}^{N}\|\hat{w}_l - w_l\|^2$ as shown in Figure 2.5. We denote this method as DNN_1.

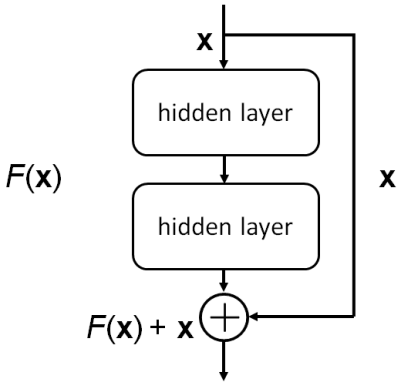


Figure 2.6: Residual block. An input x is first passed into two hidden layers to get $F(x)$ and it also goes through a short-cut connection, which skips the hidden layers. The final output of the residual block is a summation of $F(x)$ and x.

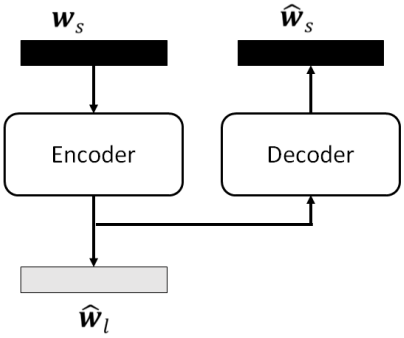


Figure 2.7: DNN_2: single-stage training of i-vector mapping. A short-utterance i-vector w_s is passed to an encoder and the output of the encoder is first used to generate the estimated long-utterance i-vector $\hat{w}_l$ and it is also fed into a decoder to generate the reconstructed short-utterance i-vector $\hat{w}_s$. The two tasks are optimized jointly.

2.4.2.2 DNN_2 (single-stage method): semi-supervised training

The two-stage method mentioned in the previous section, needs to first train a joint representation using the autoencoder and then perform a fine-tuning to train the supervised mapping. In this section, we introduce another unified semi-supervised framework based on our previous work [GNA17] which can jointly train the supervised mapping with an autoencoder to minimize the reconstruction error. The joint framework is motivated by the fact that by sharing the hidden representations among supervised and unsupervised tasks, the network generalizes better and it can also avoid using the two-stage training procedures and speed up training. This method is denoted as DNN_2.

We adopt the same autoencoder framework as mentioned in the previous section, which has an encoder and a decoder, but the input to the encoder here is the short-utterance i-vector w_s. The output from the encoder will be connected to a linear regression layer to predict the long-utterance i-vector w_l, and it will also be used to reconstruct the short-utterance i-vector w_s itself by inputting it into a decoder, which gives rise to the autoencoder structure. The entire framework is shown in Figure 2.7.

We define a new objective function to jointly train the network. Let us use $\hat{w}_l$ and $\hat{w}_s$ to represent the output from the supervised regression model and autoencoder, respectively. We can define the objective loss function L_{total} which combines the loss from the regression model and the autoencoder in a weighted fashion as:

$$L_{total} = (1 - \alpha)L_r + \alpha L_a \tag{2.7}$$

where L_r is the loss of regression model defined as

$$L_r(w_s, w_l; \theta_r) = \frac{1}{N} \sum_{n=1}^{N} \|\hat{w}_l - w_l\|^2 \tag{2.8}$$

and L_a is the loss of an autoencoder defined as:

$$L_a(w_s, w_s; \theta_a) = \frac{1}{N} \sum_{n=1}^{N} \|\hat{w}_s - w_s\|^2. \tag{2.9}$$

Moreover, θ_r and θ_a are parameters of the regression model and autoencoder respectively, which are jointly trained and share the weights of the encoder layer. α is a scalar weight, which determines how much the reconstruction error is used to regularize the supervised learning. The reconstruction loss of the autoencoder L_a forces the hidden vector generated from the encoder to reconstruct the short-utterance i-vector w_s in addition to predicting the target long-utterance i-vector w_l, and helps prevent the hidden vector from over-fitting w_l. For testing, we only use the output from the regression model $\hat{w}_l$ as the mapped i-vector.

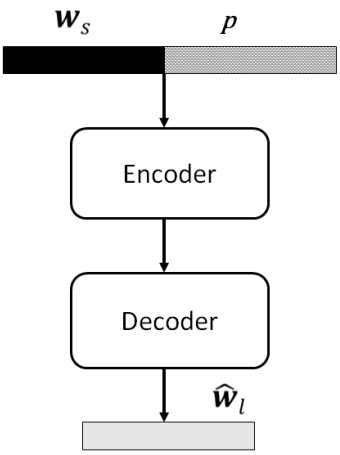


Figure 2.8: I-vector mapping with additional phoneme information. A short-utterance i-vector w_s is concatenated with a phoneme vector p to generate the estimated long-utterance i-vectors $\hat{w}_l$.

2.4.2.3 Adding phoneme information

The large variance of short utterances is mainly due to phonetic differences. In order to aid the neural network to train this non-linear mapping, for a given utterance, we extract the senone (clusters of tri-phones) posteriors for each frame and calculate the mean posterior across frames as a phoneme vector, which is then appended to a short utterance i-vector as input (Figure 2.8). The training procedure still follows the proposed joint modeling methods (DNN_1 or DNN_2). The phoneme vectors are expected to help normalize the short-utterance i-vector, and provide extra information for this mapping. The phoneme vector p is defined as:

$$p = \frac{1}{N}\sum_{t=1}^{N} P(c|y_t, \Theta) \tag{2.10}$$

The posterior $P(c|y_t, \Theta)$ is generated from the TDNN-based senone classifier.

Table 2.4: Datasets used for developing I-vector_GMM and I-vector_DNN systems

	I-vector_GMM	I-vector_DNN
UBM (3472)	Switchboard, NIST 04, 05, 06, 08	Fisher English
T (600)	Switchboard, NIST 04, 05, 06, 08	Switchboard, NIST 04, 05, 06, 08
PLDA	NIST 04, 05, 06, 08	NIST 04, 05, 06, 08

2.4.3 Experimental set-up

2.4.3.1 I-vector baseline systems

We evaluate our techniques using the state-of-the-art GMM- and DNN-based i-vector/G-PLDA systems using the Kaldi toolkit [PGB11]. The general frameworks of these systems were described in Chapter 1.

Configurations of I-vector_GMM system

For the I-vector_GMM system, the first 20 MFCC coefficients (discarding the zeroth coefficient) and their first and second order derivatives are extracted from the detected speech segments after an energy-based voice activity detection (VAD). A 20 ms Hamming window, a 10 ms frame shift, and a 23 channels filterbank are used. Universal background models with 3472 Gaussian components are trained, in order to have a fair comparison with the I-vector_DNN system, whose DNN has 3472 outputs. Initial training consists of four iterations of expectation-maximization (EM) algorithm using a diagonal covariance matrix and then additional four iterations with a full-covariance matrix. The total variability subspace with low rank (600) is trained for five iterations of EM. The backend training consists of i-vector mean subtraction and length normalization, followed by PLDA scoring.

The UBM and i-vector extractor training data consist of male and female utterances from the SWB and NIST SRE datasets. The SWB data contains 1000 speakers and 8905 utterances of SWB-2 Phases II. The SRE dataset consists of 3805 speakers and 36614 utterances from SRE 04, 05, 06, 08. The PLDA backends are trained only on the SRE data. The dataset information is summarized in Table 2.4.

Configurations of I-vector_DNN system

For the I-vector_DNN system, a TDNN is trained using about 1,800 hours of the English portion of Fisher [CMW04]. In the TDNN acoustic modeling system, a narrow temporal context is provided to the first layer and the context width increases for subsequent hidden layers, which enables the higher levels of the network to learn greater temporal relationships. The features are 40 mel-filterbank features with a frame-length of 25 ms. Cepstral mean subtraction is performed over a window of 6 s. The TDNN has six layers, and a splicing configuration similar to those described in [PPK15]. In total, the DNN has a left-context of 13 and a right-context of 9. The hidden layers use the p-norm (where p = 2) activation function [ZTP14], an input dimension of 350, and an output dimension of 3500. The softmax output layer computes posteriors for 3472 triphone states, which is the same as the number of components for I-vector_GMM system. No fMLLR or i-vectors are used for speaker adaptation.

The trained TDNN is used to create a UBM which directly models phonetic content. A supervised-GMM with full-covariance is created first to initialize the i-vector extractor based on TDNN posteriors and speaker recognition features. Training the T matrix also requires TDNN posteriors and speaker recognition features. During i-vector extraction, the only difference between this and the standard GMM-based systems is the model used to compute posteriors. In the I-vector_GMM system, speaker recognition features are selected using a frame-level VAD. However, in the I-vector_DNN system, in order to maintain the correct temporal context for the concatenated TDNN input features, we cannot remove frames using VAD in the beginning. Instead, the VAD results are used to filter out posteriors corresponding to non-speech frames.

2.4.3.2 Evaluation databases

We first evaluate our systems on condition 5 (extended core task) of SRE10 [MG10]. The test consists of conversational telephone speech in enrollment and test utterances. There are 416119 trials, over 98% of which are nontarget comparisons. Among all trials, 236781 trials

are for female speakers and 179338 trials are for male speakers. For short-utterance speaker verification tasks, we extracted short utterances which contain 10 s and 5 s speech (after VAD) from condition 5 (extended task). We train the PLDA and evaluate the trials in a gender-dependent way.

Moreover, in order to validate our proposed methods in real conditions and demonstrate the models' generalization, we use SITW, a relatively recent speech database [MFC16]. The SITW speech data were collected from open-source media channels with considerable mismatch in terms of audio conditions. We designed an arbitrary-length short-utterance task using the SITW dataset to represent real-life conditions. We show the evaluation results using the best-performing models validated on the SRE10 dataset.

2.4.3.3 I-vector mapping training

In order to train the i-vector mapping model, we selected 39754 long utterances, each having more than 100 s of speech after VAD, from the development dataset. For each long utterance, we used a 5 s or 10 s window to truncate the utterance, and the shift step is half of window size (2.5 s or 5 s). We applied the aforementioned procedures to all long utterances, and in the end we got 1.2M 10 s utterances and 2.4M 5 s utterances. All short-utterance i-vector together with its corresponding long-utterance i-vector are used as training pairs for DNN-based mapping models. We train the mapping models for each gender separately and evaluate the model in a gender-dependent way.

For the proposed two DNN-based mapping models, we use the same encoder and decoder configurations. For the encoder, we first use two fully-connected layers. The first layer has 1200 hidden nodes and the second layer has 600 hidden nodes which is a bottleneck layer (1.44M parameters in total). In order to investigate the depth of the encoder, we design a deep structure with two residual blocks and a bottleneck layer, in a total of 5 layers. Each residual block (as defined in Section 2.4.2.1) has two fully connected layers with 1200 hidden nodes and the bottleneck layer has 600 hidden nodes (5.76M parameters in total). For the decoder, we always use one fully-connected layer (1200 hidden nodes) with a linear output

layer (1.44M parameters in total).

In order to add phoneme information for i-vector mapping, phoneme vectors are generated for each utterance by taking the average of the senone posteriors across frames. Since the phoneme vectors have a different value range compared with i-vectors, this difference will de-emphasize their effect for training the mapping. Therefore we scale up the phoneme vector values by a factor of 500, in order to match the range of i-vector values. The up-scaled phoneme vector is then concatenated with short-utterance i-vector for i-vector mapping.

All neural networks are trained using the Adam optimization strategy [KB14] with the mean square error criterion and exponentially decaying learning rate starting from 0.001. The networks are initialized with the Xavier initializer [GB10], which is better than the Gaussian initializer as shown in [GNA17]. The relu activation function is used for all layers. For each layer, before passing the tensors to the nonlinearity function, a batch normalization layer [IS15] is applied to normalize the tensors and speed up the convergence. For the combined loss of DNN_2, we set equal weights (α=0.5) for both regression and autoencoder loss for initial experiments. The shuffling mechanism is applied on each epoch. The Tensorflow toolkit [AAB16] is used for neural network training.

2.4.4 Evaluation of proposed i-vector mapping methods

In order to investigate the effect of the proposed i-vector mapping algorithms, we first calculate the average square Euclidean distance between short and long utterance i-vector pairs on the SRE10 evaluation dataset before and after mapping. The average mean square Euclidean distance D_{sl} between short and long utterance i-vector is defined as follow:

$$D_{sl} = \frac{1}{N}\Sigma_{s=1}^{N}(\Sigma_{i=1}^{L}(w_s(i) - w_l(i))^2) \tag{2.11}$$

where w_s and w_l represent the short-utterance and long-utterance i-vector respectively, L is the length of i-vectors and N is number of short and long i-vector pairs.

We compare the D_{sl} values for 10 s and 5 s short-utterance i-vectors and also the mapped

Table 2.5: Square Euclidean distance (D_{sl}) between short and long utterance i-vector pairs from SRE10 before and after mapping.

	D_{sl}			
	10 s		5 s	
	original	mapped	original	mapped
female	558.3	306.8	618.8	352.1
male	493.2	308.8	556.1	346.5

Table 2.6: J-ratio for short-utterance i-vectors from SRE10 before and after mapping.

	J-ratio			
	10 s		5 s	
	original	mapped	original	mapped
female	87.96	92.97	82.73	85.18
male	85.23	90.25	80.41	84.39

10 s and 5 s short-utterance i-vectors for female and male speakers in Table 2.5. From the table, we observe that, after mapping, the mapped short-utterance i-vectors have considerably smaller D_{sl} compared to the ones before mapping. After mapping, the D_{sl} in the 10 s condition is smaller compared with the 5 s condition.

Moreover, we calculate and compare the J-ratio [Fuk13] of the short-utterance i-vectors from SRE10 before and after mapping in Table 2.6. This ratio measures speaker separation ability. Given i-vectors for M speakers, the J-ratio can be computed using Eqs. 2.12-2.14:

$$S_w = \frac{1}{M}\Sigma_{s=1}^{M} R_i \tag{2.12}$$

$$S_b = \frac{1}{M}\Sigma_{s=1}^{M} (w_i - w_o)(w_i - w_o)^T \tag{2.13}$$

$$J = \mathrm{Tr}((S_b + S_w)^{-1} S_b) \tag{2.14}$$

where S_w is the within-class scatter matrix, S_b is the between-class scatter matrix, w_i is the mean i-vector for the i_{th} speaker, w_o is the mean of all w_is, and R_i is the covariance matrix for the i_{th} speaker (a higher J-Ratio means better separation).

From Table 2.6, we can observe that the mapped i-vectors have considerably higher J-ratios compared with original short-utterance i-vectors for both 5 s and 10 s conditions.

These results indicate that the proposed DNN-based mapping methods can generalize well to unseen speakers and utterances, and improve the speaker separation ability of i-vectors. In the next section, we will explore how these mapping algorithms can help to improve speaker verification systems.

2.4.5 Speaker verification experiments

2.4.5.1 I-vector baseline systems

In this section, we present and compare two baseline systems: an I-vector_GMM system and an I-vector_DNN system, with standard NIST SRE 10 full-length condition and truncated 10 s-10 s and 5 s-5 s conditions.

Table 2.7 shows the equal error rate (EER) and minimum detection cost function (minDCF) of the two baseline systems under full-length evaluation condition and truncated short-length evaluation conditions. Both DCF08 and DCF10 (defined in NIST 2008 and 2010 evaluation plans) are shown in the table. From the table, we can observe that the I-vector_DNN system gives significant improvement under the full-length condition compared with I-vector_GMM system and achieved a max of 52.94% relative improvement for the male condition, which is consistent with previously reported results [SGP15]. This is mainly because the DNN model provides phonetically-aware class alignments, which can better model speakers. The good performance is also due to the strong TDNN-based senone classifier, which makes alignments more accurate and robust. When both systems were evaluated on the truncated 10 s-10 s, 5 s-5 s evaluation conditions, the performances degrade significantly compared with the full-length condition. The main reason is, as mentioned earlier, significant phonetic mismatch between utterances. However, the performance of the I-vector_DNN system still outperforms the I-vector_GMM system by 8%-24%, even though the improvement is not as big as the full-length condition. From the table, we can also observe that the improvement is more significant for male speakers across all conditions. It may be the fact that phoneme clas-

Table 2.7: Baseline results for I-vector_GMM and I-vector_DNN systems under full-length and short-length utterances conditions reported in terms of EER, Relative Improvement (Rel Imp), and minDCF on NIST SRE10 database.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
Full-length condition				
I-vector_GMM	2.2	0.011/0.043	1.7	0.008/0.036
I-vector_DNN	1.4 (36.36%)	0.005/0.022	0.8 (52.94%)	0.003/0.017
10 s-10 s condition				
I-vector_GMM	13.8	0.063/0.097	13.3	0.057/0.099
I-vector_DNN	12.2 (11.59%)	0.054/0.093	10.2 (23.31%)	0.048/0.095
5 s-5 s condition				
I-vector_GMM	21.7	0.083/0.099	20.4	0.080/0.100
I-vector_DNN	19.9 (8.29%)	0.078/0.099	17.0 (16.67%)	0.072/0.100

sification is more accurate for male speakers, which could lead to a better phoneme-aware speaker modeling.

2.4.5.2 I-vector mapping results

In this section, we show and discuss the performance of the proposed algorithms when only short utterances are available for evaluation. Since, from Table 2.7, we observed better performance using I-vector_DNN systems, we will mainly use the I-vector_DNN system to investigate the mapping methods. We first show the results on the 10 s-10 s condition.

Previous work [KMA18, GNA17] highlights the importance of duration matching in PLDA model training. For instance when the PLDA is trained using long utterances and evaluated on short utterances, there is degradation in speaker verification performance compared to PLDA trained using matched-length short utterances. Therefore, we not only show our baseline results for the PLDA trained using the regular SRE development utterances, but also show the results for the PLDA condition using truncated matched-length short utterances.

For other baseline comparison, we first apply dimensionality reduction on i-vectors using linear discriminant analysis (LDA) and reduce the dimension of i-vectors from 600 to 150.

