UNIVERSITE BRETAGNE LOIRE

Le Mans Université

Thèse de Doctorat

ITMO UNIVERSITY

Natalia TOMASHENKO

Mémoire présenté en vue de l'obtention du
grade de Docteur de Le Mans Université
sous le sceau de l'Université Bretagne Loire
en cotutelle avec ***ITMO University***

École doctorale : MathSTIC

Discipline : Informatique
Spécialité : Informatique
Unité de recherche : EA 4023 LIUM

Soutenue le 1 décembre 2017
Thèse N° : 2017LEMA1040

Speaker adaptation of deep neural network acoustic models using Gaussian mixture model framework in automatic speech recognition systems

JURY

Rapporteurs :	**Jean-François BONASTRE**, Professeur, Université d'Avignon et des Pays de Vaucluse **Denis JOUVET**, Directeur de Recherche, LORIA-INRIA
Examinateurs :	**Alexey KARPOV,** Professeur, ITMO University **Lori LAMEL**, Directrice de Recherche, LIMSI-CNRS
Directeur de Thèse :	**Yannick ESTEVE**, Professeur, Le Mans Université
Co-directeur de Thèse :	**Yuri MATVEEV**, Professeur, ITMO University
Co-encadrant de Thèse :	**Anthony LARCHER**, Maître de conférences, Le Mans Université

UNIVERSITE BRETAGNE LOIRE

Le Mans Université

Thèse de Doctorat

ITMO UNIVERSITY

Prénom NOM Natalia Tomashenko

Titre de thèse **Utilisation de modèles gaussiens pour l'adaptation au locuteur de réseaux de neurones profonds dans un contexte de modélisation acoustique pour la reconnaissance de la parole**

Title of thesis : **Speaker adaptation of deep neural network acoustic models using Gaussian mixture model framework in automatic speech recognition systems**

Résumé

Les différences entre conditions d'apprentissage et conditions de test peuvent considérablement dégrader la qualité des transcriptions produites par un système de reconnaissance automatique de la parole (RAP). L'adaptation est un moyen efficace pour réduire l'inadéquation entre les modèles du système et les données liées à un locuteur ou un canal acoustique particulier. Il existe deux types dominants de modèles acoustiques utilisés en RAP : les modèles de mélanges gaussiens (GMM) et les réseaux de neurones profonds (DNN). L'approche par modèles de Markov cachés (HMM) combinés à des GMM (GMM-HMM) a été l'une des techniques les plus utilisées dans les systèmes de RAP pendant de nombreuses décennies. Plusieurs techniques d'adaptation ont été développées pour ce type de modèles. Les modèles acoustiques combinant HMM et DNN (DNN-HMM) ont récemment permis de grandes avancées et surpassé les modèles GMM-HMM pour diverses tâches de RAP, mais l'adaptation au locuteur reste très difficile pour les modèles DNN-HMM. L'objectif principal de cette thèse est de développer une méthode de transfert efficace des algorithmes d'adaptation des modèles GMM aux modèles DNN. Une nouvelle approche pour l'adaptation au locuteur des modèles acoustiques de type DNN est proposée et étudiée : elle s'appuie sur l'utilisation de fonctions dérivées de GMM comme entrée d'un DNN. La technique proposée fournit un cadre général pour le transfert des algorithmes d'adaptation développés pour les GMM à l'adaptation des DNN. Elle est étudiée pour différents systèmes de RAP à l'état de l'art et s'avère efficace par rapport à d'autres techniques d'adaptation au locuteur, ainsi que complémentaire.

Mots clés
adaptation au locuteur, apprentissage adaptatif au locuteur (SAT), réseaux de neurones profonds, modèles de mélanges Gaussiens (GMM), paramètres acoustiques dérivés de GMM (GMMD), reconnaissance automatique de la parole (RAP), modèles acoustiques, apprentissage profond

Abstract

Differences between training and testing conditions may significantly degrade recognition accuracy in automatic speech recognition (ASR) systems. Adaptation is an efficient way to reduce the mismatch between models and data from a particular speaker or channel. There are two dominant types of acoustic models (AMs) used in ASR: Gaussian mixture models (GMMs) and deep neural networks (DNNs). The GMM hidden Markov model (GMM-HMM) approach has been one of the most common technique in ASR systems for many decades. Speaker adaptation is very effective for these AMs and various adaptation techniques have been developed for them. On the other hand, DNN-HMM AMs have recently achieved big advances and outperformed GMM-HMM models for various ASR tasks. However, speaker adaptation is still very challenging for these AMs. Many adaptation algorithms that work well for GMMs systems cannot be easily applied to DNNs because of the different nature of these models. The main purpose of this thesis is to develop a method for efficient transfer of adaptation algorithms from the GMM framework to DNN models. A novel approach for speaker adaptation of DNN AMs is proposed and investigated. The idea of this approach is based on using so-called GMM-derived features as input to a DNN. This technique of processing features for DNNs makes it possible to use GMM adaptation algorithms for neural network AMs. The proposed technique provides a general framework for transferring adaptation algorithms, developed for GMMs, to DNN adaptation. It is explored for various state-of-the-art ASR systems and is shown to be effective in comparison with other speaker adaptation techniques and complementary to them.

Key Words
speaker adaptation, speaker adaptive training (SAT), deep neural network (DNN), Gaussian mixture model (GMM), GMM-derived (GMMD) features, automatic speech recognition (ASR), acoustic models, deep learning

L'Université Bretagne Loire

Acknowledgements

This thesis was done in collaboration between two universities – University of Le Mans in France and ITMO University in Saint-Petersburg – and I thank all the people from both universities who organized this collaboration and made this double-degree program possible.

I would like to express my deep gratitude to my supervisor from the LIUM (Laboratoire d'Informatique de l'Université du Maine), Yannick Estève, and my advisor, Anthony Larcher, for supervision, scientific freedom, motivation, enthusiasm, valuable pieces of advice about my work, kindness, inspiration, interesting scientific discussions, encouraging and friendly atmosphere they create for students, for bringing happiness and joy to the work, and for all the opportunities and support that were provided for my work. It is not possible to express in words how happy and grateful I was to work in the LIUM, and I thank everyone there for contributing towards the exciting academic environment and friendly atmosphere: Carole, Mercedes, Sahar, Kévin, Salima, Anthony R., Etienne, Grégor, Sylvain, Walid, Mélanie, Dominique, Antoine, Edwin, Marie, Adrien, Fethi, Ozan, Florent, Daniel, Nathalie, Loïc, Amira, Abdessalam, Emmanuelle, Bruno, Simon, Hakim, and all the others.

Also, I would like to thank my supervisor from the ITMO University, Yuri N. Matveev, and the Department of Speech Information Systems (SIS) in the ITMO University for the opportunity to develop my work within the framework of the double-degree program.

I would like to express my sincere gratitude to Jean-François Bonastre and Denis Jouvet for accepting to be the reviewers of this work and for their insightful comments and suggestions which led to many improvements. My gratitude also goes to other jury members: president of the jury, Lori Lamel, and Alexey Karpov, whose valuable feedback helps to improve this manuscript. I thank them a lot.

I thank the Speech Technology Center (STC) in Saint-Petersburg and many current and former STC colleagues. The years that I worked at the STC on various industrial and research projects related to speech technologies gave me experience in the field and inspired me to complete my PhD in speech recognition. I particularly thank the Research and Development Department of the STC and all the people with whom I worked there: Alexey Prudnikov, Olga Khomitsevich, Irina Chernych, Yuri Khokhlov, and many, many others.

Finally, I thank all my friends and my family for their love and support.

Thank you! Merci ! Спасибо!

Contents

List of Figures

List of Tables

Nomenclature

Notations

Δ	First-order time derivatives of a feature vector
$\Delta\Delta$	Second-order time derivatives of a feature vector
Δ_{abs}WER	Absolute Word Error Rate Reduction
Δ_{rel}WER	Relative Word Error Rate Reduction.
$\mathcal{F}(\mathbf{\Lambda})$	Objective function to be optimized over parameters $\mathbf{\Lambda}$
$\mathbf{h}^l$	Activation vector (typically hidden) at l-th layer in a neural network
$\mathbf{\Lambda}$	Set of model parameters
$\mathcal{N}(.;\boldsymbol{\mu},\mathbf{\Sigma})$	Multivariative Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$
$\mathbf{o}_t$	Speech observation vector at time t
$\widehat{\mathbf{o}}_t$	Context vector composed of several neighboring speech observation vectors: $[\mathbf{o}_{t-T_1}, \ldots, \mathbf{o}_{t+T_2}]$. Usually, except for time-delay neural networks, we consider symmetrical intervals ($T_1 = T_2 = T$).
τ	Parameter that controls the balance between the maximum likelihood estimate of the mean and its prior value in maximum a posteriori adaptation
$[-T_1..T_2]$	where $(T_1, T_2 \geq 0)$ – Sequence of $T_1 + T_2 + 1$ integer numbers in this interval: $-T_1, -T_1+1, \ldots, T_2-1, T_2$. Notation $[-T_1..T_2]$ for feature vector $\mathbf{o}_t$ means, that for this vector a new context vector $\widehat{\mathbf{o}}_t$ is obtained through concatenation of neighboring vectors: $\widehat{\mathbf{o}}_t = [\mathbf{o}_{t-T_1}, ..., \mathbf{o}_t, ..., \mathbf{o}_{t+T_2}]$. Usually, except for time-delay neural networks, we consider symmetrical intervals ($T_1 = T_2 = T$).