Table 2.8: Results for baseline (I-vector_DNN), matched-length PLDA training, LDA dimension reduction, DNN direct mapping and proposed DNN mapping in the 10 s-10 s condition of NIST SRE10 database.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
baseline	12.2	0.054/0.093	10.2	0.048/0.095
matched length PLDA	11.3 (7.38%)	0.052/0.093	9.4 (7.84%)	0.043/0.095
LDA 150	11.6 (5.00%)	0.052/0.093	9.8 (3.92%)	0.047/0.093
DNN direct mapping	10.5 (13.93%)	0.054/0.096	9.7 (4.90%)	0.047/0.093
DNN1 mapping	9.5 (22.13%)	0.047/0.091	7.7 (24.51%)	0.039/0.090
DNN2 mapping	9.5 (22.13%)	0.047/0.091	7.7 (24.51%)	0.039/0.089

This value has been selected according to the results of previous research [CL16]. LDA can maximize inter-speaker variability and minimize intra-speaker variability. We train the LDA transformation matrix using the SRE development dataset, and then, perform the dimension reduction for all development utterances and train a new PLDA model. For evaluation, all i-vectors are subjected to dimensionality reduction first and then we use the new PLDA model to get similarity scores. To compare with another short-utterance compensation technique, we evaluate the i-vector mapping methods proposed in [BR17], which use DNNs to train a direct mapping from short-utterance i-vectors to the corresponding long version. Similar to [BR17], we also add some long-utterance i-vectors as input for regularization purposes.

For our proposed DNN mapping methods, we first show the mapping results for both DNN_1 and DNN_2 with three hidden layers. Note that for mapped i-vectors, we use the same PLDA as the baseline system to get similarity scores. We further investigate the effect of pretraining iterations for DNN_1, the weight α of the reconstruction loss for DNN_2 and the depth of encoder, compare the results for different durations, and investigate the effect of additional phoneme information. We also compare with mapping results for both I-vector_GMM and I-vector_DNN systems. In the end, we test the generalization of the trained models on the SITW dataset.

Table 2.8 presents the results for regular PLDA training condition (baseline), matched-length PLDA condition, LDA dimensionality reduction method, DNN-based direct mapping method, DNN-based two-stage method (DNN_1) and DNN-based single-stage method

(DNN_2, α=0.5). We observe that matched-length PLDA training gives considerable improvement compared with non-matched PLDA training (baseline), which is consistent with previous work. When training the PLDA using short-utterance i-vectors, the system can capture the variance of short-utterance i-vectors. Using LDA to do dimension reduction also results in some improvement, since it reduces the variance of the i-vectors. DNN-based direct mapping gives more improvement for female speakers (13.93%) compared with male speakers (5%) in terms of EERs, and it may be due to the fact that more training data are available for female speakers and thus the over-fitting problem is less severe for females. In the last two rows, we show the performance of our proposed DNN-based mapping methods on short-utterance i-vectors. From the results, we can observe that they both result in significant improvements over the baseline for both the EER and minDCF metrics, and they also outperform the other short-utterance compensation methods by a large margin. DNN_1 and DNN_2 methods have comparable performance, which prove the importance of learning joint representation of both short and long utterance i-vectors. The proposed methods outperform the baseline system by 22.13% for female speakers and improve the male speaker baseline by 24.51%. One of the advantages using DNN_2 is that the unified framework avoids using the two-stage training procedure, which speeds up the training.

2.4.5.3 Effect of pre-training for DNN_1

In this section, we will show how first-stage pre-training influences the second-stage mapping training for DNN_1. We investigated the number of training iterations used for first-stage pre-training from 10000-50000. What we find interesting is that when the number of training iterations is small, the second stage fine-tuning will over-fit the data, but when the number of training iterations is large, the fine-tuning results are not optimal. In the end, 25000 iterations was a roughly good initialization for second stage fine-tuning. This indicates that the number of iterations for unsupervised training does influence the second-stage supervised training.

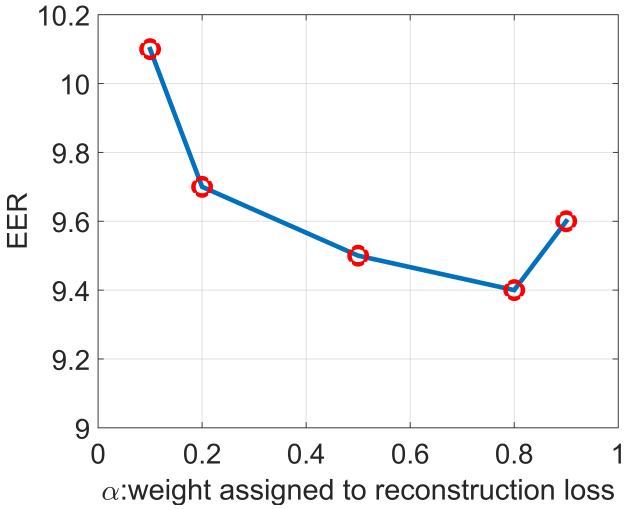


Figure 2.9: EER as a function of reconstruction loss α for DNN_2.

2.4.5.4 Effect of reconstruction loss for DNN_2

In this section, we investigate the impact of the weights for the reconstruction loss in DNN_2. We set $\alpha = \{0.1,0.2,0.5,0.8,0.9\}$. Since the weight of regression loss is $1 - \alpha$, the larger α is, the less weight will be assigned to regression loss. Figure 2.9 shows the EER for female speakers as a function of the weights assigned to reconstruction loss. The reconstruction loss is clearly important for this joint learning framework. It forces the network to learn the original representations for short utterances, which can regularize the regression task and generalize better. The optimal reconstruction weight is $\alpha = 0.8$, which indicates that the reconstruction loss is even more important for this task. Hence, it appears that unsupervised learning is very crucial for a speaker recognition task.

2.4.5.5 Effect of encoder depth

The depth of a neural network has been proven to be important for network performance. Adding more layers will make the network more efficient and powerful to model the data. Therefore, as discussed in Section 2.4.3.3, we will compare a shallow (2-layer) and a deep (5-layer) encoder for both DNN_1 and DNN_2. It is well known that training a deep model suffers from gradient vanishing/exploding problems and also it can be easily stuck into

Table 2.9: DNN-based mapping results using DNNs with different depths in the 10 s-10 s condition of NIST SRE10 database.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
baseline	12.2	0.054/0.093	10.2	0.048/0.095
DNN1 mapping (3 layer)	9.5 (22.13%)	0.047/0.091	7.7 (24.51%)	0.039/0.090
DNN2 mapping (3 layer)	9.5 (22.13%)	0.047/0.091	7.7 (24.51%)	0.039/0.089
DNN1 mapping (6 layer + residual block)	9.1 (25.41%)	0.046/0.091	7.5 (26.47%)	0.038/0.089
DNN2 mapping (6 layer + residual block)	9.3 (23.77%)	0.047/0.091	7.6 (25.49%)	0.038/0.089

local minimum points. Therefore, we use two methods to alleviate this problem. Firstly, as stated in Section 2.4.3.3, we use a normalized initialization (Xavier initialization) and a batch normalization layer to normalize the intermediate hidden output. Secondly, we apply residual learning, which uses several residual blocks (defined in Section 2.4.2.1) with no extra parameter compared with regular fully-connected layers. The residual blocks will make the information flow between layers easy and enable very smooth forward/backward propagation, which makes it feasible to train deep networks. To our knowledge, this is one of the first studies to investigate the effect of residual networks for the auto-encoder and unsupervised learning. Here, for the deep encoder, we use 2 residual blocks and 1 fully-connected bottleneck layer (in total 5 layers). For the decoder, we use a single hidden layer with a linear regression output layer.

From Table 2.9, we can observe that a deep encoder does result in improvements compared with a shallow encoder. Especially for DNN_1, the residual networks give a 25.41% relative improvement for female speakers and 26.47% relative improvement for male speakers. The results indicate that learning a good joint representation of both short and long utterance i-vectors is very beneficial for this supervised mapping task, and the deep encoder can help learn a better bottleneck joint embedding. The deep encoder can also decrease the amount of training data needed to model the non-linear function, which can also alleviate the over-fitting problem. In order to show the effect of residual short-cuts, we performed experiments using a deep encoder without short-cut connections, and the system resulted in even worse

Table 2.10: DNN-based mapping results with additional phoneme information in the 10 s-10 s condition of NIST SRE10 database.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
baseline	12.2	0.054/0.093	10.2	0.048/0.095
DNN mapping	9.1 (25.41%)	0.046/0.091	7.5 (26.47%)	0.038/0.089
DNN mapping + phoneme info	8.9 (27.05%)	0.046/0.090	7.3 (28.43%)	0.037/0.090

performance compared with the shallow encoder. Therefore, residual blocks with short-cut connections are very crucial for deep neural network training, since they alleviate the hard optimization problems of deep networks.

2.4.5.6 Effect of adding phoneme information

In this section, we show the results when adding phoneme vector (mean of phoneme posteriors across frames) with short-utterance i-vectors to learn the mapping. We will investigate the effect of adding phoneme information based on the best performed DNN-mapping structures. From Table 2.10, we observe that when adding a phoneme vector, the EER further improves to 8.9% for female speakers and 7.3% for male speakers from the previous best DNN-mapping results, and achieves the best results for this task. The results prove the hypothesis that adding a phoneme vector can help the neural network reduce the variance of short-utterance i-vectors, which will lead to better and more generalizable mapping results. In Section 2.4.5.8, we will also show the effect of adding phoneme vectors to GMM-i-vectors.

2.4.5.7 Results with different durations

In this section, the results for different durations of evaluation utterances are listed. Table 2.11 shows the baseline and the best mapping results for 10 s-10 s, 5 s-5 s and mixed duration conditions. From the table, we observe that the proposed methods give significant improvements for both 10 s-10 s and 5 s-5 s conditions, which indicates that the proposed method generalizes to different durations. In real applications, however, the duration of short utterances can not be controlled, therefore we train the mapping using the i-vectors

Table 2.11: DNN-based mapping results with different utterance durations on NIST SRE10 database.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
10 s-10 s				
baseline	12.2	0.054/0.093	10.2	0.048/0.095
DNN mapping (best)	9.1 (25.41%)	0.046/0.091	7.5 (26.47%)	0.038/0.089
5 s-5 s				
baseline	19.9	0.078/0.099	17.0	0.072/0.100
DNN mapping (best)	14.8 (25.62%)	0.067/0.099	13.5 (20.59%)	0.061/0.100
mix				
baseline	17.8	0.068/0.097	14.4	0.061/0.100
DNN mapping (best)	13.2 (25.84%)	0.061/0.097	11.8 (18.06%)	0.053/0.096

generated from mixed 10 s and 5 s utterances and show the results also on a mixed-duration evaluation task (mixed of 5 s and 10 s). From Table 2.11, we see that the baseline results for the mixed condition range between the EER results of 10 s-10 s and the 5 s-5 s evaluation tasks. The proposed mapping algorithms can model i-vectors extracted from various durations, and thus give consistent improvement as shown in the table.

2.4.5.8 Comparison of mapping results for both I-vector_GMM and I-vector_DNN systems

In the previous sections, we only show the mapping experiments for an I-vector_DNN system, therefore, in this section, we will show the mapping results for the I-vector_GMM system. In Section 2.4.5.1, we show that for the baseline results, I-vector_DNN system outperforms the I-vector_GMM system, but it is also interesting to compare the results after mapping. From Table 2.12 we observe that the proposed mapping methods give significant improvement for both systems. After mapping, the I-vector_DNN systems still outperform the I-vector_GMM systems and the superiority of I-vector_DNN systems is even more significant. We also compare the mapping results when adding phoneme vectors. The table shows that the effect of adding phoneme information is more significant for GMM-i-vectors and it can achieve as much as a 10% relative improvement on the best DNN mapping baseline. The reason is

Table 2.12: Results for I-vector_GMM and I-vector_DNN systems in the 10 s-10 s conditions of NIST SRE10 database.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
I-vector_GMM				
baseline	13.8	0.063/0.097	13.3	0.057/0.099
DNN mapping (best)	11.0 (20.29%)	0.054/0.095	10.6 (20.30%)	0.051/0.096
DNN mapping (best) + phoneme info	10.4 (24.64%)	0.053/0.094	9.6 (27.82%)	0.048/0.096
I-vector_DNN				
baseline	12.2	0.054/0.093	10.2	0.048/0.095
DNN mapping (best)	9.1 (25.41%)	0.046/0.091	7.5 (26.47%)	0.038/0.089
DNN mapping (best) + phoneme info	8.9 (27.05%)	0.046/0.090	7.3 (28.43%)	0.037/0.090

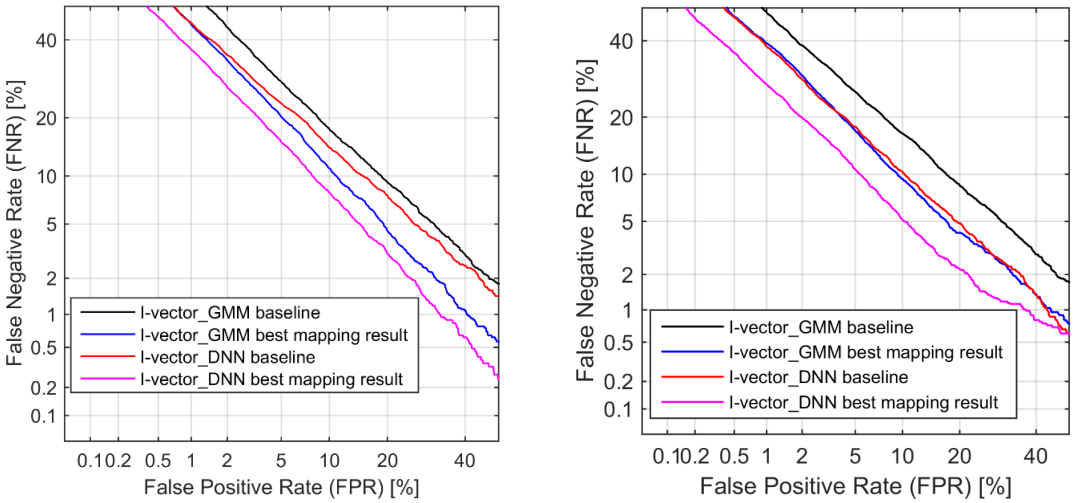


Figure 2.10: DET curves for the mapping results of I-vector_GMM and I-vector_DNN systems under 10 s-10 s conditions of NIST SRE10 database. Left figure corresponds to female speakers and right one corresponds to male speakers.

that DNN-i-vectors already contain some phoneme information, while GMM-i-vectors do not have clear phoneme representation. Therefore GMM-i-vectors can benefit more from adding phoneme vectors. In the end, we summarize the baseline and the best mapping results for both systems in Figure 2.10. The DET (Detection Error Trade-off) curves are presented for both female and male speakers. The figures indicate that the proposed mapping algorithms give significant improvement from the baseline across all operation points.

Table 2.13: DNN-based mapping results on SITW using arbitrary durations of short utterances.

	Female		Male	
	EER (Rel Imp)	DCF08/DCF10	EER (Rel Imp)	DCF08/DCF10
Arbitrary durations				
baseline	17.3	0.061/0.089	12.0	0.046/0.083
DNN mapping (best models from SRE10)	13.3 (23.12%)	0.050/0.086	9.4 (21.67%)	0.039/0.078

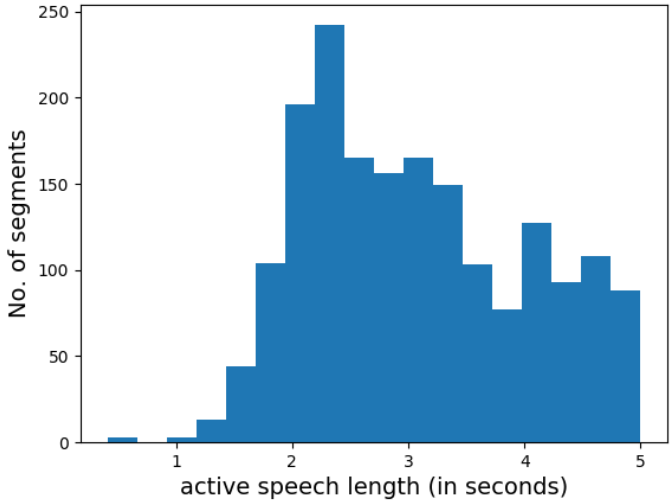


Figure 2.11: Distribution of active speech length of truncated short utterances in the SITW database.

2.4.5.9 Performance on the SITW database

In previous subsections, we showed the performance of our proposed DNN-mapping methods on NIST data. In this subsection, we apply our technique on the recently published database SITW which contains real-world audio files collected from open-source media channels with considerable mismatch conditions. In order to generate a large number of random-duration short utterances, we first combined the dev and eval datasets and then selected 207 utterances from relatively clean condition. We truncated each of 207 utterances into several non-overlapped short utterances with duration 5 s, 3.5 s, 2.5 s (including both speech and non-speech portions). In the end, a total of 1836 utterances were generated. We plot the distribution of active speech length across these 1836 utterances in Figure 2.11. From the

figure, we observe that the active speech length varies between 1 s-5 s across those short utterances. Therefore, we can use these short utterances to design trials, which represent real-world conditions (arbitrary-length short utterances). In total, we designed 664672 trials for our arbitrary-length short-utterance speaker verification task.

For each short utterance, we first down-sampled the audio files to 8 kHz sampling rate, and then extracted the i-vectors using the previously trained I-vector_DNN system introduced in Section 2.4.3.1. For PLDA scoring, we use the same PLDA in Section 2.4.3.1, which is trained using the SRE dataset. For i-vector mapping, we use the best-validated models on SRE10 dataset (5 s condition) to apply to the SITW dataset. Evaluation results of EERs and minDCFs are show in Table 2.13. From the table, we can observe that the best models validated on SRE10 dataset generalize well to the SITW dataset, which give a 23.12% relative improvement of EERs for female speakers and a 21.67% relative improvement for male speakers. The results also indicate that the proposed methods can be used in real-life conditions, such as smart home and forensic applications.

2.5 Conclusion

In this chapter, we explored two different methods to learn speaker-specific and phoneme-invariant representation from short-duration utterances. Both methods made use of DNN models to learn non-linear mappings. We first learned speaker-specific and phoneme-invariant subglottal features from speech spectra at the frame level. The estimated subglottal features have higher J-ratios and show complementary information when combined with MFCC features on speaker verification tasks. We then explored an utterance-level method, which learns to reconstruct the long-utterance i-vector from its short-utterance version. Long-utterance i-vectors have less within-speaker variation and can provide richer speaker information. The proposed mapping algorithms using autoencoders improve the speaker verification performance significantly. The proposed utterance-level representation learning method is more effective than the frame-level one.

CHAPTER 3

Joint feature learning and acoustic modeling for automatic speech recognition

3.1 Introduction

Standard speech recognition systems use hand-designed features to train a classifier. Designing a feature extraction algorithm and building an appropriate classifier have often been treated as separate problems in the speech community. However, one drawback of these systems is that the designed features might not be best for the classification task at hand.

Therefore, we design several new neural network models which take complex DFT features or raw waveforms as input, and perform the feature extraction and phone classification jointly. These unified frameworks can tune the features to the target classification task, and thus improve classification accuracy. This work was published in [GKS18] and [GXC18].

3.2 Related work

Recently, a great deal of attention has been paid to training ASR acoustic models directly from raw waveforms or Fourier coefficients. In [BR15], feedforward DNNs were adopted to learn features from raw waveforms. The authors in [GMP16] proposed a network-in-network architecture that uses CNN filters to extract features from the signals and showed significant improvement over MFCCs. Similarly, shallow CNN models were used in [HWW15, PC15] for a noise-robust ASR task. In [SWS15], a combination of convolutional layers and long short-term memory (LSTM) layers was proposed to show the effectiveness of raw-waveform modeling. Moreover, a complex linear projection (CLP) layer with LSTM layers was pro-

posed in [VSS16], which achieved superior performance compared to filterbank features.

3.3 Feature learning in the frequency domain

The most common ASR feature extraction method applies auditory-inspired Mel-filter banks to the squared magnitude of the short-time Fourier transform of the windowed speech signal. Filter parameters are inspired by knowledge of the human speech perception. Filter bank outputs are then used to train an acoustic model (AM). These perceptually motivated filterbank features are not always the best choice in statistical modeling frameworks such as ASR acoustic modeling, where the end goal is phone classification. This motivates the data-driven learning schemes for joint feature learning from audio signals and acoustic modeling. Audio signals can be represented in both the time and frequency domains. Applying the Discrete Fourier Transform (DFT) on time-domain waveforms, can normalize the more variant waveform signals (due to phase shifts and temporal distortions) and generate equivalent feature representation (complex DFT features), to make it easier to train acoustic models, such as neural-network based models.

In this section, we propose a unified Highway network with a time-delayed bottleneck layer (TDB-HW), to learn features automatically from complex DFT features. Our TDB-HW network has two parts: a feature extractor and a phone classifier. Each of them consists of stacked highway blocks, which can control the information flow between layers and make it feasible to train a very deep neural network. The bottleneck layer can force the network to learn the most salient features. The proposed TDB-HW network can be trained from scratch without stage-wise training, which can save a significant amount of training time. We will evaluate the proposed model on a large scale keyword spotting (wake-word detection) task.

3.3.1 Baseline Wake-word Detection System

Wake word detection is the first important step before interactions through voice commands. In this section, we first employ the HMM-based baseline approach with the wake-word (WW) and filter/background HMMs [PSK16]. Figure 3.1 illustrates an example of the finite state

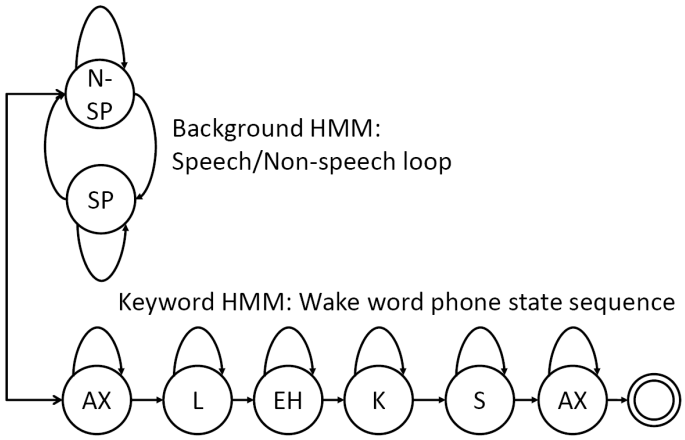


Figure 3.1: HMM-based Keyword Spotting.

transducer (FST) at the phone level for a WW, with six phones in the word "ALEXA". Each phone state is further divided into HMM states. The HMM state is associated with a DNN. The output layer of the DNN models the HMM states of the keyword(s) of interest (i.e., WW-specific phone state distributions) and the two 1-state background speech and non-speech nodes.

Figure 3.2 shows a schematic view of the baseline DNN system. The baseline system first computes the Log Mel-filterbank Energy (LFBE) feature from the enhanced speech using beamforming [PSK16]. In our system, an audio signal is divided into overlapping frames of 25 ms with a frame shift of 10 ms. LFBE features concatenated over multiple frames are then fed into the acoustic-modeling DNN. The DNN consists of several layers of affine transform and sigmoid activation components. In addition to those layers, we insert two separate branches for WW and ASR tasks to jointly classify the WW-specific phones and LVCSR senones (cluster of triphones) based on our previously proposed multi-task training technique [PSK16]. After the DNN is pre-trained layer by layer in a supervised fashion using a small subset of the training data, the entire DNN is further optimized with a distributed, asynchronous, stochastic gradient descent (SGD) training method [Str15] over the full dataset.

As illustrated in the FST of Figure 3.1, the WW hypothesis is generated when the final state of the WW FST is reached. We tune transition parameters and exit penalties in the

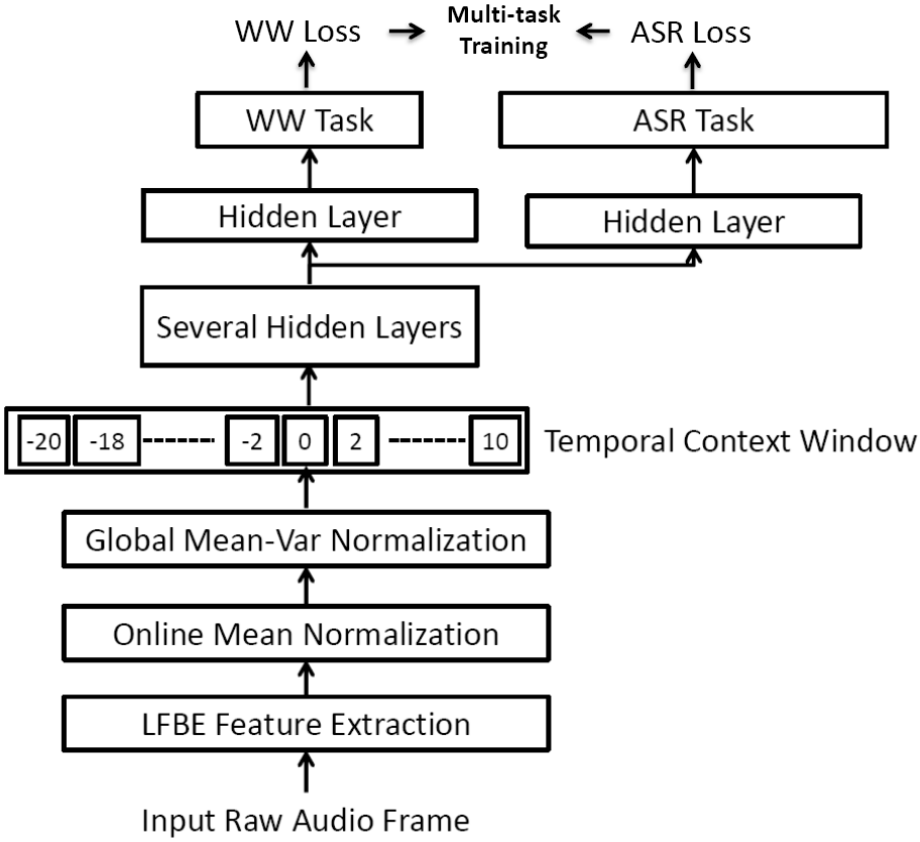


Figure 3.2: Baseline WW DNN with the LFBE feature.

WW and background HMMs for better accuracy, and a detection error trade-off (DET) curve can be obtained by plotting the lowest achievable false alarm rate (FAR) at a given miss rate (MR) or false reject rate (FRR).

3.3.2 DFT-Input Highway networks

In this section, we will introduce the highway blocks, and show the structures of the proposed TDB-HW networks. We will also list different structures for comparison.

3.3.2.1 Highway Blocks

Highway networks were first proposed by [SGS15], and the basic element of the network is the highway block. In the highway block, the output of the l-th layer is controlled by two

gating functions: a carry gate $C(\boldsymbol{h}_{l-1})$, that controls the information flow directly from the previous hidden layer ($\boldsymbol{h}_{l-1}$), and a transform gate, $T(\boldsymbol{h}_{l-1})$ that controls the information from the hidden activations of the current layer ($f(\boldsymbol{h}_{l-1})$). The final output is defined by:

$$\boldsymbol{h}_l = f(\boldsymbol{h}_{l-1}) \cdot T(\boldsymbol{h}_{l-1}) + \boldsymbol{h}_{l-1} \cdot C(\boldsymbol{h}_{l-1}) \tag{3.1}$$

Both carry and control gate functions are defined by a nonlinear layer with a sigmoid function:

$$T(\boldsymbol{h}) = \sigma(\boldsymbol{W}_T\boldsymbol{h} + \boldsymbol{b}_T), \tag{3.2}$$

$$C(\boldsymbol{h}) = \sigma(\boldsymbol{W}_C\boldsymbol{h} + \boldsymbol{b}_C) \tag{3.3}$$

From our preliminary experiments, we observed that the HW network was easier to train without a bias vector. Therefore, we do not use the bias vector in the gate function. Moreover, in contrast to [SGS15], we do not impose any constraint on the two gates. The highway blocks can control information flow and gradient propagation between layers, which makes it feasible to train a very deep neural network and can also speed up the convergence rate.

3.3.2.2 Highway networks with the time-delayed bottleneck layer

In order to make the DNN learn the feature representation from complex DFT features generated from the linear signal normalizer, we first use 4 stacked HW blocks and a bottleneck layer as the feature extractor. The bottleneck layer can reduce the large dimension of the input features (concatenated complex DFT coefficients), which will force the network to learn the most salient representations. The design of the bottleneck layer can also significantly reduce the network size, which makes it feasible for resource-constrained conditions. In this architecture, we use a linear bottleneck layer since experimental results indicate that a linear layer performs slightly better than a non-linear layer. After the bottleneck layer, we use a time-delayed window to splice the bottleneck features from several past and future frames

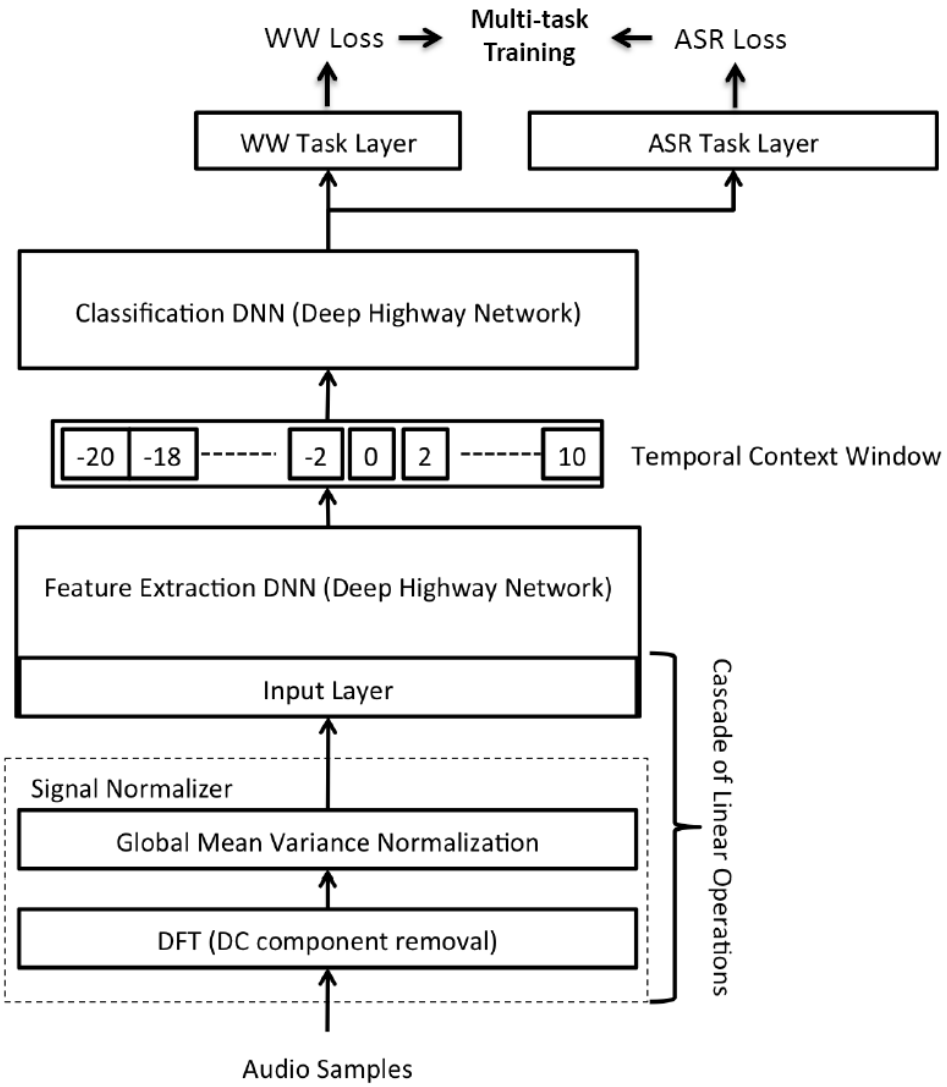


Figure 3.3: Whole WW Highway DNN with the DFT input.

so as to capture the temporal information for phone classification. For the classification DNN, we use 6 HW blocks stacked together. Figure 3.3 illustrates our entire highway DNN architecture with the DFT input for WW.

In the bottleneck layer, we use 28 hidden nodes. For the time-delayed window, we select 20 left and 10 right contexts from the bottleneck layer output. For back-propagation, the weights are updated when the gradients are accumulated from all the contexts in the window. In order to reduce the number of parameters, we tie the two gate weights of each layer inside the feature extraction DNN and also inside the classification DNN. The entire DNN is also

optimized based on the multi-task cross-entropy criterion. The TDB-HW networks are directly trained from scratch with random initialization. Here random initialization refers to light supervised pre-training in a layer-wise manner on a small subset of the training data.

3.3.2.3 Compared architectures

In this section, we will compare the TDB-HW networks with TDB-DNNs using complex DFT features. For the TDB-DNNs, we follow the same three-stage training procedure as described in [KPW17]: first, we need to train a feature extraction DNN with a bottleneck layer on top. Then, we use a context window to splice the bottleneck features across several frames and use the stacked features to train the acoustic modeling DNNs. Finally, we need to jointly optimize the feature extraction and acoustic modeling DNNs, by training the unified network as a DNN with a time-delayed bottleneck layer in the middle.

We will also compare the complex DFT systems with LFBE systems. In order to have a fair comparison, besides the feed-forward DNN baseline system described in Section 3.3.1, we also design a regular HW network (tie the two gate weights for each layer) using LFBE.

All the compared networks have the same depth (11 layers) and a similar number of parameters (around 3 M).

3.3.3 Experiments and results

The results are shown in the form of DET curves along with area under the curve (AUC) numbers. The DET curves and AUC numbers presented here indicate the relative improvement or degradation against the baseline system.

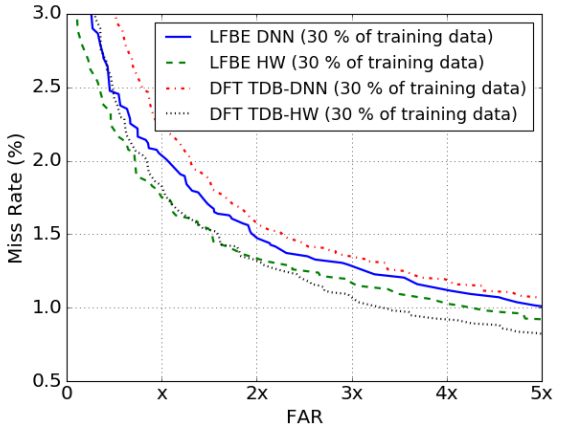
Training data used in this work consist of several thousand hours of the real far-field data captured in various rooms, and contains approximately several hundred thousand subjects. In order to improve the robustness against noise unseen in the training data, the training data are artificially corrupted and the SNR is adjusted from 0 to 40 dB uniformly. Our test set contains over several thousands of speech segments uttered by hundreds of subjects. The test data were recorded over three months and contain approximately 26,000 speech

instances. The captured far-field array data are processed with beamforming and acoustic echo cancellation [WM09].

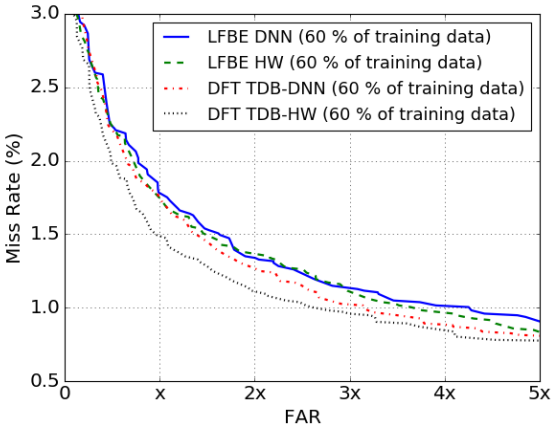
3.3.3.1 Comparison of different DNN architectures

Figure 3.4 shows the DET curves obtained with the LFBE DNN, LFBE HW, DFT TDB-DNN and DFT TDB-HW on the test set with different amounts of training data. In order to generate the DET curves for Figure 3.4, we choose the best Finite State Transducer (FST) parameters with 4 HMM thresholds. Since we choose the FST parameters from the same pool of the FSTs which was selected empirically beforehand, this comparison is fair. DET curves on the test data indicate the best possible WW performance without the grammatical language constraint.

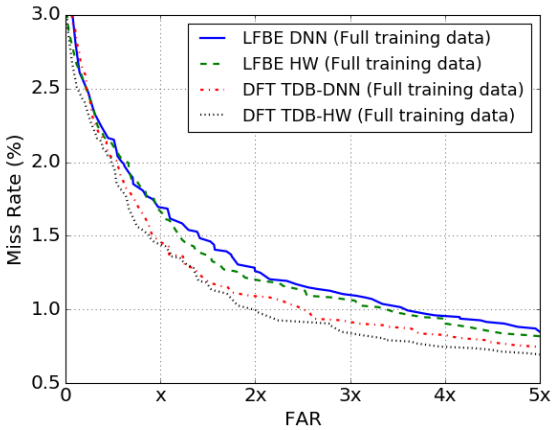
It is clear from Figure 3.4 that the HW-based networks provide better accuracy than the DNN-based networks with both LFBE and DFT features. The effect of HW blocks is very prominent on training a deep network, especially under the 30%-training-data condition. As the amount of training data increases, the difference between the regular HW networks and DNNs becomes smaller for the LFBE system. This indicates that the hard optimization problem of a deep network can be alleviated by a large amount of training data. For the DFT systems, the improvement of TDB-HW networks is still significant compared with TDB-DNN, even when trained on a large amount of data. The good performance of the TDB-HW networks may rely on its ability to train a unified structure from scratch (without stage-wise training). By jointly optimizing the feature extractor and phone classifier using HW blocks from scratch, the TDB-HW networks are able to learn the most useful features, which are highly optimized for phone classification. TDB-DNN's three-stage-training procedure, on the other hand, may not be able to achieve global optimization. In general, the proposed DFT system is clearly better than LFBE systems. From the AUC numbers in Figure 3.5 we can observe that, using full training data, the proposed DFT TDB-HW networks can outperform the baseline system (LFBE DNN) by 19.4%.