Acronyms / Abbreviations

AM	Acoustic Model
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
BLSTM	Bidirectional Long Short Term Memory
BMMI	Boosted Maximum Mutual Information
BN	Bottle-Neck
BP	Backpropagation
BRNN	Bidirectional Recurrent Neural Network
CAT	Cluster Adaptive Training
CD	Context-Dependent
CDF	Cumulative Distribution Function
CE	Cross-Entropy
CI	Context-Independent
CMLLR	Constrained Maximum Likelihood Linear Regression
CMN	Cepstral Mean Normalization
CMVN	Cepstral Mean and Variance Normalization
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
DB	Davies-Bouldin (index)
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
DP	Dynamic Programming

EM	Expectation Maximization
fbanks	filter-bank (coefficients)
FER	Frame Error Rate
fMLLR	feature-space Maximum Likelihood Linear Regression
GMMD	GMM-Derived
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HL	Hidden Layer
HMM	Hidden Markov Model
KLD	Kullback-Leibler Divergence
LDA	Linear Discriminant Analysis
LHN	Linear Hidden Network
LHUC	Learning Hidden Unit Contributions
LIN	Linear Input Network
LM	Language Model
LON	Linear Output Network
LSTM	Long Short Term Memory
LVCSR	Large Vocabulary Continuous Speech Recognition
MAP	Maximum A Posteriori
MBR	Minimum Bayes Risk
MFCC	Mel-Frequency Cepstral Coefficients
MGB	Multi-Genre Broadcast
MLLR	Maximum Likelihood Linear Regression

MLLT	Maximum Likelihood Linear Transform
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MMI	Maximum Mutual Information
MTL	Multi-Task Learning
PCA	Principal Component Analysis
PLP	Perceptual Linear Prediction
PPB	Phoneme Posterior Based
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SAT	Speaker Adaptive Training
SD	Speaker Dependent
SGD	Stochastic Gradient Descent
SGMM	Subspace Gaussian Mixture Model
SI	Speaker Independent
SMAP	Structural Maximum a Posteriori
sMBR	state-level Minimum Bayes Risk
t-SNE	t-Distributed Stochastic Neighbor Embedding
TDNN	Time-Delay Neural Network
UBM	Universal Background Model
VFS	Vector Field Smoothing
VTLN	Vocal Tract Length Normalization

VTS	Vector Taylor Series
WER	Word Error Rate
WSJ	Wall Street Journal

Chapter 1

Introduction

Automatic speech recognition (ASR) is the technology that enables human–machine interaction by allowing human beings to speak with a computer interface. ASR has been an active research area for decades. The progress in the development of ASR technology achieved in the last years increased the use of different ASR systems in everyday life. Interactive voice response (IVR) systems, information extraction and retrieval, automatic closed captioning, dictation, transcription of recorded speech, language learning systems, speech-to-speech translation systems (such as Skype translator), virtual personal assistant devices (such as Apple's Siri, Google Assistant, Amazon Alexa, Microsoft Cortana, Facebook's M), and many other ASR applications are becoming an integral part of our lives.

This progress has become possible largely due to the recent advances in *deep learning* research [Goodfellow et al., 2016], which represents now the mainstream direction for speech recognition development [Yu and Deng, 2014]. Two key factors have contributed to this process: (1) substantial increase of computational power of multi-core processors, general purpose graphical processing units (GPGPUs), and GPU clusters, which allow to train more complex models with a greater number of parameters, and (2) access to more training data.

Despite all these advances, ASR systems are still domain dependent and usually can show high performance only if they are designed for a specific task or environment. Any mismatch between training and testing conditions may degrade the performance. In particular, this mismatch can be caused by different acoustic conditions (such as different speakers, recording channels, background noises, etc.). To overcome this problem, acoustic adaptation is typically applied.

The aim of *acoustic model* (AM) adaptation is to reduce mismatches between training and testing acoustic conditions and improve the accuracy of the ASR system for a target speaker or channel, using a limited amount of adaptation data from the target acoustic source.

This thesis focuses mainly on speaker adaptation, which is aimed to reduce the mismatch caused by inter-speaker variability. Nevertheless, the developed approaches are applicable to a wider range of adaptation tasks.

Adaptation of *deep neural network* (DNN) AMs is a rapidly developing research area. In the recent years, DNNs have replaced conventional *Gaussian mixture models* (GMMs) in most state-of-the-art ASR systems, because it has been shown that DNN *Hidden Markov Models* (HMMs) outperform GMM-HMMs in different ASR tasks [Hinton et al., 2012a]. Many adaptation algorithms that have been developed for GMM-HMM systems, such as *maximum a posteriori adaptation* (MAP) [Gauvain and Lee, 1994], *maximum likelihood linear regression* (MLLR) [Gales, 1998; Leggetter and Woodland, 1995] and others [Shinoda, 2011; Woodland, 2001], cannot be easily applied to DNNs because of the different nature of these models.

Among the adaptation algorithms developed for DNNs, only a few take advantage of robust adaptability of GMMs (Chapter 4). The most common way of using GMMs for DNN model adaptation is through GMM-adapted features. For example, acoustic features adapted with *feature-space maximum likelihood linear regression* (fMLLR) technique are used as input for DNN training in [Kanagawa et al., 2015; Parthasarathi et al., 2015; Rath et al., 2013; Seide et al., 2011a]. In [Lei et al., 2013] likelihood scores from DNN and GMM models, both adapted in the feature space using the same fMLLR transform, are combined at the state level during decoding. A *temporally varying weight regression* (TVWR) is explored in [Liu and Sim, 2014], where DNN posteriors are transformed into time-varying scaling factors for Gaussian weights, using a regression model. However, none of these approaches suggests a universal method to transfer adaptation algorithms from GMM models to DNNs.

The main purpose of this thesis is to develop a framework for efficient transfer of all adaptation algorithms, developed for GMM AMs, to DNN AMs. To achieve this goal, we proposed to use so-called *GMM-derived features* (GMMD) as input to a DNN [Tomashenko and Khokhlov, 2014b]. Then, the proposed adaptation algorithm was extended to the concept of *speaker adaptive training* (SAT) for DNNs [Tomashenko and Khokhlov, 2015].

The desirable property of an adaptation algorithm is *flexibility*. That means that adaptation makes use of all available adaptation data: it improves the speech recognition accuracy even with a small amount of adaptation data, and, when the amount of adaptation data increases, speech recognition accuracy also continues to improve, asymptotically approaching the accuracy of *matched* (in our case, *speaker-dependent* (SD)) AM [Shinoda, 2011]. The effectiveness of the proposed adaptation approach was explored using the most common adaptation algorithms for GMM-HMM – MAP and fMLLR adaptation, as an example. We mostly focus on MAP adaptation in the experiments, because it can provide an additional flexibility to DNN adaptation, as it is not restricted to a single transform, as fMLLR.

Different ways for adaptation performance improvement, such as using confidence scores [Tomashenko et al., 2016b], data selection and data augmentation strategies [Tomashenko et al., 2016d], were proposed and investigated in this thesis.

Starting our research from a classical DNN architecture and simple auxiliary GMM model trained for GMM-derived feature extraction, we are interested in the following questions:

- The first question is how, in terms of topology and basic features, to more effectively train an auxiliary GMM model, which is used for GMM-derived features, in order to achieve better DNN adaptation performance [Tomashenko et al., 2016b].

- The second question concerns the way of efficient integration of GMM-derived features into neural network architectures of state-of-the-art ASR systems [Tomashenko et al., 2016a]. Most of these systems already use normalization and speaker adaptation techniques. How the best improvement over these systems can be obtained using the proposed adaptation approach?

- Is the proposed technique complementary to other widely used algorithms for DNN adaptation, such as fMLLR or i-vectors?

- And finally, can the other more advanced neural network architectures, such as *time-delay neural networks* (TDNN), *recurrent neural networks* (RNN), and others, which nowadays have become dominant in state-of-the art ASR systems, also benefit from the proposed adaptation technique? We are particularity interested in *end-to-end* deep AMs (Chapter 9), because end-to-end systems are an important trend in current ASR technology.

In addition, we aim to look more deeply into the nature of the GMMD features and adaptation techniques associated with them to better understand their properties, strengths and weaknesses and the potential for improvement (Chapter 10).

Thesis structure

The thesis is organized as follows:

- **Chapter** 2 provides an introduction to DNNs, key aspects of their training procedure, and some of the most common neural network architectures currently used in ASR.

- **Chapter 3** gives an overview of HMM-based ASR systems, describes their principal components (including feature extraction; acoustic, pronunciation and language modeling; decoding techniques) with the main focus on acoustic modeling part. Two types of AMs are described, differing in the way they model the state probability distribution: HMM-GMM and hybrid HMM-DNN models. In addition, end-to-end ASR systems are reviewed, as they represent an important trend in current ASR technology. Performance evaluation for ASR systems is described in the final section.

- **Chapter 4** reviews speaker adaptation techniques for both GMM-HMM and DNN-HMM acoustic models.

- **Chapter 5** describes the speech corpora and language models (LM) used to carry out the experiments. Experimental results in each chapter follow theoretical descriptions.

- **Chapter 6** introduces a GMM framework for training DNNs and provides preliminary experimental results with supervised speaker adaptation.

- **Chapter 7** extends the scheme for GMM-derived feature extraction by applying a concept of SAT. It can be considered as consisting of two main parts, representing independent ideas: (1) Section 7.1 and (2) the rest of this chapter (Sections: 7.2, 7.3 and 7.4). Each of these parts has a corresponding section with experimental results.

  In the first part, a SAT procedure for DNN training is presented. In the experiments for this chapter we explore the effectiveness of MAP and fMLLR adaptation, as well as their combination in the proposed approach for adapting an auxiliary GMM model, used for GMMD feature extraction. We look at the dependence of the adaptation behavior for different algorithms on the amount of adaptation data.

  In the second part, several techniques for adaptation performance improvement are proposed and studied: using lattice scores, data augmentation and data selection. For this techniques we used a different experimental setup, with larger and more complex models.

  For this chapter as well as for all the following ones experimental results are reported for unsupervised adaptation mode.

- **Chapter 8** investigates various ways of integrating GMM-derived features into different state-of-the-art neural network architectures (DNN and TDNN). To build a stronger ASR system, we took as a basis conventional Kaldi [Povey et al., 2011b] recipes for AMs and aim to integrate our adaptation algorithm in these recipes. We perform this integration in different ways: feature concatenation of the proposed GMMD features with conventional features for training DNNs, fusion of posterior from different DNNs, lattice-level fusion of recognition results, and others. Also we compare the proposed approach with the most common feature-space adaptation techniques, such as fMLLR and i-vectors, for DNN and TDNN AMs. In addition, in Section 8.7 we report some results of applying the proposed GMMD features in the *MGB Challenge 2016*[1] as a part of the LIUM ASR system [Tomashenko et al., 2016f].