(a) 30 % of training data



(b) 60 % of training data



(c) Full amount of training data

Figure 3.4: DET curves of LFBE DNN, LFBE HW, DFT TDB-DNN, DFT TDB-HW using different amounts of training data

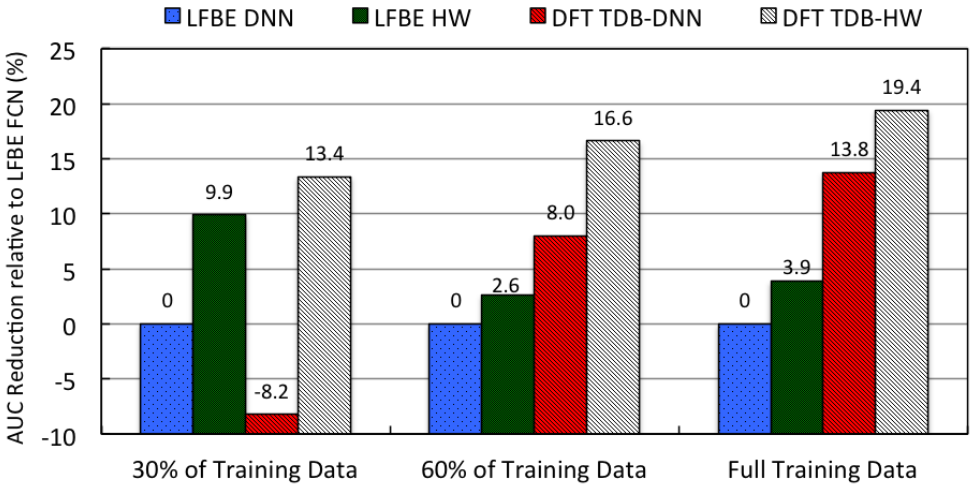
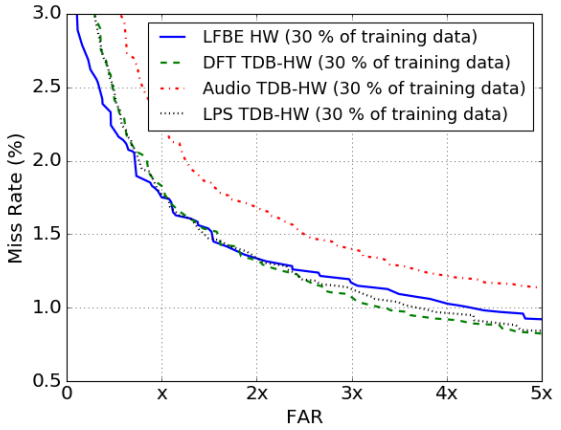


Figure 3.5: AUCs calculated from Figure 3.4.

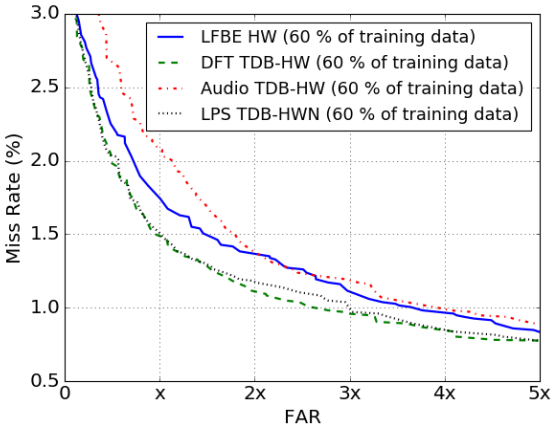
3.3.3.2 Effect of different feature inputs

From the speech feature extraction point of view, it could be interesting to investigate the effect of different features on the HW network input. Figure 3.6 shows the DET curves of the HW-based networks obtained with the LFBE (LFBE HW), DFT coefficients (DFT TDB-HW), raw audio (AUDIO TDB-HW) and log-power spectrum (LPS) (LPS TDB-HW) features. Figure 3.7 shows the AUC graphs that correspond to Figure 3.6. It is clear from Figures 3.6 and 3.7 that, when using 30% of training data, the LFBE, DFT and LPS features have similar performance. As the number of training data increase, the DFT coefficients provide the best performance, which is slightly better than LPS features and significantly better than raw audio and LFBE features. The results indicate that the phase information (which are kept for only the complex DFT features) can provide slight improvement for acoustic modeling and a fully trainable front-end can provide significant improvement compared with auditory-based features (LFBE). The AUC numbers in Figure 3.7 shows that the DFT TDB-HW network gives more than 16% improvement compared with the LFBE HW network, when using 60% or 100% of training data.

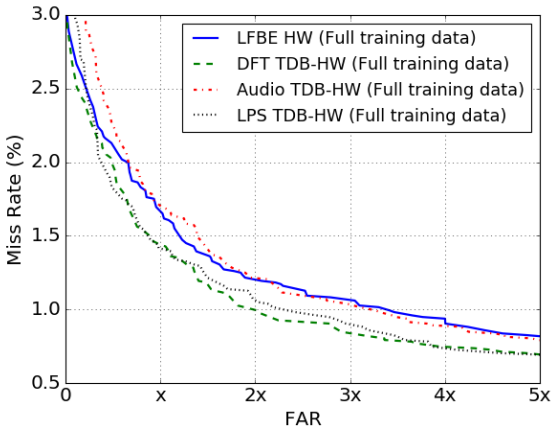
The DFT TDB-HW network uses the linear-normalized raw audio input, without any non-linear pre-processing. While the LPS feature's computation involves a non-linear process (e.g. logarithm) which takes more computation. For the raw-audio-input condition, we believe that the raw audio DNN may easily converge to a trivial local minima due to the absence of adequate normalization, and proper initialization, also indicated in Bhargava's work [BR15] .



(a) 30 % of training data



(b) 60 % of training data



(c) Full amount of training data

Figure 3.6: DET curves of LFBE HW, DFT TDB-HW, Audio TDB-HW, LPS TDB-HW using different amounts of training data

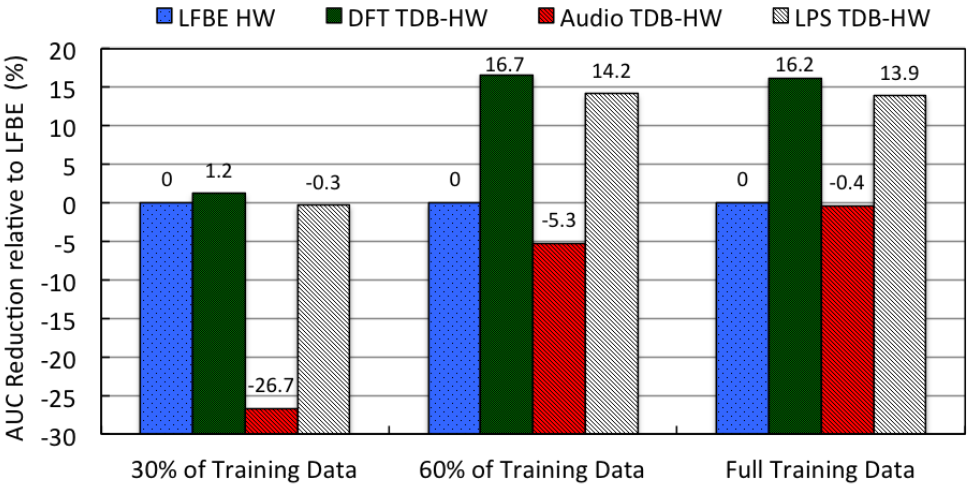


Figure 3.7: AUCs calculated from Figure 3.6

3.4 Feature learning from raw waveforms

The previous section explores complex DFT feature learning and acoustic modeling. In this section, we will focus on feature extraction using neural networks from time-domain raw waveforms.

3.4.1 CNN-based acoustic modeling using raw waveforms

3.4.1.1 Raw waveform feature extraction and modeling

As discussed in Section 3.3, the most popular features to train acoustic models are Mel-filterbank-based features. The mel filter banks are inspired by of human perception, and not always guaranteed to be the optimal features in a statistical modeling framework. In order to train learnable filterbanks, which are optimum for the end goal of classification, a 1-D convolutional layer can be applied to the raw time-domain waveforms. 1-D convolutional filters can be thought of as finite impulse-response filterbanks. Such a layer is capable of approximating filterbanks, such as Mel filterbanks. Moreover, to alleviate the large variations of waveform signals due to phase shifts and temporal distortions, pooling layers can be applied to extract invariant features. Stacked convolutional layers and pooling layers can efficiently perform hierarchical feature extraction. Therefore, in this section, we will introduce 1-D fully convolutional neural networks for acoustic modeling using raw waveforms.

3.4.1.2 1-D CNN baseline system

We first introduce our baseline system, which uses a CNN-based neural network architecture and raw waveforms as input for acoustic modeling.

The input features to the neural network are long duration segments (110ms or 1760 samples) of raw waveform signals. The raw waveforms are mean and variance normalized at the speaker level. For the neural network architecture, we propose a deep 1-D CNN model to extract features from raw waveforms as shown in Table 3.1. This CNN structure has 7 convolutional layers and 2 fully-connected layers. The first three layers have larger filter sizes

Table 3.1: Proposed 1-D CNN structure (the last column shows the number of parameters for each layer).

Layer	**Filter size**	**#filters**	**#para**
$conv1$	32	32	$1K$
$conv2$	32	64	$65K$
$conv3$	16	128	$131K$
$conv4$	8	128	$131K$
$conv5$	8	256	$262K$
$conv6$	8	512	$1M$
$conv7$	4	512	$1M$
$fc1$	2048	512	$1M$
$fc2$	512	512	$262K$

in order to capture larger reception fields, such that more useful low-level features can be extracted from raw waveforms. Filters from the first layer are expected to learn perception-based filterbanks automatically. The last four convolutional layers have a smaller filter size but a larger number of filters, which can efficiently extract higher-level features. The output of each convolutional layer is fed into a max-pooling layer with stride equal to two, in order to reduce the dimension of the feature maps and extract invariant features. In the end, two fully-connected layers are stacked, which can transform the extracted features into a space for discriminative classification. A fully-connected layer can be also treated as a 1-D CNN layer which has larger-size filters with depths equal to one.

In this section, since we are aiming at designing compact and efficient neural network models for ASR, the parameters of the baseline CNN model are highly optimized and well designed. The total number of nonlinear-layer parameters of the baseline CNN is around 3.84M.

3.4.2 Filters learned from raw waveforms

The proposed 1-D CNN model uses independent filters to learn representations from the input. In order to investigate the relationship between filters, we plot filters that are learned from the convolutional layers in Figure 3.8. We select 6 one-dimensional filters from each layer. From the figure, we can observe that the learned filters in the same layer are redundant.

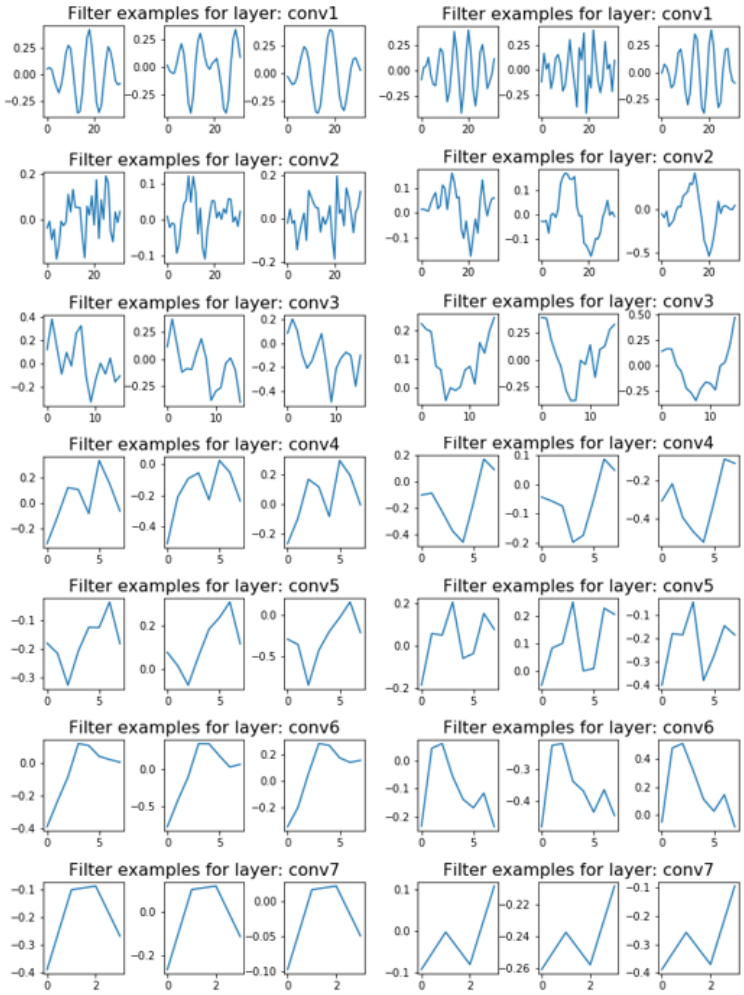


Figure 3.8: Filters learned from each conv layer (order from top to bottom). Each row represents a layer and 6 different filters from that layer are shown.

Many of the filters have similar shapes but different ranges. This motivates us to use fewer parameters to represent the filters and still keep their learning ability. Therefore, in this section, we propose a novel filter generation method, which first samples the filter parameters from a low dimensional space of parameters, and then uses a set of trainable scalar vectors to perform a linear combination. By doing filter sampling and combination, we are able to get various filters which share weights in a hidden low-dimensional space. The technique can also alleviate over-fitting problems by introducing a smoother loss function, which is modeled with fewer parameters. In the following subsection, we will introduce the proposed filter sampling and combination method in detail.

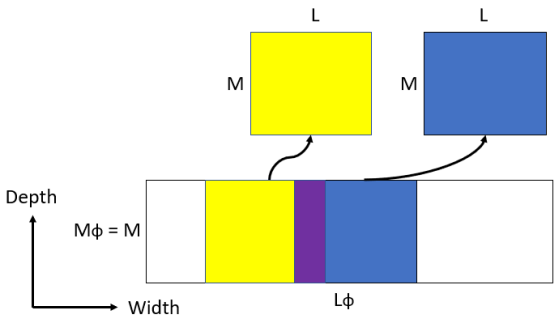


Figure 3.9: Widthwise filter sampling in space Φ.

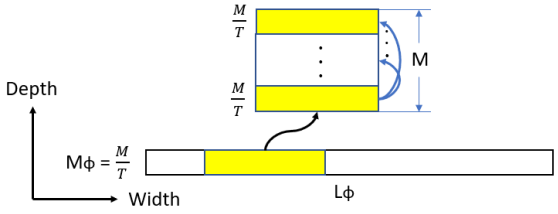


Figure 3.10: Depthwise filter sampling in space Φ.

3.4.3 Filter sampling and combination CNN

3.4.3.1 Filter sampling

For a given convolutional layer, the width of a filter is defined as the filter size of each 1-D filter, and the depth of a filter equals the number of feature maps output from the last layer. Let L denote the filter width and M denote the filter depth.

For each convolutional layer, all filters are sampled from a low dimensional filter-sampling space Φ as illustrated in Figure 3.9. Φ is a two-dimensional space with width L_ϕ and depth M_ϕ.

To start, we set the depth of the sampling space M_ϕ equal to the depth of filter M and only do the sampling along the width. We then use a sliding window with the same size L and depth M as the filters to do the sampling in Φ, as shown in Figure 3.9. The stride (or the skipping step) of the sliding window is denoted as S. The smaller S is, the more parameter sharing among filters. Suppose that we sample N filters from Φ, then the equation between

the width of Φ and the size and stride of the window is:

$$L_\phi = NS + L - S. \tag{3.4}$$

The filter sampling method ensures that each generated filter shares the weights with adjacent filters, such that the number of independent parameters is reduced significantly. The compression ratio for widthwise sampling is $\frac{LN}{L_\phi} \approx \frac{L}{S}$, given that N is much larger than L.

Besides the widthwise sampling, we can also perform depthwise sampling by repeating the sampling several times, as illustrated in Figure 3.10. We set the depth of sampling space M_ϕ to be $\frac{M}{T}$, where T denotes the number of times sampling is repeated (i.e. reusing the same parameters) so that the compression ratio for depthwise sampling is $\frac{M}{M_\phi} = T$. Note that both widthwise and depthwise sampling can be applied simultaneously to generate filters.

3.4.3.2 Filter combination

Directly applying filter sampling can reduce the number of parameters significantly. However, simply tying the weights between adjacent filters will limit the learning ability of filters. Therefore, in order to avoid tying parameters directly between filters, we introduce a set of trainable scalar vectors α_i as in Eq. 3.5, where $i = 1, 2, ..., N$ and N is number of filters for a given layer. Each α_i consists of M scalar α_{ij}, where $j = 1, 2, ..., M$, and M is the depth of the filters. Let F_i denote the i_{th} generated filter from the filter sampling step as in Eq. 3.5, where F_{ij} is a filter of size L in j_{th} depth of filter F_i . Each α_{ij} is multiplied to F_{ij} to generate a new set of filters $\hat{F}_i$ as shown in Eq. 3.5.

$$\alpha_i = \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{iM} \end{bmatrix}, F_i = \begin{bmatrix} F_{i1} \\ F_{i2} \\ \vdots \\ F_{iM} \end{bmatrix}, \hat{F}_i = \begin{bmatrix} \alpha_{i1} F_{i1} \\ \alpha_{i2} F_{i2} \\ \vdots \\ \alpha_{i2} F_{iM} \end{bmatrix} \tag{3.5}$$

The new filter $\hat{F}_i$ can be interpreted as a linear combination of the original filter F_i, and the linear weights are all trainable and optimized based on the final loss of the neural network. The proposed filter combination method ensures that all the generated filters will now have unique parameters and they still naturally share the weights in the hidden sampling space Φ.

By adding a set of scalar vectors α_i, we introduce $M * N$ extra parameters for each layer. The motivation of adding α_i is to ensure that each generated filter will have different parameters to model them, and the filters generated from the filter-sampling step only share the weights with a limited number of filters. Therefore, it is not necessary to use $M * N$ independent parameters to represent α_i, and instead we can tie the weights of α_i, such that each generated filter has a unique combination of weights from Φ and scalars from α_i . Similar to the idea of weight sampling, we can tie the weights of either dimension M or N. If we do the widthwise filter sampling in the first step, we can tie the weights of α_i along dimension N by a ratio of R_N (i.e. weights are repeated every $\frac{N}{R_N}$); if depthwise filter sampling is conducted, the weights of α_i can be tied along dimension M by a ratio of R_M. By applying weight tying, we can significantly reduce the number of parameters needed for the linear combination step.

The above weight sampling and combination method can be conveniently generalized from convolutional layers to fully connected layers. As mentioned in Section 3.4.1.2, for a fully-connected layer, since its input has a single channel, its weights can be treated as filters with depth one. Those filters have large filter sizes which are equal to the size (vector dimension) of the input to that layer. Since the depth of the filters already equals to one, we can only perform widthwise filter sampling for fully-connected layers.

3.4.4 Experiments and results

3.4.4.1 Evaluation setup

We conduct our experiments on 80 hours of speech data using the Wall Street Journal (WSJ) continuous speech corpus [PB92]. We use the standard configuration: si284 dataset

for training, dev93 for validation and eval92 for testing. We compare with filterbank and raw waveforms based systems. For filterbank feature based systems, we use 40-dim Mel-filterbank features normalized on a per-speaker level, which are then spliced by a context window of 11 frames (i.e. ± 5). DNNs and CNNs with various sizes are compared. For raw waveforms based systems, in order to make a fair comparison, we use 110ms raw waveforms as input, which covers the same context as Mel-filterbank features. The raw waveforms are also normalized on a per-speaker level. For the raw waveforms baseline, we use the 1-D CNN structure as introduced in Section 3.4.1.2. The number of tied tri-phone states is 3362 and all the neural network systems are trained with the same alignment. No speaker adaptation is performed for any of the systems. All experiments in this paper are conducted using the Tensorflow neural network training toolkit [AAB16] with the Kaldi decoder [PGB11].

All neural networks are trained using the Adam optimization strategy [KB14] with cross-entropy criterion. The networks are initialized with Gaussian random normal distributed weights with a standard deviation equal to 0.01. The relu activation function is used for all layers. For each layer, before passing the tensors to the nonlinearity function, a batch normalization layer [IS15] is applied to normalize the tensors and speed up the convergence. The shuffling mechanism is applied on each epoch. All neural networks are trained from scratch. For decoding, we use Kaldi WSJ's default setup, which uses trigram language modeling and a large dictionary.

3.4.4.2 Mel-filterbank features vs. Raw waveforms

In this section, we compare the performance of using Mel-filterbank features and raw waveforms. For Mel-filterbank features, both DNNs and CNNs with various numbers of parameters are investigated. The Vocal Tract Length Normalization (VTLN) based data augmentation approach is effective in generating new speech data by transforming spectrograms, using a random linear warping along the frequency dimension. Therefore, we also apply the technique here, to increase the data by 5 times for Mel-filterbank features. For raw waveform features, the proposed 1-D CNN model is used. Table 3.2 compares results of different

Table 3.2: Baseline comparison: different features and neural network structures.

Set-up	**WER**	**#para**
Mel-fbank+5-layer DNN 2048	4.55	$17.6M$
Mel-fbank+5-layer DNN 930	4.78	$3.86M$
Mel-fbank+2-D CNN	4.71	$3.86M$
Mel-fbank+5-layer DNN 2048 +data augmentation	4.02	$17.6M$
raw waveform+1-D CNN	**3.70**	**3.84M**

setups. From the first two rows, we observe that for Mel-filterbank features, decreasing the number of parameters from 17.6M to 3.86M results in performance degradation. Given a similar number of parameters as in rows 2, 3 and 5, 2-D CNNs perform better than DNNs for Mel-filterbank features. When using raw waveforms with the proposed 1-D CNN model, the WER decreases to 3.70, which is 22.6% relatively better than Mel-filterbank with DNNs and 21.4% relatively better than Mel-filterbank with CNNs. The proposed 1-D CNN model without any data augmentation also outperforms a 17.6M-parameter Mel-filterbank-based DNN model with data augmentation by relatively 8% as shown in the 4th row. The results indicate that the proposed 1-D CNN model with raw speech waveforms as input is very efficient for acoustic modeling, and is able to extract useful features automatically. Therefore, we will use the 1-D CNN as our baseline system for the following subsections.

3.4.4.3 Filter sampling

In this section, we first show ASR performance using the proposed CNN models with decreasing number of parameters in Table 3.3. We observe that when the number of parameters decreases from 3.84M to 1.27M (from row 1-4), by reducing the number of filters in the convolutional layers or the hidden nodes in the fully-connected layers, the WERs increase significantly. This result indicates that the number of filters and hidden nodes are very important for this CNN model.

Then, we apply the filter sampling method (denoted as FS) introduced in Section 3.4.3.1 to the baseline CNN model. For convolutional layers, we apply filter sampling along ei-

Table 3.3: Filter sampling results for raw waveform CNNs. 'c^*' indicates that the convolutional layers have half the number of filters in each layer compared with the baseline CNN. 'cw' and 'cd' represent compressing the parameters in the convolution layers using widthwise and depthwise filter sampling, respectively. 'fw' represents performing widthwise filter sampling in the fully connected layers. '/4' means reducing the number of parameters by a factor of 4.

Set-up	**WER**	**#para**
CNN: 7c+f512-512	3.70	3.84M
CNN2: 7c+f512-256	3.88	3.71M
CNN3: 7c^*+f512-512	4.00	1.41M
CNN4: 7c^*+f512-256	4.09	1.27M
CNN+FS (cw/4, fw/4)	4.00	0.96M
CNN+FS (cd/4, fw/4)	4.04	0.96M

ther width (denoted as cw in row 5) or depth (denoted as cd in row 6), respectively. For fully-connected layers, we can only do the sampling along width (denotes as fw). Both convolutional and fully-connected layers are compressed with a ratio of 4 (denoted as cw/4, cd/4 and fw/4), and therefore the total number of parameters is reduced by a factor of 4, which is 0.96M. From Table 3.3, we can see that the performance of filter sampling CNNs (FS-CNNs) degrades compared with the baseline CNNs due to fewer parameters. However, rows 3 and 5 show similar performance even though FS-CNNs have only two thirds of the parameters of a standard CNN. Also note that filter sampling along width or depth has comparable performance to each other (compare the last two rows in Table 3.3).

3.4.4.4 Filter sampling and combination

From the previous section, we noted that only using filter sampling will lead to performance degradation. One of the possible reasons for the degradation is that tying weights between filters can limit their learning ability in extracting features. Therefore, in this section, we show the results when applying both filter sampling and combination in Table 3.4. Rows 4 and 8 show the effect of using both filter sampling and combination to generate CNN filters. Clearly, adding a linear combination step significantly improves the performance of the filter sampling CNNs. When performing widthwise sampling, FSC-CNNs improve

Table 3.4: Filter sampling and combination results for raw waveform CNNs. lin-MxN means doing linear combination using MxN different scalars.

	Set-up	WER	#para
1	**CNN: 7c+f512-512**	3.70	3.84M
2	CNN4: 7c^*+f512-256	4.09	1.27M
3	CNN+FS (cw/4, fw/4)	4.00	0.96M
4	**CNN+FSC (cw/4, fw/4, lin-MxN)**	**3.77**	**1.41M**
5	**CNN+FSC (cw/4, fw/4, lin-MxN/2)**	**3.67**	**1.19M**
6	**CNN+FSC (cw/4, fw/4, lin-MxN/4)**	**3.81**	**1.07M**
7	CNN+FS (cd/4, fw/4)	4.04	0.96M
8	**CNN+FSC (cd/4, fw/4, lin-MxN)**	**3.86**	**1.41M**
9	**CNN+FSC (cd/4, fw/4, lin-M/2xN)**	**3.85**	**1.19M**
10	**CNN+FSC (cd/4, fw/4, lin-M/4xN)**	**3.86**	**1.07M**

the performance of FS-CNNs by 5.7%; when performing depthwise sampling, FSC-CNN outperforms FS-CNN by 4.5%. Hence, the improvement is more significant for widthwise sampling.

As mentioned in Section 3.4.3.2, we can significantly reduce the number of parameters of the linear combination step by simply tying the weights of scalar vectors. In Table 3.4, results for compressing scalar-vector parameters by factors of 2 and 4 (denoted as /2 and /4) are shown in rows 5, 6, 9 and 10. We notice that by using less parameters for a linear combination, the FSC-CNN can achieve even better performance due to less over-fitting. When performing widthwise filter sampling and compressing the linear combination weights by a factor of 2, we achieve a WER of 3.67 with only 1.19M parameters, which is even better than the strong baseline CNN with x3.2 more parameters. When this best performing system is compared with the CNN4 model, in the 2nd row, with a similar number of parameters, the WER decreases by 10.26%. When further reducing the number of parameters to 1.07M as in row 6, the WER increases to 3.81, which is only a small degradation.

3.4.4.5 Discussion

Theoretically, the proposed FSC-CNN can be easily generalized to 2D CNN with time-frequency features. However, we believe FSC-CNN is more effective with 1-D CNN and

raw-audio input, since the filter size for 1-D CNN is usually quite large in order to capture lager reception fields of the raw waveforms, which may result in larger redundancy between the learned filters. Moreover, the proposed FSC-CNN can be combined with other model compression techniques, such as weight quantization, to further reduce the number of parameters significantly with no loss in accuracy.

3.5 Conclusion

In this chapter, we explored novel neural network architectures, which can take complex DFT features or raw waveforms as input and perform feature extraction and phone classification jointly. In the first part of this chapter, a unified deep Highway (HW) network with a time-delayed bottleneck layer (TDB) in the middle was proposed. The TDB-HW networks learn a bottleneck feature automatically from complex DFT features, and can be trained from scratch. The TDB-HW networks with complex DFT features as input provide significant lower error rates compared with LFBE (generated from perception-based Mel filterbanks) on a large-scale wake-word detection task. In the second part of this chapter, we presented a 1-D Convolutional Neural Network (CNN) model, which takes raw waveforms as input and uses convolutional and pooling layers to do hierarchical feature extraction. The proposed 1-D CNN model outperforms standard systems with Mel-filter bank features on WSJ LVCSR task. In order to reduce the filter redundancy of the 1-D CNN model, we proposed a filter sampling and combination (FSC) technique, which can naturally enforce parameter sharing, but also adds to the learning capacity of filters. FSC can reduce the model size by 70% and still improve the performance on ASR tasks.

CHAPTER 4

Sequence modeling for acoustic and language models

4.1 Introduction

Sequence modeling is one of the most important speech-related tasks. Both acoustic and language modeling can be treated as sequence modeling problems. Acoustic modeling models a time series of acoustic features, and language modeling models a sequence of text units. Neural-network based architectures, such as Recurrent Neural Networks (RNNs), have been widely used for sequence modeling. In this chapter, we present several novel neural-network architectures for sequence modeling.

In the first part of this chapter, we introduce a Convolutional, Long Short-Term Memory (LSTM), Deep Neural Network (CLDNN) for acoustic modeling on an acoustic scene classification task. Based on the CLDNN architecture, an novel attention-based mechanism is proposed and applied to the LSTM layer. The attention mechanisms are motivated by human behavior, and can automatically predict the importance of each LSTM time step and select the most important information from sequences. This work was published in [GXL17].

In the second part of this chapter, we present a sequence-to-sequence based error correction model for end-to-end ASR. End-to-end ASR models require audio-text pairs during training, which uses far less text data compared to the language model component of a conventional recognizer and thus can make many errors on rare/unseen words. Therefore, we propose an error correction model trained on text-to-text pairs (errors and groundtruth) generated from text-only data, which can effectively correct errors made by the system. This work was published in [GSW19]

4.2 Related work

For acoustic scene and event recognition, a variety of deep architectures have been proposed. In [MZX15], the authors apply a fully connected DNNs to this task, which is initialized using unsupervised training with deep belief neural networks (DBNs). Recently, both CNNs and recurrent neural networks (RNNs) (especially LSTMs) have shown improvements over DNNs. Various deep CNN architectures with multiple convolutional and pooling layers are employed for hierarchical feature extraction from audio signals [ZMS15, Pic15, HCE17], and CNNs also show robustness for audio event detection [PHM16]. In [PHV16], the authors propose multi-label RNNs in the form of bidirectional LSTMs for polyphonic audio event detection, which outperform DNN-based methods by a large margin. Besides neural network based systems (NNs), i-vector based systems also show effectiveness for long-duration ASC, and can provide complementary information to NNs [ELD16].

In terms of incorporating text-only data into an end-to-end ASR framework, there have been several research studies that look at incorporating an RNN-LM trained on text-only data into the end-to-end model. These approaches include rescoring the n-best decoded hypotheses from the end-to-end model [CJL16], or incorporating an RNN-LM into the first-pass beam search [BCS16, CJ17, KWN18], via fusion. While such RNN-LM fusion techniques fix some tail words cases, we have found that numerous rare word and proper noun errors still exist

4.3 Learning attention mechanism for acoustic modeling

4.3.1 Acoustic scene classification

Acoustic scene classification (ASC) aims to recognize environmental sounds, which is very useful for applications like multimedia content retrieval [XXD08] and audio and video classification and segmentation [ZK01]. Motivated by the success of deep neural networks, a variety of deep architectures have been recently proposed to model acoustic sequences. In this section, we apply CLDNNs with bidirectional LSTMs to ASC in a unified architecture.

CLDNNs (first introduced in [SVS15]) can take advantage of CNNs for frequency variation reduction, LSTMs for sequence modeling, and DNNs for discriminative training, which we believe are all suitable for ASC. Moreover, since the informative part of audio signals might not span the entire duration of the audio segments, we further propose a novel attention mechanism in the LSTM layer, to predict the importance of each LSTM time step. The weighted LSTM output using attention scores is propagated to the next layer. We would like to find out whether and how the attention framework can help ASC and what the attention model can learn in the end-to-end training framework.

4.3.2 Neural network architectures

In this section, we fisrt introduce two neural network architectures (CNNs and CLDNNs) for ASC tasks.

4.3.2.1 CNNs

The CNN model used in this section is an Alexnet-like [KSH12] structure, which comprises 3 stacked pairs of convolution and max-pooling layers, and one fully connected layer with a softmax layer on top. The first convolutional layer uses 16 filters of size 5*5, the second and third layers use 32 and 64 filters of size 3*3 respectively. All the strides for the convolutional layer are set to 1. The kernel size 2*2 and stride 2 are used for all pooling layers. After the third pooling layer, the output is flattened and passed to the fully-connected layer with 256 nodes.

4.3.2.2 CLDNNs

In order to have a fair comparison with CNNs, the CLDNN architecture proposed in this section uses the same configuration as the CNN model described above, except that the third convolution and max-pooling layers are replaced by an LSTM layer. A diagram of the proposed CLDNN architecture is shown in Figure 4.1. For the LSTM layer, we propose 3 different layers, which are forward layer (denoted as FWLSTM), backward layer (denoted as

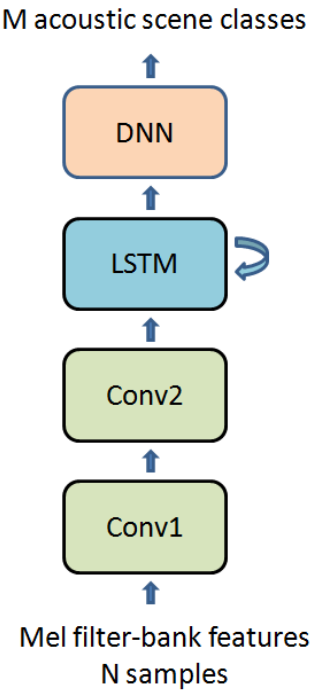


Figure 4.1: The CLDNN framework

BWLSTM), and bidirectional layer (denoted as BLSTM). In the forward layer, each hidden layer connects to the following time period, while the backward layer's hidden layer connects to the previous time period. The bidirectional layer combines both backward and forward layers, propagating information not only from the past but also from the future.

The output from the last convolution layer is reshaped to a sequence of vectors before being fed to the LSTM layer, and each vector represents the feature extracted for the corresponding time step. For FWLSTM and BWLSTM, 256 hidden nodes are used and the output of the final time step is passed to the fully connected layer. BLSTM concatenates the outputs of the final steps for both forward and backward directions and passes it to the next layer, as shown in Figure 4.2 (left).

4.3.3 Attention mechanisms for sequence modeling

For the proposed CLDNN model, the LSTM layer only passes the output of the final time step to the fully connected layer for classification, which summarizes all the previous time

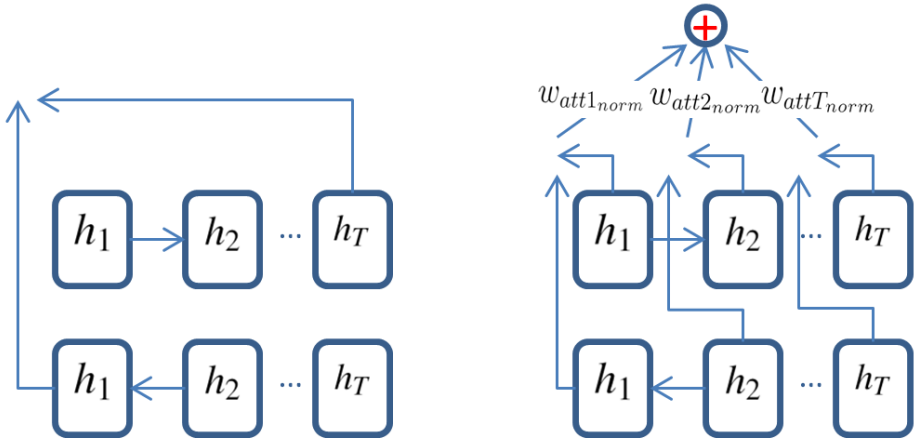


Figure 4.2: Standard BLSTM layer (left), attention-based BLSTM layer (right)

steps' information. However, humans usually discriminate acoustic scenes by specific events which correspond to certain important time steps. For example, when we need to tell whether a scene is in a restaurant or not, the impact sound between dishes and people chatting play an important role; when we recognize a scene in a park, we may focus more on bird sounds. Therefore, in this paper, we introduce a novel attention mechanism, which can automatically predict the importance of each time step and improve acoustic scene classification.