- **Chapter 9** explores the effectiveness of the proposed adaptation technique in application to end-to-end AMs, taking as an example bidirectional long short term memory (BLSTM) acoustic models trained with connectionist temporal classification (CTC) criterion. Three different speaker adaptation algorithms have been implemented to this type of AMs and experimentally analyzed: (1) fMLLR adaptation, (2) adaptation using i-vectors, and (3) the proposed algorithm with MAP adaptation using GMMD features. Furthermore, a comparative study of the adaptation techniques was conducted for CTC AMs and TDNN AMs trained with the traditional frame-wise cross-entropy criterion.

- **Chapter 10** analyzes properties of the proposed GMM-derived features and adaptation algorithm. For this analysis, we use *phoneme posterior based* (PPB) features, obtained from decoding lattices. Visual *t-distributed stochastic neighbor embedding* (t-SNE) analysis [Maaten and Hinton, 2008] is performed for different phoneme groups. *Davies-Bouldin* (DB) index [Davies and Bouldin, 1979] and other statistics are also used in this study.

- **Chapter 11** contains a summary and discusses possible future work.

[1]The Multi-Genre Broadcast (MGB) challenge: http://www.mgb-challenge.org/

Chapter 2

Deep neural networks

This chapter introduces deep neural networks (DNNs) and provides an overview of some of the most common neural network architectures used in ASR.

2.1 Introduction

In the ASR literature, the term *deep neural network* (DNN) was originally referred to as *feed-forward* artificial neural network (or *multilayer perceptron* (MLP) [Rosenblatt, 1961]) with more than one *hidden layer* [Hinton et al., 2012a; Seide et al., 2011b]. Later this term was extended to the meaning of any neural network with a deep structure [Yu and Deng, 2014]. In this thesis we will use the term DNN mainly when referring to feed-forward models and explicitly specify the type of the architecture (feed-forward, convolutional, recurrent, etc.), when it is necessary.

DNNs play an important role in modern ASR systems, particularly in acoustic modeling. It was shown, that for various ASR tasks, DNN AMs outperform traditional GMM AMs [Hinton et al., 2012a].

Further in this chapter we review a feed-forward DNN architecture, training procedure including some practical considerations, and different alternative types of neural network architecture that nowadays are used in ASR systems.

2.2 Deep neural network architecture

The architecture of a conventional DNN can be described as follows [Hinton et al., 2012a]. Let denote the l-th layer of a DNN as $\mathbf{h}^l$, and the total number of layers in a DNN as $L+1$, so that layers have indexes: $0, \dots, L$. Then

$$\mathbf{h}^l = f^l(\mathbf{z}^l) = f^l\left(\mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l\right) \text{ for } 0 < l \leq L, \tag{2.1}$$

where

$$\begin{cases} \mathbf{z}^l = \mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l \in \mathbb{R}^{N_l} \text{ – \textit{excitation vector}}, \\ \mathbf{h}^l \in \mathbb{R}^{N_l} \text{ – \textit{activation vector}}, \\ \mathbf{W}^l \in \mathbb{R}^{N_l \times N_{l-1}} \text{ – weight matrix}, \\ \mathbf{b}^l \in \mathbb{R}^{N_l} \text{ – bias vector}, \\ N_l \text{ – number of neurons in layer } l \text{ (or \textit{layer dimension})}, \\ \mathbf{h}^0 = \mathbf{o} \in \mathbb{R}^{N_0} \text{ – input observation feature vector}, \\ N_0 = D \text{ – feature dimension}, \\ f^l(\cdot) : \mathbb{R}^{N_l} \to \mathbb{R}^{N_l} \text{ – \textit{activation function}}. \end{cases} \tag{2.2}$$

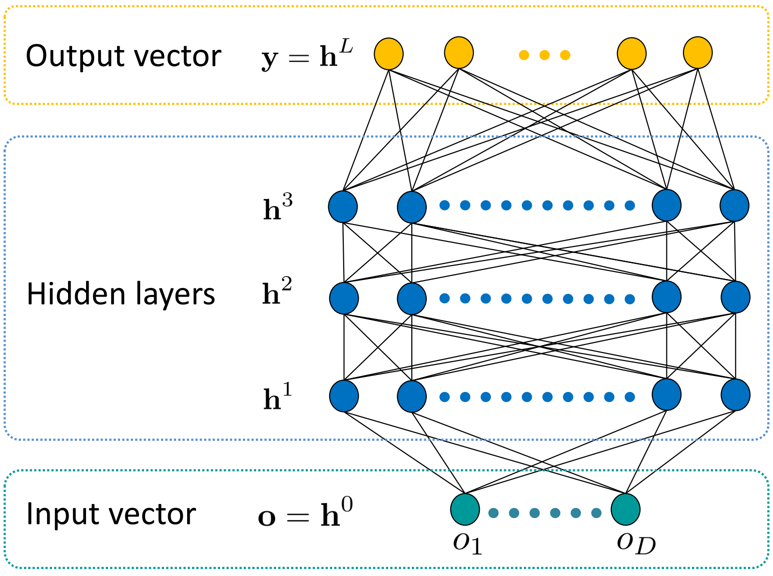


Figure 2.1 Example of a deep neural network with an input layer, three hidden layers, and an output layer

The activation function f^l is applied element-wise to the excitation vector. Let z_i be i-th element of vector $\mathbf{z}$ of linear activations, obtained from (2.2). For simplicity, we omit layer index l in the notation. There are several types of activation functions, among which the most commonly used are:

- *Identity function*:
$$\mathrm{I}(z_i) = z_i. \tag{2.3}$$
Identity function is usually used in the output layer for regression tasks.

- *Sigmoid (standard logistic) function*:
$$\sigma(z_i) = \frac{1}{1+e^{-z_i}}. \tag{2.4}$$

- *Hyperbolic tangent function*:
$$\tanh(z_i) = \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}}. \tag{2.5}$$
Hyperbolic tangent function is connected with the sigmoid function: $\tanh(z) = 2\sigma(2z) - 1$ and is different from the sigmoid function in the output range of values: $\tanh(z) \in (-1,1)$, while $\sigma(z) \in (0,1)$. So the *tanh* function is symmetric around zero, and is recommended, for example in [LeCun et al., 2012], for more efficient training.

- *Rectified linear unit (ReLU) function*:
$$\mathrm{ReLU}(z_i) = \max(0, z_i). \tag{2.6}$$
The ReLUs [Jaitly and Hinton, 2011; Nair and Hinton, 2010] create sparse representations in a DNN structure. They are efficient in combination with *dropout* [Srivastava et al., 2014] regularization techniques [Dahl et al., 2013].

- *Maxout function* [Goodfellow et al., 2013]:
$$\mathrm{maxout}\left(\{z_i\}_{i\in R}\right) = \max\left(\{z_i\}_{i\in R}\right), \tag{2.7}$$
where R is the number of linear activations. A single maxout unit can be considered as a piecewise linear approximation to any convex function, and maxout network with two hidden units can approximate arbitrarily well any continuous functions.

- *Softmax function* [Bridle, 1990]:
$$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N_l} e^{z_j}}. \tag{2.8}$$

- *L_p-norm function* [Boureau et al., 2010; Gulcehre et al., 2014; Zhang et al., 2014]:

$$\left\|\{z_i\}_{i\in R}\right\|_p = \left(\sum_{i\in R} |z_i|^p\right)^{\frac{1}{p}}, \tag{2.9}$$

  where $p \geq 1$ is the order of the norm, and R is the set of linear activations, for which the norm is calculated.

In this thesis we will apply sigmoid (2.4) and ReLU (2.6) activation functions for hidden layers in different DNN-HMM AM setups, and the softmax activation function (2.9) will be used for the output layer.

The type of the output layer depends on the task:

- For the *regression task* it is a linear layer (with the identity activation function):

$$\mathbf{h}^L = \mathbf{z}^L = \mathbf{W}^L\mathbf{h}^{L-1} + \mathbf{b}^L. \tag{2.10}$$

- For the *multi-class classification task* each output neuron represents a class $i \in 1, \ldots, N_L$, and the value of i-th output neuron h_i^L represents the probability $P_{DNN}(i|\mathbf{o})$ that vector $\mathbf{o}$ belongs to class i. To simulate a probability distribution, the softmax function (2.9) is used:

$$h_i^L = P_{DNN}(i|\mathbf{o}) = \mathrm{softmax}(z_i). \tag{2.11}$$

2.3 Training

Let $\mathbf{y}$ be an output vector of a DNN, corresponding to an observation vector $\mathbf{o}$. As before (see Formula (2.2)), $[\mathbf{W}, \mathbf{b}]$ denotes a set of DNN parameters to be estimated from training samples $\mathbb{T} = \{(\mathbf{o}^m, \mathbf{y}^m) : 0 \leq m \leq M\}$, where $\mathbf{o}^m$ is the m-th observation vector, and $\mathbf{y}^m$ is the corresponding target vector. The process of $[\mathbf{W}, \mathbf{b}]$ parameter estimation (or DNN training) is characterized by a *training criterion* and a *learning algorithm*.

2.3.1 Training criteria

In order to estimate parameters of a DNN, a suitable training criterion (or a loss function) should be specified. Two aspects should be taken into account when choosing a loss function $\mathcal{F}$:

1. Simplicity of evaluation.
2. Correlation with the final goal of the task.

Several popular training criteria are used for DNN training:

- *Mean square error (MSE)*:

$$\mathcal{F}_{MSE}(\mathbf{W},\mathbf{b};\mathbb{T}) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{F}_{MSE}(\mathbf{W},\mathbf{b};\mathbf{o}^m,\mathbf{y}^m), \tag{2.12}$$

  where

$$\mathcal{F}_{MSE}(\mathbf{W},\mathbf{b};\mathbf{o},\mathbf{y}) = \frac{1}{2}\|\mathbf{h}^L-\mathbf{y}\|^2 = \frac{1}{2}\left(\mathbf{h}^L-\mathbf{y}\right)^{\mathrm{T}}\left(\mathbf{h}^L-\mathbf{y}\right). \tag{2.13}$$

  This criterion is used for regression tasks.