4.3.3.1 Mathematical representation of attention mechanism

Let h_t denote the hidden state of each LSTM time step with length T and we design a mapping function $f(.)$, which uses the hidden states to predict an attention score w_{att} for each time step; the final output O_{att} is the normalized weighted sum of all the hidden states as shown in Figure 4.2 (right). Eqs. 4.1-4.3 show the mathematical implementation of the attention mechanism. The softmax function is used to normalize the score, and it also presents a probabilistic interpretation of the attention scores.

$$w_{att} = f(h_t) \tag{4.1}$$

$$w_{att_{norm}} = softmax(w_{att}) \tag{4.2}$$

$$O_{att} = \sum_{t=1}^{T} h_t * w_{att_{norm}}(t) \tag{4.3}$$

The key of this attention mechanism is to train a proper mapping function $f(.)$, such that we can get reasonable attention scores from the hidden states. In this paper, we investigate two mapping strategies with a specific mapping function. For mapping strategies, firstly, we use each hidden state to predict its own weight, which is a one-to-one mapping; secondly, all the hidden states are used together to predict all the weights for each time step, which is an all-to-all mapping. For the choice of mapping function, a shallow neural network with a single fully connected layer and a linear output layer is adopted.

Note that, in the BLSTM condition, the hidden states of the same time step from both forward and backward directions, need to be concatenated to represent h_t.

4.3.3.2 Combination of the attention model and standard LSTM models

The LSTM model uses the output from the final time step as a summary information of the whole sequence, which has a long memory for previous time steps. The attention model tries to find the most important time steps in the sequence. Both the LSTM and attention models have advantages for specific scenes and they get information from different views of the sequence. Therefore, we also want to investigate the combination of these two models.

We propose three-stage ensemble methods, which are early stage, middle stage, and late stage fusion. For early stage combination, we concatenate the LSTM output of the final time step with the attention output, then the combined output is passed to the fully connected layer. For the middle stage combination, the outputs of the fully connected layer from both models are concatenated, and then the combined output is passed to the softmax layer. For late stage fusion, the output of the softmax layer from both models are linearly combined to

make a final decision and the combination weights are jointly trained with neural networks. For the combined model training, the weights and biases before the concatenation layer are initialized with the pre-trained LSTM and attention models.

4.3.4 Evaluation set-up

In this section, we are interested in short-duration ASC (e.g around 6s), since in many real applications, like video and audio classification using data from social media networks, only short-duration audio segments are available.

4.3.4.1 Dataset and evaluation protocol

To evaluate the performance of the proposed methods, we use the TUT acoustic scenes classification 2016 dataset (DCASE) which consists of recordings from 15 different acoustic scenes [MHV16]. There are 78 audio segments for each scene, which are 30s-long each and recorded with a 44.1kHz sampling rate. The database provided a 4-fold cross validation setting to test the generalization of the algorithm, which guarantees that all audio files recorded in the same location are on the same side of evaluation (i.e., training or testing). For each fold, around 880 segments are used for training and 290 are used for testing, and classes are evenly distributed in both the training and testing data. Since our goal is to improve scene classification accuracy using short-duration segments and 6s is a common length for short video and audio files, we truncate each of the 30s-segment into 6s continuous audio files. In order to get more training data, we apply small shifts to the recordings. In the end, in each fold we have around 1 million 6s-segments for training and 340k 6s-segments for testing.

4.3.4.2 Audio processing and feature extraction

40 log mel-filterbank coefficients are extracted at 20ms intervals using a 40 ms Hamming window. All features are normalized to zero mean and unit variance. Therefore for each 6s segment, we have 300 feature vectors and each of them is of 40 dimensions. We combine

Table 4.1: Classification accuracy (%) of CNNs and CLDNNs

Neural Networks Architectures	Accuracy
CNNs	73.95
CLDNNs (FWLSTM)	73.86
CLDNNs (BWLSTM)	72.48
CLDNNs (BLSTM)	**74.48**

the 300 vectors into a 40*300 2-D feature map, which represents the mel-filterbank features distributed along both frequency (using filterbank index) and time (using the frame number). The 2-D feature map is the input we use for neural network training.

4.3.4.3 Neural network training

The proposed CNNs, CLDNNs, and attention models are evaluated and compared. All neural networks are trained using the Adam optimization strategy [KB14] with cross-entropy criterion and a scheduled learning rate starting from 0.005. The networks are initialized with Gaussian random normal distributed weights with *std* equaling 0.05. The sigmoid activation function is used for all layers. The shuffling mechanism is applied on each epoch. CNN and CLDNN models are trained from scratch. The attention model is initialized using the pre-trained CLDNN parameters of the first 3 layers (2 conv and 1 LSTM), and only the attention, fully-connected and softmax layers are trained. The shallow attention neural network is jointly trained with the whole network structures. For the combined models, the weights of the pre-trained CLDNNs and attention models are used to initialize the layers before the concatenation layer, and only the layers after the concatenation layer are trained. The Tensorflow toolkit is used here for neural network training [AAB16].

4.3.5 Experimental results

4.3.5.1 Comparison of CNNs and CLDNNs

First, we establish a comparison of the CNNs and the proposed CLDNN model. For CLDNNs, we compare FWLSTM, BWLSTM, and BLSTM layers. Table 4.1 shows the

Table 4.2: Classification accuracy (%) of CBLDNNs and different attention models

Neural Networks Architectures	Accuracy
CBLDNNs	74.48
CBLDNNs, att_{fc}, att_{one}	73.31
CBLDNNs, att_{fc}, att_{all}	**74.90**

results of the four different neural network structures. From the results we can see that CNNs and CLDNNs with the FWLSTM layer have similar performance, which is much better than CLDNNs with the BWLSTM layer. This may indicate that when modeling the audio sequence for an acoustic scene using LSTMs, the direction of the sequence is important. Moreover, the convolutional layers are reasonably good for frequency and time feature extraction and modeling. The combination of the final outputs of the forward and backward LSTMs, which is the BLSTM case, gives performance improvement compared with CNNs. The combined information from both directions give complementary and more complete information about the audio sequence. From now on, we use CLDNNs with the BLSTM layer (denoted as CBLDNNs) as our new strong baseline to investigate the attention mechanism.

4.3.5.2 Comparison of CBLDNNs and attention model

In this section, we apply the attention mechanism on the BLSTM layer. Note that, for CBLDNNs, the BLSTM layer concatenates the final outputs from both forward and backward directions. However, as mentioned in Section 4.3.3.1, the hidden states $h(t)$ of the BLSTM layer used for the attention mechanism, is the concatenation of the hidden states from both directions for the same time step. Therefore, $h(t)$ will have information passed from both directions and also show more information of the current time step.

We use a shallow fully connected neural network with one hidden layer (denoted at att_{fc}) to represent the mapping function $f(.)$. There are 1024 units for the hidden layer. We denote the one-to-one mapping between hidden states and attention weights as att_{one}, and the all-to-all mapping as att_{all}. The results can be seen in Table 4.2. The one-to-one mapping gives worse performance compared with standard CBLDNNs, which indicates that it's difficult

Table 4.3: Classification accuracy (%) of CBLDNNs, attention model and 3 combined models

Neural Networks Architectures	Accuracy
CBLDNNs	74.48
CBLDNNs, att_{fc}, att_{all}	74.90
$combination_{early\ stage}$	**76.19**
$combination_{mid\ stage}$	75.33
$combination_{late\ fusion}$	75.52

to learn the mapping using only local information due to large variations. As expected the att-to-all mapping gives improvement compared with the strong baseline, and it shows that using global information to predict the attention scores is feasible.

Moreover, for each class, the attention model and standard CBLDNN model have different performances. Pilot experiments show that for some scene classes, like restaurants, the attention model is more useful since certain time steps are more important than others; while for some other scenes, it is better to use the overall information of the entire time sequence to make decisions. Therefore, it is natural to expect that by combining the attention-based information with the LSTM final summarization information, we should get better performance due to the complementary nature of the two models.

4.3.5.3 Comparison of different combination methods for standard CBLDNNs and attention models

In this section, we will combine the attention model with the CBLDNNs. We denote the three combination methods described in Section 4.3.3.2 as $combination_{early\ stage}$, $combination_{mid\ stage}$ and $combination_{late\ fusion}$ respectively. The performances of the different combined models are summarized in Table 4.3. The results show significant improvement using the combined models compared with the standard CBLDNNs and attention model, which proves the complementary information provided by the two models. Moreover, the early stage concatenation of the BLSTM and attention outputs gives the best performance compared with the combination of the outputs from the fully connected layer and the score level fusion. The reason may be that by combining the two models in the early stage, the joint fully connected

layer and softmax layer can better transform the combined features into a space that makes the output easier to classify.

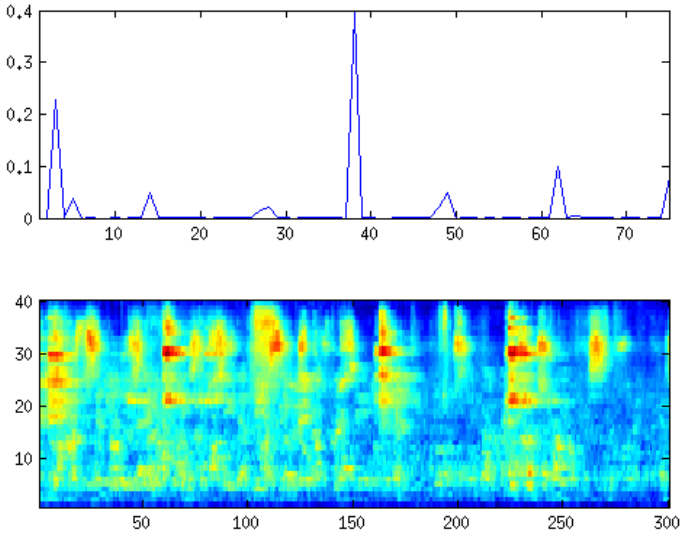


Figure 4.3: The mel-filterbank features (bottom) with time-aligned attention scores (top) for the sample segment recorded in a cafe/restaurant

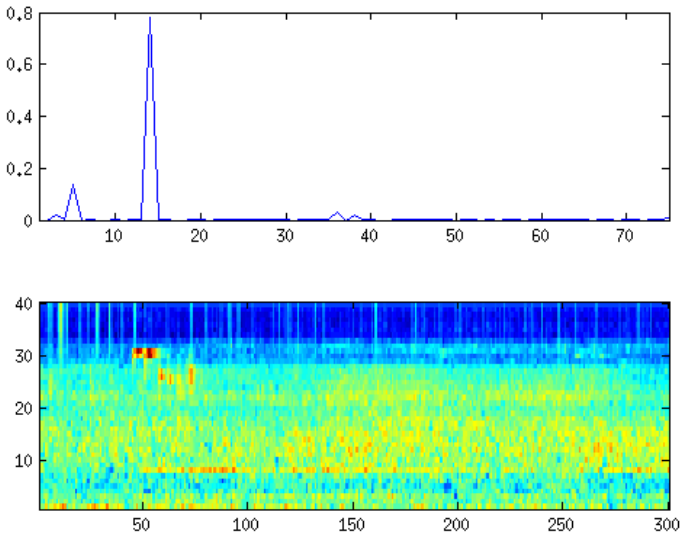


Figure 4.4: The mel-filterbank features (bottom) with time-aligned attention scores (top) for the sample segment recorded in a park

4.3.6 Analysis of learned attention weights

It is interesting to investigate the attention scores predicted by the proposed attention model under the CBLDNN architecture. We select several 6s audio segments from the test dataset, which are recorded in a cafe/restaurant and a park. We show the 2D feature maps with the time aligned attention scores for each segment in Figures 4.3 and 4.4.

Figure 4.3 shows an audio segment recorded in cafe/restaurant condition. The bottom figure is the mel-filterbank features along time stamps, and the upper figure is the predicted attention scores from the attention model. Since we have 75 attention scores corresponding to each LSTM hidden state, we stretch the upper figure to align with the 300 frames of the mel-filterbank features in the bottom. We can see from the figures that there are 2 significant high scores at time stamps around #10 and #150 frames. Based on what we listen to in the audio file and observe from the feature map, there are clear impact sounds of dishes around those two time stamps. It appears that the attention model is trained to pay more attention to the impact sounds for the cafe/restaurant scene. Moreover, we can see that the attention model only gives higher scores when acoustic events occur, like when people are talking.

We then show an audio segment recorded in a park in Figure 4.4. Despite the relative strong constant noise in this segment, a significant high score is predicted around frame #55, where a clear bird sound can be observed.

Another interesting phenomenon we observed is that, when we compare attention scores generated by attention-based CLDNNs with a forward LSTM layer and a bidirectional LSTM layer, the scores sometimes tend to be higher for the last several time steps for forward LSTM condition and more balanced for the bidirectional LSTM condition. This may be because each concatenated hidden state for BLSTM contains summaries for both previous and future information, which makes each time stamp more balanced and helps in predicting better attention weights.

4.4 Learning a spelling correction model for end-to-end speech recognition

The previous section focused on neural network models for acoustic sequences. In this section, we present a sequence-to-sequence based spelling error correction model using text sequences for end-to-end speech recognition.

End-to-end models for automatic speech recognition (ASR) have gained popularity in recent years as a way to fold separate components of a conventional ASR system (i.e., acoustic, pronunciation and language models) into a single neural network [CJL16, BCS16]. The Listen, Attend and Spell (LAS) model [CJL16], is one such model which has shown competitive performance on a Voice search task compared to a strong conventional baseline model [CSW18].

4.4.1 Motivation

End-to-end models require audio-text pairs during training. They are therefore trained using far less data compared to the language model (LM) component of a conventional recognizer, which is often trained with an order of magnitude more text-only data. Due this reduced training data, end-to-end models do not perform as well on utterances containing rare words which occur infrequently in the audio-text training set.

To address this issue, there have been several studies that consider incorporating an RNN-LM trained on text-only data into the end-to-end framework. While such RNN-LM fusion techniques fix some tail words cases, our pilot experiments showed that numerous rare word and proper noun errors still exist. One hypothesis is that the LM is not integrated into the end-to-end model with the objective of correcting errors that the end-to-end model makes.

In this section, we incorporate a module trained on text-only data into the end-to-end framework, with the objective of correcting errors made by the system. Specifically, we investigate using unpaired text-only data to synthetically generate corresponding audio sig-

nals using a text-to-speech (TTS) system, a process similar to backtranslation in machine translation [SHB16]. We then run the baseline LAS speech recognizer on the TTS output to create a set of text-to-text pairs representing an error hypothesis and its corresponding ground truth. We train a spelling corrector (SC) model on these text-to-text pairs, to correct potential errors made by the first-pass recognizer. We compare our proposed approach to two other approaches of incorporating text-only data which do not account for the error distribution of the LAS model: rescoring an n-best list with an RNN-LM, and directly training the LAS model on TTS-synthesized speech.

4.4.2 Baseline LAS model

The baseline speech recognition model is an LAS-inspired encoder-decoder architecture with attention based on [ZCJ17]. The encoder consists of a stack of covolutional and LSTM layers, which takes as input mel spectrogram features, $\mathbf{x}$, and maps them to a higher order feature representation $\mathbf{h}^{enc}$. The encoder output is passed onto an attention mechanism, which aligns the input audio sequence with the output sequence representing the transcript, determining which encoder frames should be used to predict the next output symbol, y_i. The output of the attention is a context vector $\mathbf{c_i}$ that is passed to the decoder. Finally, the decoder takes the attention context $\mathbf{c_i}$ as well as an embedding of the previous prediction, y_{i-1}, and generates logits $\mathbf{h}^{dec}$. These logits are passed through a softmax to compute a probability distribution $P(y_i|\mathbf{h}^{dec})$ across output tokens. We can think of the decoder as being similar to a language model. The model is trained to minimize the cross-entropy loss on the training data. In our work, the decoder outputs a sequence of wordpiece units y_i, which has shown good performance for both ASR and machine translation (MT) tasks [CSW18, WSC16, ZIS18].

4.4.3 Approaches of utilizing text-only data

4.4.3.1 External LM

A common approach for incorporating text-only data is to train an RNN-LM on text-data [CJL16, BCS16, CJ17, KWN18], and then incorporating the LM during beam search decoding through various mechanisms proposed in the literature. In this section we focus on n-best rescoring similar to [CJL16]. LM rescoring makes it easier to evaluate the spelling corrector which we will introduce in Section 4.4.4.3, and thus for a fair comparison we did the same for the baseline LAS model. Specifically, during inference our objective is to find the most likely sub-word unit sequence given the score from the LAS model $P(\mathbf{y}|\mathbf{x})$ and the LM $P_{LM}(\mathbf{y})$:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} \log P(\mathbf{y}|\mathbf{x}) + \lambda \log P_{LM}(\mathbf{y}) \tag{4.4}$$

where λ is an interpolation weight determined on a held-out set.

4.4.3.2 Training on synthesized speech

Another approach is to use text-only data to synthesize audio-text training data using a text-to-speech (TTS) system. An analogous approach has been explored in machine translation, where Sennrich [SHB16] passed unparied texts in the target language into a pretrained "backtranslation" model in order to generate corresponding text in the source language. This synthetic data were then used to augment the existing parallel training data to train the translation model. In this work we synthesize the text-only data using a high quality TTS system based on parallel WaveNet [OLB18] and use the resulting synthetic audio when training the LAS model. A similar approach was recently applied to speech recognition in [HWZ18].

4.4.4 Spelling correction model

A disadvantage of the previously described approaches for incorporating text-only training data is that they do not take into account the characteristic error distribution made by the

speech recognizer. In this section we propose a novel approach to utilizing text-only data, by training a supervised "spelling correction" model to explicitly correct the errors made by the LAS recognizer. Intuitively, this task is simpler than unsupervised language modeling because it is able to take advantage of the existing language modeling capacity of the LAS model. Instead of predicting the likelihood of emitting a word based on the surrounding context as in an RNN-LM, the SC model needs only to identify likely errors in the LAS output and propose alternatives. Since the baseline LAS model already has a relatively low word error rate, most of the time this task reduces to simply copying the input transcript directly to the output.

4.4.4.1 Training data

Training an SC model requires a parallel text training set consisting of LAS hypotheses to be corrected and ground truth text sequences. In order to generate a training corpus from text-only data which is representative of the baseline LAS model's error distribution we generate TTS utterances $\{u_1, u_2, ...\}$ using text sequences $\{y_1, y_2, ...\}$ from the text corpus. Next we perform decoding on the TTS data using the pretrained LAS model. Each TTS utterance u_i can be paired with N hypotheses $\{H_{i1}, H_{i2}, ..., H_{iN}\}$ after beam-search decoding. By using all hypotheses from the n-best list to generate training data, we create a diverse training set that captures more variance of the LAS model's underlying error distribution. During SC training we randomly sample a hypothesis H_{ij} from the LAS n-best list and combine it with the ground-truth transcript y_i to form a training pair.

4.4.4.2 Architecture and training

We use an attention-based encoder-decoder sequence-to-sequence architecture for our spelling corrector, similar to the neural machine translation model from [CFB18]. The input and output sentences are first decoded as wordpieces [SN12]. The encoder takes the embedding learned from the input sequence H_{ij} and maps it to a higher-level representation through a stack of bi-directional LSTM layers. The decoder also consists of stacked unidirectional

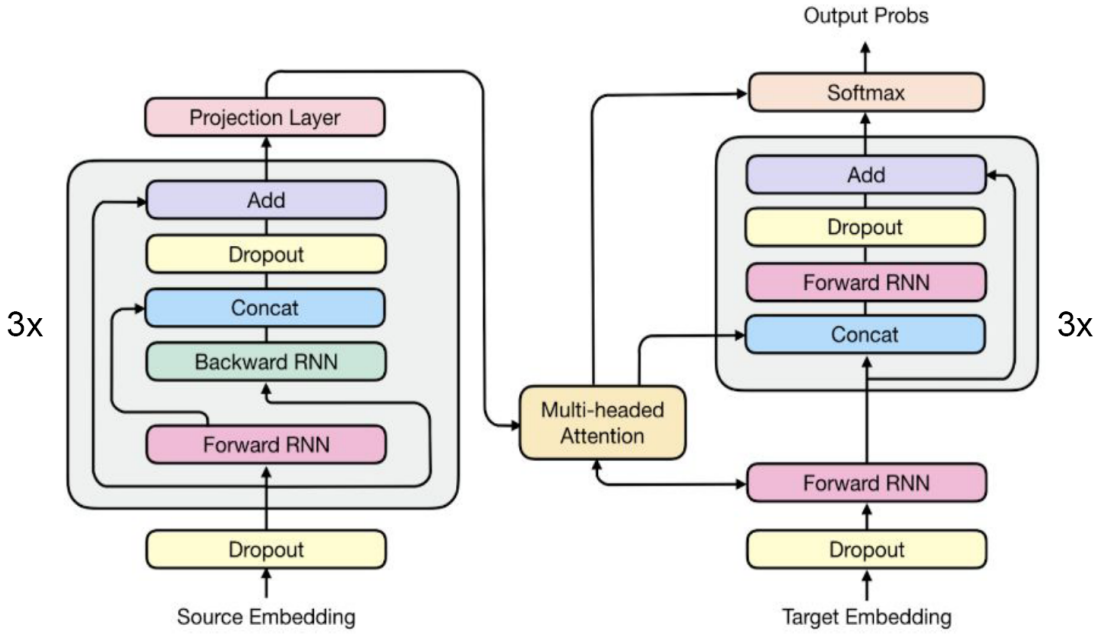


Figure 4.5: Spelling Correction model architecture.

LSTM layers that uses an attention mechanism to attend to the encoder representation and generate the output sequence y_i one token at a time.

Figure 4.5 shows the model architecture. Compared with the standard attention-based sequence-to-sequence model, there are several differences. First, residual connections are added between layers in both the encoder and decoder. Per-gate layer normalization is applied within each LSTM cell, and multi-head additive attention is used. The bottom decoder layer and the final encoder layer output are used to obtain the recurrent attention context, which is fed to all decoder LSTM layers and also the softmax layer by concatenation. The model is trained using the standard maximum-likelihood criterion, to maximize the sum of log probabilities of the ground-truth outputs given the corresponding inputs.

4.4.4.3 Inference

During inference, decoding the LAS model with beam search takes an utterance u and produces N hypotheses $\{H_1, H_2, \ldots, H_N\}$ with corresponding log probability scores $\{p_1, p_2, \ldots, p_N\}$. For hypothesis H_i, the SC model can similarly be used to generate M new hypotheses $\{A_{i1}, A_{i2}, \ldots, A_{iM}\}$ with corresponding log probability scores $\{q_{i1}, q_{i2}, \ldots, q_{iM}\}$. Therefore,

for a given utterance u, the cascaded LAS and SC models can generate a total of $N \times M$ hypotheses with associated scores. Rescoring all $N \times M$ candidates with an LM gives a set of LM scores $\{r_{11}, r_{12}, \ldots, r_{MN}\}$. Finally, we can find the most likely hypothesis using the following criteria:

$$\mathbf{A}^* = \underset{\mathbf{A}}{\operatorname{argmax}} \lambda_{LAS} * p_i + \lambda_{SC} * q_{ij} + \lambda_{LM} * r_{ij} \tag{4.5}$$

where λ_{LAS}, λ_{SC} and λ_{LM} are the weights for LAS, SC, and LM scores respectively, which are determined on a held-out set. Note that independently correcting each of the N LAS hypotheses with the SC model increases the total computational cost of the system by a factor of N. In the following experiment section we compare configurations where the SC model is used to correct only the top LAS hypothesis, i.e. using $N = 1$, to the full $N \times M$ configuration.

4.4.5 Experimental setup

We conduct our experiments on LibriSpeech [PCP15]. The training data contains around 960 hours of speech from read audio book recordings. We evaluate on the "clean" dev and test sets, which each contains around 5.4 hours of speech. For feature extraction, 80 dimensional log-mel filterbank features are computed from 25ms windows shifted by 10ms, stacked with their deltas and accelerations.

As an external text-only training dataset, we use the 800M word LibriSpeech language modeling corpus which was carefully selected to avoid overlap with the text in the dev and test sets. We preproccess the corpus by filtering out 0.5M sequences which contain only single letter words or are longer than 90 words. We use the remaining 40M text sequences to train the language model, generate a TTS dataset to train the LAS model, and generate error hypotheses to train the SC model.

4.4.5.1 Speech recognition model

The baseline LAS recognition model uses 2 convolutional layers and 3 bidirectional LSTM layers in the encoder, and a single undirectional LSTM layer in the decoder. A multi-

headed additive attention mechanism with 4 heads is used. We use a wordpiece model with 16K tokens, which is generated using the byte pair encoding algorithm. The wordpieces are represented with 96 dimensional embeddings which are learned jointly with the rest of the model. For regularization, label smoothing [CJ17] is applied by weighing the ground truth token at each output step by 0.9, and uniformly distributing the remaining probability mass among other tokens. During inference we use beam search decoding with a beam size of 8. The baseline LAS model is trained using asynchronous stochastic gradient descent optimization with 16 workers. All models are implemented in Tensorflow [AAB16] and trained using the Adam optimizer [KB14].

4.4.5.2 Language model

We train a stacked RNN with two unidirectional LSTM layers to use as an external language model. The same 16K wordpiece token set is used as the LAS model. Early stopping during training is based on the dev set perplexity. Similar to [CJL16], we use the LM to rescore the n-best list generated by decoding the LAS model with beam search. The interpolation weight λ is swept on a held-out dev set.

4.4.5.3 TTS model

In order to synthesize speech, we use a Parallel WaveNet model [OLB18], which can generate high fidelity speech very efficiently, trained on 65-hours of speech from a single female speaker. We use this model to perform inference on the full text-only dataset, and generate 40M audio utterances. We train a combined real+TTS LAS recognizer by combining these synthetic speech utterances with the 960 hour LibriSpeech training set. During training, each batch is comprised of 70% real speech and 30% synthetic speech.

4.4.5.4 Spelling correction model

The proposed SC model uses 3 bidirectional LSTM layers in the encoder and 3 unidirectional LSTM layers in the decoder. Residual connections are added to the third layer of both the

encoder and decoder. Four-headed additive attention is used. The same 16K wordpiece model is adopted as in the LAS and language models. Beam search decoding is performed with a beam size of 8. For regularization, a dropout rate of 0.2 is applied to both embedding layers and each LSTM layer output. Attention dropout is also applied with the same rate. Uniform label smoothing with uncertainty 0.1 is applied, and parameters are L2 regularized with weight 10^{-5}.

The learning rate schedule includes an initial linear warm-up phase, a constant phase, and a exponential decay phase following [CFB18]. SC model is trained with synchronous training using 32 GPUs and adaptive gradient clipping is used to further stabilize training.

To generate training data for this model, we first decode the 40M clean TTS utterances using the baseline LAS model. Each TTS utterance results in 8 hypotheses, and each hypothesis is grouped with the corresponding ground-truth transcript to form an SC training pair. This process results in a total of about 320M training pairs.

We additionally experiment with a multi-style training (MTR) configuration using an augmented dataset by corrupting the synthesized utterances with additive noise and reverberation using a room simulator. Noise signals are collected from YouTube and daily life noisy environmental recordings, and randomly reverberated and mixed with the speech such that the overall SNR is between 20dB and 40dB [KMC17]. We run the full synthesized speech set through this process, resulting in a total of 40M noisy utterances. These utterances are decoded by the baseline LAS model following the same procedure described above to generate the noisy SC training set. The union of clean and noisy train sets yields a total of 640M MTR pairs.

System	Dev-clean	Test-clean
LAS	5.80	6.03
LAS LM (8)	4.56	4.72
LAS-TTS	5.68	5.85
LAS-TTS LM (8)	**4.45**	**4.52**
LAS SC (1)	**5.04**	**5.08**

Table 4.4: Word error rates (WERs) on LibriSpeech "clean" sets comparing different techniques for incorporating text-only training data. Numbers in parentheses indicate the number of input hypotheses considered by the corresponding model.

4.4.6 Experimental results

In this section, we compare different methods of incorporating text-only data.

4.4.6.1 Baseline methods

Baseline results found using the LAS model with and without rescoring the n-best list using an external LM are shown in the top of Table 4.4 (top two rows). The optimal LM weight of $\lambda = 0.5$ was selected by tuning on the development set. LM rescoring leads to a relative improvement of 21.7% over the LAS baseline.

Next we present results by augmenting the LAS training set using speech synthesized from the text-only training data. As mentioned in Section 4.4.5.3, in order to avoid overfitting we train the LAS model using a combination of real and TTS audio, referred to as LAS-TTS. Table 4.4 shows that this improves performance slightly, but the gains are not as large as LM rescoring the baseline. However, the combination of TTS-augmented training and LM rescoring improves over LM rescoring along, demonstrating the complementarity of the two methods.

4.4.6.2 Correcting the top hypothesis using SC model

First, we calculate performance of the SC method when only considering the top hypothesis output by the recognizer. As shown in row 5 of Table 4.4, correcting the top hypothesis gives a 15.8% relative improvement over the baseline.

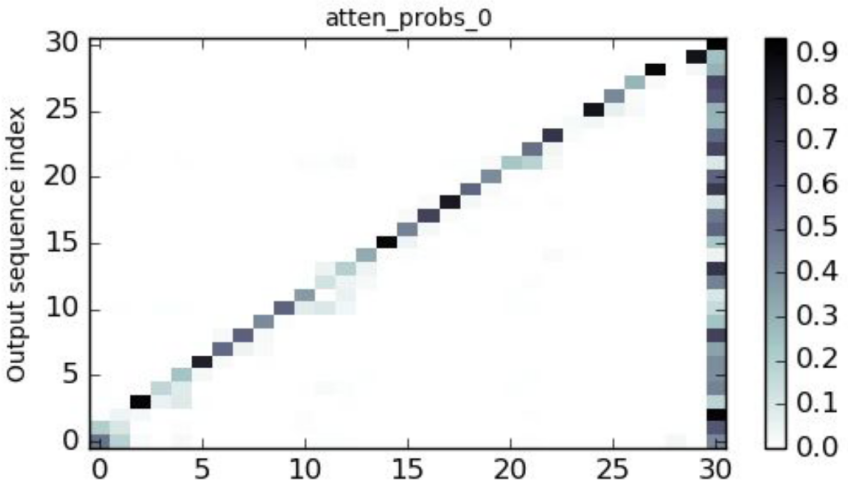


Figure 4.6: Example attention weights from the SC model.

Figure 4.6 shows example attention weights using in the SC model. We find that the attention weights are generally monotonic, and where errors occur, the attention weights are aligned to the adjacent context, helping the model to choose a more suitable output. This behavior can be seen between output tokens 10 to 15 in Figure 4.6.

4.4.6.3 Generating richer n-best lists using SC model

Table 4.5 compares oracle WERs of the LAS model with and without spelling correction. Two configurations of the SC model are considered: correcting the top hypothesis emitted by the LAS model, leading to a final list of 8 candidates, and correcting all 8 entries in the LAS n-best list, leading to an expanded list of 64 candidates. When correcting only the top LAS hypothesis, the SC model gives only a small improvement in oracle WER. However, when applying the SC model independently to each entry in the full LAS n-best list, the oracle WER is significantly reduced to almost half. This demonstrates that the SC model is able to generate a richer and more realistic list of hypotheses, which is more likely to include the correct transcript.

This motivates the use of LM rescoring on the expanded n-best list as introduced in Section 4.4.4.3. The optimal weights found by tuning on the dev set are: $\lambda_{LAS} = 0.7$,

System	Dev-clean	Test-clean
LAS	3.11	3.28
LAS SC (1)	3.01	3.02
LAS SC (8)	**1.63**	**1.68**

Table 4.5: Oracle WER before and after applying the SC model.

System	Dev-clean	Test-clean
LAS	5.80	6.03
LAS LM (8)	4.56	4.72
LAS-TTS	5.68	5.85
LAS-TTS LM (8)	**4.45**	**4.52**
LAS SC (1)	5.04	5.08
LAS SC (8) LM (64)	**4.20**	**4.33**

Table 4.6: Word error rates (WERs) on LibriSpeech "clean" sets comparing different techniques for incorporating text-only training data. Numbers in parentheses indicate the number of input hypotheses considered by the corresponding model.

$\lambda_{SC} = 1.0$, and $\lambda_{LM} = 0.1$. We use the same parameters on the test set. As shown in row 6 of Table 4.6, this leads to a significant performance improvement compared to using the SC model alone, corresponding to more than a 28% relative improvement over the baseline LAS model alone. This promising result shows that the probability scores emitted by each of the three models are complementary to each other.

4.4.6.4 Train the SC model on more realistic TTS dataset

Using TTS data to synthesize errors the LAS model makes is not perfect as there is a mismatch between the TTS data and real audio. Thus, when we compare the decoded errors generated from TTS data and real-audio data, as shown in Table 4.7, there is a big mismatch between them. Specifically, the table shows the performance of the LAS model on LibriSpeech dev set and a TTS dev set generated using the same transcripts. The SC model which is applied on a TTS test set performs clearly better than the real audio test set, even after LM rescoring.

Therefore, to address this audio mismatch issue, we added noise to the TTS data to make

System	Dev-clean	Dev-TTS
LAS baseline	5.80	5.26
LAS SC (1)	5.04	**3.45**
LAS SC (8) LM (64)	4.20	**3.11**

Table 4.7: WER comparison on a real audio and TTS dev sets.

System	Dev-clean	Test-clean
LAS	5.80	6.03
LAS LM (8)	4.56	4.72
LAS-TTS	5.68	5.85
LAS-TTS LM (8)	**4.45**	**4.52**
LAS SC (1)	5.04	5.08
LAS SC (8) LM (64)	**4.20**	**4.33**
LAS SC-MTR (1)	4.87	4.91
LAS SC-MTR (8) LM (64)	**4.12**	**4.28**

Table 4.8: Word error rates (WERs) on LibriSpeech "clean" sets comparing different techniques for incorporating text-only training data. Numbers in parentheses indicate the number of input hypotheses considered by the corresponding model.

it less "clear" and thus have a noisier n-best list. The last two rows in Table 4.8 summarizes the results of SC models trained on errors decoded from MTR TTS data. Performance of the SC model with MTR data improves over clean data. Overall, after applying LM rescoring to the MTR-ed SC model, we achieve a 29.0% relative improvement over the LAS baseline.

4.4.7 Error analysis

To understand the errors made by the LAS model with LM rescoring before and after spelling correction, we pulled a few representative examples. Table 4.9 shows examples of where "LAS + SC + LM rescore" system wins over the "LAS + LM rescore" system. The examples show that SC model does correct many errors of proper nouns and rare words, and also some tense and other grammatical errors.

LAS + LM rescore (error hypotheses)	**LAS + SC + LM rescore** (correct hypotheses)
ready to hand over to trevellion	ready to hand over to **trevelyan**
has countenance the belief the hope the wish that the epeanites or at least the nazarines	has **countenanced** the belief the hope the wish that the **ebionites** or at least the **nazarenes**
a wandering tribe of the blamis or nubians	a wandering tribe of the **blemmyes** or nubians

Table 4.9: LAS + SC + LM rescore Wins. LAS + LM rescore (in bold)

4.5 Conclusion

In this chapter, we explore novel neural-network based approaches for sequence modeling. In the first part of this chapter, CLDNNs were presented to model acoustic sequences for acoustic scene classification tasks. CLDNNs are able to reduce feature variation, perform sequence modeling and discriminate training in an unified architecture. CLDNNs with bidirectional LSTM layers outperform a CNN model with a similar number of parameters. Based on the CLDNN model, we further applied a novel attention mechanism on the LSTM layer, which can predict the importance of each time step and select the most important information from sequences. The proposed attention model is able to generate attention weights on each time step effectively, and provide complementary information to the standard CLDNN model.

In the second part of this chapter, a sequence-to-sequence based spelling error correction model was proposed for end-to-end ASR. The proposed spelling correction (SC) model takes error hypotheses of the LAS model as input and uses an attention-based encoder and decoder architecture to predict the correct transcription sequences. To train the SC model, we generate error hypotheses by decoding the TTS data synthesized from a large text-only corpus. Our results show that the SC model yields clear improvement over the baseline LAS model when directly correcting top LAS hypothesis. By correcting all the entries in the LAS n-best list, the SC model can generate an expanded list which has significantly lower oracle WERs. When further rescoring the expanded n-best list with an external LM, the proposed approach outperforms the simple LM rescoring and direct LAS model training on the TTS

data. In order to make further improvement by alleviating the mismatch between TTS data and real audio, we train the SC model on MTR TTS data. Performance of the SC model with MTR data clearly improves over clean data.