- *Cross-entropy (CE)*:

$$\mathcal{F}_{CE}(\mathbf{W},\mathbf{b};\mathbb{T}) = \frac{1}{M}\sum_{m=1}^{M}\mathcal{F}_{CE}(\mathbf{W},\mathbf{b};\mathbf{o}^m,\mathbf{y}^m), \tag{2.14}$$

  where

$$\mathcal{F}_{CE}(\mathbf{W},\mathbf{b};\mathbf{o},\mathbf{y}) = -\sum_{i=1}^{N_L} y_i \log h_i^L, \tag{2.15}$$

  where $y_i = P_{target}(i|\mathbf{o})$ is the observed in the training set empirical probability that the observation vector $\mathbf{o}$ belongs to class i, and $h_i^L = P_{DNN}(i|\mathbf{o})$ is the same probability estimated from the DNN. This criterion is used for the classification tasks, where $\mathbf{y}$ is a probability distribution. Minimizing CE criterion is equivalent to minimizing the Kullback-Leibler divergence (KLD) between the empirical probability distribution (of targets) and the probability distribution estimated from the DNN. In many tasks, hard class labels are used as targets:

$$y_i = \begin{cases} 1, & \text{if } c_{\mathbf{o}} = i \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

  where $c_{\mathbf{o}}$ is the class label of the observation vector $\mathbf{o}$ in the training set. In this case the CE criterion (2.15) becomes the *negative log-likelihood criterion* (NLL):

$$\mathcal{F}_{NLL}(\mathbf{W},\mathbf{b};\mathbf{o},\mathbf{y}) = -\log h_{c_{\mathbf{o}}}^L. \tag{2.17}$$

2.3.2 Learning algorithm

Given the training criterion, the parameters of a DNN $[\mathbf{W},\mathbf{b}]$ can be discriminatively trained (DT) with the error *backpropagation algorithm* (BP) [Rumelhart et al., 1986] by propagating derivatives of a loss function. In the simplest version, the DNN parameters can be improved using the first-order gradient information as follows [Yu and Deng, 2014]:

$$\mathbf{W}_{t+1}^{l} \leftarrow \mathbf{W}_{t}^{l} - \varepsilon \Delta \mathbf{W}_{t}^{l} \tag{2.18}$$

and

$$\mathbf{b}_{t+1}^{l} \leftarrow \mathbf{b}_{t}^{l} - \varepsilon \Delta \mathbf{b}_{t}^{l}, \tag{2.19}$$

where ε is a *learning rate*; $\mathbf{W}_{t}^{l}$ and $\mathbf{b}_{t}^{l}$ are the weight matrix and the bias vector of the layer l after the t-th update;

$$\Delta \mathbf{W}_{t}^{l} = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{\mathbf{W}_{t}^{l}} \mathcal{F}(\mathbf{W},\mathbf{b};\mathbf{o}^{m},\mathbf{y}^{m}) \tag{2.20}$$

and

$$\Delta \mathbf{b}_{t}^{l} = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{\mathbf{b}_{t}^{l}} \mathcal{F}(\mathbf{W},\mathbf{b};\mathbf{o}^{m},\mathbf{y}^{m}) \tag{2.21}$$

are the average weight matrix gradient and the average bias vector gradient correspondingly at iteration t computed on M_b samples and $\nabla_{\mathbf{x}}\mathcal{F}$ is the gradient of $\mathcal{F}$ with respect to $\mathbf{x}$.

The parameter updates in Formulas (2.18) and (2.19) are estimated on a batch of training samples. The choice of the batch size influences the final result of the training, as well as the convergence speed. In the simplest approach, which is referred to the *batch training*, the batch is the whole training set. An alternative approach is to use the *stochastic gradient descent* (SGD) algorithm [Bishop, 2006; Bottou, 1998], where gradients are estimated from a single sample. However, the most common technique is to compute the derivatives on small, randomly chosen *mini-batches* of the training samples. In this thesis we will use the last approach, which also referred to as the *mini-batch SGD* algorithm in the literature [Li et al., 2014c].

There are some practical considerations that have to be taken into account for efficient DNN training [Ruder, 2016; Yu and Deng, 2014]. One important question is the choice of the learning rate ε. If a learning rate is too small the learning algorithm has too slow convergence, and on the contrary, if it is too large, it can prevent learning from convergence to optimal solution. Learning rate schedules [Darken and Moody, 1991] aim to regulate the learning rate during the training according to a fixed learning rate schedule, depending on the changes in the objective function, or component-wise, depending on the geometry of the observed data and the parameter sparseness. Popular gradient descent optimization algorithms include [Ruder, 2016]: *momentum*, *Nesterov accelerated gradient* (NAG) [Nesterov, 1983], *Adagrad* [Duchi et al., 2011], *Adadelta* [Zeiler, 2012], *adaptive moment estimation (Adam)* [Kingma and Ba, 2014], natural gradient descent [Amari, 1998], and *RMSprop* [Tieleman and Hinton, 2012].

Batch normalization technique, proposed in [Ioffe and Szegedy, 2015], allows significant acceleration of the DNN training. It consists in a normalization step that fixes the means and variances of layer inputs.