CHAPTER 5

Summary and future work

5.1 Summary

This dissertation mainly focuses on feature representation learning and modeling using deep neural networks for speech and speaker recognition.

In Chapter 2, ways were explored to improve speaker verification performance when only short utterances are available. In order to alleviate possible phoneme mismatch in text-independent short utterance situations, two novel neural-network based representation learning approaches were proposed. The first approach uses a deep neural network to estimate subglottal features from speech signals. The estimated subglottal features are speaker-specific and largely phoneme-invariant providing higher J-ratios and show complementary information when combined with MFCC features on SRE speaker verification tasks. Another utterance-level method was further explored, which learns to reconstruct the long-utterance i-vector from its short-utterance version. Long-utterance i-vectors have less within-speaker variation compared with short-utterance i-vectors and can provide richer speaker information. The proposed mapping algorithms make use of an autoencoder and are able to train a regression model that generalize to unseen speakers. Experiments on various benchmark databases (SRE and SITW) indicate that the proposed method achieves significant improvement over baseline model.

In Chapter 3, research on joint feature learning and acoustic modeling for ASR was presented. Several novel neural network models are proposed, which take complex DFT features or raw waveform as input. First of all, a unified deep Highway network with a time delayed bottleneck layer in the middle was introduced. The TDB-HW networks learn

a bottleneck feature automatically from complex DFT features and provide significant lower error rates compared with LFBE on a large-scale wake-word detection task. Secondly, a 1-D CNN model was presented, which takes raw waveforms as input and uses convolutional and pooling layers to do hierarchical feature extraction. The proposed 1-D CNN model outperforms standard systems with Mel-filter bank features on WSJ large-vocabulary ASR task. We proposed a filter sampling and combination technique, which can naturally enforce parameter sharing and reduce the filter redundancy, but also add to the learning capacity of filters. FSC can reduce the model size by 70% and still improve the performance on ASR tasks.

In Chapter 4, research was conducted on sequence modeling for both acoustic and text sequences. A CLDNN network was first introduced to model acoustic sequences for acoustic scene classification. CLDNNs are able to reduce feature variation, perform sequence modeling and discriminate training in an unified architecture. CLDNNs with bidirectional LSTM layers outperform a CNN model. Based on the CLDNN model, a novel attention mechanism was further applied on the LSTM layer, which can predict the importance of each time step and select the most important information from sequences. The experimental results on TUT acoustic scene classification 2016 dataset show that, the proposed attention model is able to generate attention weights on each time step effectively, and provide complementary information to the standard CLDNN model. For text sequence modeling, a sequence-to-sequence based spelling error correction model was proposed for end-to-end ASR. The proposed spelling correction model takes error hypotheses of the LAS model as input and uses an attention-based encoder and decoder architecture to predict the correct transcription sequences. To train the SC model, we generated error hypotheses by decoding the TTS data synthesized from a large text-only corpus. Our results show that the SC model yields significant improvements over the baseline LAS model on the LibriSpeech database.

5.2 Future work

Several future research opportunities arise from the work in this dissertation. We briefly discuss a few of them below:

For speaker representation learning, the approaches in the dissertation use either frame-level or utterance-level representation. However, humans usually extract representation by looking at different resolutions of speech signal. We believe that by modeling feature extraction using different signal resolutions, speaker representation learning can be more effective.

Semi-supervised learning based i-vector mapping approaches are proposed in this dissertation. Other approaches can be explored to reconstruct long-utterance i-vectors from short-utterance i-vectors. Possible methods include generative adversarial networks or variational autoencoders. Moreover, proposed semi-supervised or unsupervised learning methods should be also helpful for other speaker representation learning frameworks, such as speaker embedding learning.

For joint feature learning and acoustic modeling, in this dissertation, we only use single channel data. We can further extend the proposed framework to multi-channel conditions. We can perform multi-channel enhancement and feature extraction jointly with acoustic modeling. There are already some promising results from recent studies on multi-channel modeling.

In terms of training correction models for ASR, we use a single speaker TTS mode to generate speech data and ASR errors, which results in acoustic mismatch between TTS audio and real audio. In order to generate more realistic data, multi-speaker TTS model and condition-matched TTS model can be used.

REFERENCES

[AAB16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467*, 2016.

[ALS15] Abeer Alwan, Steven Lulich, and Mitchell Sommers. "The Subglottal Resonances Database." In *LDC2015S03*. Philadelphia: Linguistic Data Consortium, 2015.

[BCS16] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. "End-to-End Attention-based Large Vocabulary Speech Recognition." In *Proc. ICASSP*, 2016.

[BLP12] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. "Theano: new features and speed improvements." *arXiv:1211.5590*, 2012.

[BR15] Mayank Bhargava and Richard Rose. "Architectures for deep neural network based acoustic models defined over windowed speech waveforms." In *Proc. Interspeech*, 2015.

[BR17] Pierre-Michel Bousquet and Mickael Rouvier. "Duration Mismatch Compensation Using Four-Covariance Model and Deep Neural Network for Speaker Verification." In *Proc. Interspeech*, pp. 1547–1551, 2017.

[CFB18] M. X. Chen, O. Firat, A. Bapna, and et al. "The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation." In *Proc. Association for Computational Linguistics (ACL)*, 2018.

[CJ17] J. K. Chorowski and N. Jaitly. "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models." In *Proc. Interspeech*, 2017.

[CJL16] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition." In *Proc. ICASSP*, pp. 4960–4964. IEEE, 2016.

[CL16] Sandro Cumani and Pietro Laface. "I-vector transformation and scaling for PLDA based speaker recognition." In *Proc. Odyssey*, pp. 39–46, 2016.

[CMW04] Christopher Cieri, David Miller, and Kevin Walker. "The Fisher Corpus: a Resource for the Next Generations of Speech-to-Text." In *LREC*, volume 4, pp. 69–71, 2004.

[CSW18] C. Chen, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, N. Jaitly, B. Li, and J. Chorowski. "State-of-the-art Speech Recognition With Sequence-to-Sequence Models." In *Proc. ICASSP*, 2018.

[DKD10] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. "Front-end factor analysis for speaker verification." *IEEE Transactions on Audio, Speech, and Language Processing*, **19**(4):788–798, 2010.

[DM80] Steven Davis and Paul Mermelstein. "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences." *IEEE transactions on acoustics, speech, and signal processing*, **28**(4):357–366, 1980.

[DP18] Rohan Kumar Das and SR Mahadeva Prasanna. "Speaker verification from short utterance perspective: a review." *IETE Technical Review*, **35**(6):599–617, 2018.

[DPM00] George R Doddington, Mark A Przybocki, Alvin F Martin, and Douglas A Reynolds. "The NIST speaker recognition evaluation–overview, methodology, systems, results, perspective." *Speech Communication*, **31**(2-3):225–254, 2000.

[ELD16] Hamid Eghbal-Zadeh, Bernhard Lehner, Matthias Dorfer, and Gerhard Widmer. "CP-JKU submissions for DCASE-2016: a hybrid approach using binaural i-vectors and deep convolutional neural networks." *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[Fuk13] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[GFG06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In *ICML*, pp. 369–376, 2006.

[GKS18] Jinxi Guo, Kenichi Kumatani, Ming Sun, Minhua Wu, Anirudh Raju, Nikko Ström, and Arindam Mandal. "Time-delayed bottleneck highway networks using a dft feature for keyword spotting." In *Proc. ICASSP*, pp. 5489–5493, 2018.

[GMP16] Pegah Ghahremani, Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur. "Acoustic Modelling from the Signal Domain Using CNNs." In *Proc. Interspeech*, pp. 3434–3438, 2016.

[GNA17] Jinxi Guo, Usha Amrutha Nookala, and Abeer Alwan. "CNN-Based Joint Mapping of Short and Long Utterance i-Vectors for Speaker Verification Using Short Utterances." In *Proc. Interspeech*, pp. 3712–3716, 2017.

[GPY15] Jinxi Guo, Rohit Paturi, Gary Yeung, Steven M Lulich, Harish Arsikere, and Abeer Alwan. "Age-dependent height estimation and speaker normalization for children's speech using the first three subglottal resonances." In *Proc. Interspeech*, 2015.

[GSW19] Jinxi Guo, Tara N Sainath, and Ron J Weiss. "A spelling correction model for end-to-end speech recognition." *arXiv preprint arXiv:1902.07178*, 2019.

[Guo15] Jinxi Guo. "The analysis and applications of subglottal resonances in height estimation and speaker identification and normalization." *UCLA Master Thesis*, 2015.

[GXC18] Jinxi Guo, Ning Xu, Xin Chen, Yang Shi, Kaiyuan Xu, and Abeer Alwan. "Filter sampling and combination CNN (FSC-CNN): a compact CNN model for small-footprint ASR acoustic modeling using raw waveforms." In *Proc. Interspeech*, pp. 3713–3717, 2018.

[GXL17] Jinxi Guo, Ning Xu, Li-Jia Li, and Abeer Alwan. "Attention Based CLDNNs for Short-Duration Acoustic Scene Classification." In *Proc. Interspeech*, pp. 469–473, 2017.

[GXQ18] Jinxi Guo, Ning Xu, Kailun Qian, Yang Shi, Kaiyuan Xu, Yingnian Wu, and Abeer Alwan. "Deep neural network based i-vector mapping for speaker verification using short utterances." *Speech Communication*, **105**:92–102, 2018.

[GYA17] Jinxi Guo, Ruochen Yang, Harish Arsikere, and Abeer Alwan. "Robust speaker identification via fusion of subglottal resonances and cepstral features." *the Journal of the Acoustical Society of America*, **141**(4):EL420–EL426, 2017.

[GYM16] Jinxi Guo, Gary Yeung, Deepak Muralidharan, Harish Arsikere, Amber Afshan, and Abeer Alwan. "Speaker Verification Using Short Utterances with DNN-Based Estimation of Subglottal Acoustic Features." In *Proc. Interspeech*, pp. 2219–2222, 2016.

[HCE17] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. "CNN architectures for large-scale audio classification." In *Proc. ICASSP*, pp. 131–135, 2017.

[Her90] Hynek Hermansky. "Perceptual linear predictive (PLP) analysis of speech." *the Journal of the Acoustical Society of America*, **87**(4):1738–1752, 1990.

[HHB15] Kun Han, Yanzhang He, Deblin Bagchi, Eric Fosler-Lussier, and DeLiang Wang. "Deep neural network based spectral feature mapping for robust speech recognition." In *Proc. Interspeech*, 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." *Neural computation*, **9**(8):1735–1780, 1997.

[HWW15] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson. "Speech acoustic modeling from raw multichannel waveforms." In *Proc. ICASSP*, pp. 4624–4628, 2015.

[HWZ18] Tomoki Hayashi, Shinji Watanabe, Yu Zhang, Tomoki Toda, Takaaki Hori, Ramon Astudillo, and Kazuya Takeda. "Back-Translation-Style Data Augmentation for End-to-End ASR." *arXiv preprint arXiv:1807.10893*, 2018.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167*, 2015.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KMA18] Waad Ben Kheder, Driss Matrouf, Moez Ajili, and Jean-François Bonastre. "A unified joint model to deal with nuisance variabilities in the i-vector space." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **26**(3):633–645, 2018.

[KMC17] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani. "Generated of Large-scale Simulated Utterances in Virtual Rooms to Train Deep-Neural Networks for Far-field Speech Recognition in Google Home." In *Proc. Interspeech*, 2017.

[KPW17] Kenichi Kumatani, Sankaran Panchapagesan, Minhua Wu, Minjae Kim, Nikko Strom, Gautam Tiwari, and Arindam Mandai. "Direct modeling of raw audio with dnns for wake word detection." In *ASRU*, pp. 252–257, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KSO13] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md Jahangir Alam, and Pierre Dumouchel. "PLDA for speaker verification with utterances of arbitrary duration." In *Proc. ICASSP*, pp. 7649–7653. IEEE, 2013.

[KVD11] Ahilan Kanagasundaram, Robbie Vogt, David B Dean, Sridha Sridharan, and Michael W Mason. "I-vector based speaker recognition on short utterances." In *Proc. Interspeech*, pp. 2341–2344, 2011.

[KWN18] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar. "An Analysis of Incorporating an External Language Model into a Sequence-to-Sequence Model." In *Proc. ICASSP*, 2018.

[LBB98] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LSF14] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren. "A novel scheme for speaker recognition using a phonetically-aware deep neural network." In *Proc. ICASSP*, pp. 1695–1699. IEEE, 2014.

[MFC16] Mitchell McLaren, Luciana Ferrer, Diego Castan, and Aaron Lawson. "The Speakers in the Wild (SITW) Speaker Recognition Database." In *Proc. Interspeech*, pp. 818–822, 2016.

[MG10] Alvin F Martin and Craig S Greenberg. "The NIST 2010 speaker recognition evaluation." In *Proc. Interspeech*, 2010.

[MHV16] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. "TUT database for acoustic scene classification and sound event detection." In *24th European Signal Processing Conference (EUSIPCO)*, pp. 1128–1132, 2016.

[MZX15] Ian McLoughlin, Haomin Zhang, Zhipeng Xie, Yan Song, and Wei Xiao. "Robust sound event classification using deep neural networks." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **23**(3):540–552, 2015.

[OLB18] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. "Parallel WaveNet: Fast High-Fidelity Speech Synthesis." In *Proc. ICML*, pp. 3918–3926, 2018.

[PB92] Douglas B Paul and Janet M Baker. "The design for the Wall Street Journal-based CSR corpus." In *Proceedings of the workshop on Speech and Natural Language*, pp. 357–362. ACL, 1992.

[PC15] Dimitri Palaz, Ronan Collobert, et al. "Analysis of cnn-based speech recognition system using raw speech as input." In *Proc. Interspeech*, 2015.

[PCP15] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. "Librispeech: an ASR corpus based on public domain audio books." In *Proc. ICASSP*, pp. 5206–5210. IEEE, 2015.

[PGB11] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. "The Kaldi speech recognition toolkit." Technical report, IEEE Signal Processing Society, 2011.

[PHM16] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. "Robust audio event recognition with 1-max pooling convolutional neural networks." *arXiv preprint arXiv:1604.06338*, 2016.

[PHV16] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. "Recurrent neural networks for polyphonic sound event detection in real life recordings." In *Proc. ICASSP*, pp. 6440–6444, 2016.

[Pic15] Karol J Piczak. "Environmental sound classification with convolutional neural networks." In *MLSP*, pp. 1–6, 2015.

[PPK15] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. "A time delay neural network architecture for efficient modeling of long temporal contexts." In *Proc. Interspeech*, pp. 3214–3218, 2015.

[PRS17] Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly. "A Comparison of Sequence-to-Sequence Models for Speech Recognition." In *Proc. Interspeech*, pp. 939–943, 2017.

[PSK16] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. "Multi-Task Learning and Weighted Cross-Entropy for DNN-Based Keyword Spotting." In *Proc. Interspeech*, pp. 760–764, 2016.

[PSS17] Arnab Poddar, Md Sahidullah, and Goutam Saha. "Speaker verification with short utterances: a review of challenges, trends and opportunities." *IET Biometrics*, **7**(2):91–101, 2017.

[RJR93] Lawrence R Rabiner, Biing-Hwang Juang, and Janet C Rutledge. *Fundamentals of speech recognition*, volume 14. PTR Prentice Hall, 1993.

[SGP15] David Snyder, Daniel Garcia-Romero, and Daniel Povey. "Time delay deep neural network-based universal background models for speaker recognition." In *ASRU*, pp. 92–97, 2015.

[SGP17] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur. "Deep Neural Network Embeddings for Text-Independent Speaker Verification." In *Proc. Interspeech*, pp. 999–1003, 2017.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." *arXiv preprint arXiv:1505.00387*, 2015.

[SHB16] R. Sennrich, B. Haddow, and A. Birch. "Improving Neural Machine Translation Models with Monolingual Data." In *Proc. Association for Computational Linguistics (ACL)*, 2016.

[SLJ15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[SN12] Mike Schuster and Kaisuke Nakajima. "Japanese and korean voice search." In *Proc. ICASSP*, pp. 5149–5152, 2012.

[Str15] Nikko Strom. "Scalable distributed DNN training using commodity GPU cloud computing." In *Proc. Interspeech*, 2015.

[SVS15] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. "Convolutional, long short-term memory, fully connected deep neural networks." In *Proc. ICASSP*, pp. 4580–4584, 2015.

[SWS15] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals. "Learning the speech front-end with raw waveform CLDNNs." In *Proc. Interspeech*, 2015.

[VSS16] Ehsan Variani, Tara N Sainath, Izhak Shafran, and Michiel Bacchiani. "Complex Linear Projection (CLP): A Discriminative Approach to Joint Feature Extraction and Acoustic Modeling." In *Proc. Interspeech*, pp. 808–812, 2016.

[WM09] Matthias Wölfel and John W McDonough. *Distant speech recognition*. Wiley Online Library, 2009.

[WSC16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." *arXiv preprint arXiv:1609.08144*, 2016.

[WZ89] Ronald J Williams and David Zipser. "A learning algorithm for continually running fully recurrent neural networks." *Neural computation*, **1**(2):270–280, 1989.

[XXD08] Min Xu, Changsheng Xu, Lingyu Duan, Jesse S Jin, and Suhuai Luo. "Audio keywords generation for sports video analysis." *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, **4**(2):11, 2008.

[YD16] Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

[ZCJ17] Y. Zhang, W. Chan, and N. Jaitly. "Very Deep Convolutional Networks for End-to-End Speech Recognition." In *Proc. ICASSP*, 2017.

[ZIS18] A. Zeyer, K. Irie, R. Schluter, and H. Ney. "Improved training of end-to-end attention models for speech recognition." In *Proc. Interspeech*, 2018.

[ZK01] Tong Zhang and C-C Jay Kuo. "Audio content analysis for online audiovisual data segmentation and classification." *IEEE Transactions on speech and audio processing*, **9**(4):441–457, 2001.

[ZK17] Chunlei Zhang and Kazuhito Koishida. "End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances." In *Proc. Interspeech*, pp. 1487–1491, 2017.

[ZMS15] Haomin Zhang, Ian McLoughlin, and Yan Song. "Robust sound event recognition using convolutional neural networks." In *Proc. ICASSP*, pp. 559–563, 2015.

[ZTP14] Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. "Improving deep neural network acoustic models using generalized maxout networks." In *Proc. ICASSP*, pp. 215–219, 2014.