Below we consider some other important issues related to DNN training procedure, such as *momentum*, *pre-training*, and *regularization techniques*.

2.3.3 Momentum

In order to speed up the training and to smooth the parameter updates, the information about the previous updates is included into the gradient update in the form of *momentum* [Qian, 1999; Rumelhart et al., 1985]. When the momentum is applied, Formulas (2.18) and (2.19) are replaced with:

$$\mathbf{W}^l_{t+1} \leftarrow \mathbf{W}^l_t - \varepsilon \Delta \mathbf{W}^l_t + \alpha \Delta_{t-1}(\mathbf{W}^l) \tag{2.22}$$

and

$$\mathbf{b}^l_{t+1} \leftarrow \mathbf{b}^l_t - \varepsilon \Delta \mathbf{b}^l_t + \alpha \Delta_{t-1}(\mathbf{b}^l), \tag{2.23}$$

where

$$\Delta_t(\mathbf{W}^l) = \mathbf{W}^l_{t+1} - \mathbf{W}^l_t \tag{2.24}$$

and

$$\Delta_t(\mathbf{b}^l) = \mathbf{b}^l_{t+1} - \mathbf{b}^l_t, \tag{2.25}$$

and $0 < \alpha < 1$ is a *momentum coefficient (factor)*.

2.3.4 Regularization

Overfitting can be a serious problem in DNN training due to the large number of estimated parameters. There are different ways to control and prevent over-fitting in DNN training.

One solution to reduce over-fitting is to apply some *regularization terms* for the DNN parameter updates. Regularization aims to incorporate some prior information into the training criteria, to prevent the model from learning undesirable configurations. The common way is to add a complexity term (penalty) to the loss function to penalizes certain configurations. The most commonly used regularization terms include:

1. *L_1-penalty*:

$$R_1(\mathbf{W}) = \|\text{vec}(\mathbf{W})\|_1 = \sum_{l=1}^{L} \left\|\text{vec}(\mathbf{W}^l)\right\|_1 = \sum_{l=1}^{L}\sum_{i=1}^{N_l}\sum_{j=1}^{N_{l-1}} |W_{ij}^l| \tag{2.26}$$

   is based on L_1-norm.

2. *L_2-penalty*:

$$R_2(\mathbf{W}) = \|\text{vec}(\mathbf{W})\|_2^2 = \sum_{l=1}^{L} \left\|\text{vec}(\mathbf{W}^l)\right\|_2^2 = \sum_{l=1}^{L}\sum_{i=1}^{N_l}\sum_{j=1}^{N_{l-1}} (W_{ij}^l)^2 \tag{2.27}$$

   is based on L_2-norm.

In Formulas (2.26) and (2.27), W_{ij} denotes the (i, j)-th element in matrix $\mathbf{W}$, and $\text{vec}(\mathbf{W}^l)$ is the vector, obtained by concatenation all the columns in the matrix $\mathbf{W}^l$. The regularization terms are often referred to as *weight decay* in the literature. When the regularization is applied, the training criterion is changed to:

$$\mathcal{F}_{reg}(\mathbf{W},\mathbf{b};\mathbb{T}) = \mathcal{F}(\mathbf{W},\mathbf{b};\mathbb{T}) + \lambda R_p(\mathbf{W}), \tag{2.28}$$

where $p \in \{1, 2\}$, depending on the chosen penalty type; and $\mathcal{F}(\mathbf{W},\mathbf{b};\mathbb{T})$ is a CE or MSE loss function, that optimizes the empirical loss on the training set $\mathbb{T}$.

2.3.5 Dropout

Dropout is an alternative powerful regularization technique [Dahl et al., 2013; Hinton et al., 2012b; Srivastava et al., 2014] which consists in removing a specified percentage of randomly selected hidden units or inputs during training or pre-training to improve generalization. When a hidden neuron is dropped out in training, its activation is set to 0. During the training with dropout, a random subset of units should be repeatedly sampled. This slows down the training process. A fast dropout training algorithm was proposed in [Wang and Manning, 2013] and is based on using Gaussian approximation instead of doing Monte Carlo optimization.

2.3.6 Pre-training

Pre-training techniques provide an effective way of weight initialization and also can propose a solution for the over-fitting problem [Erhan et al., 2010]. Various approaches to DNN pre-training include [Yu and Deng, 2014]:

- *Generative pre-training*:
  - *restricted Boltzmann machines* (RBMs) [Hinton, 2010; Nair and Hinton, 2010; Smolensky, 1986]. An RBM is a stochastic neural network, where binary activations depend on their neighbors and have probabilistic binary activation functions.
  - *Deep belief networks* (DBNs) [Hinton et al., 2006].
  - *Denoising autoencoders* [Bengio et al., 2007].
- *Discriminative pre-training* (DPT). In DPT, layers can be pre-trained layer-wise using BP as follows. First, a DNN with only one hidden layer is trained discriminatively to convergence using labels. Then, another hidden layer with randomly initialized weights is inserted before the output layer. After that the new network is discriminatively trained again. Each time, when a new hidden layer is added to the network, all layers are updated using BP. This procedure is repeated until the required number of hidden layers is reached.

2.4 Alternative neural network architectures

2.4.1 Convolutional neural network (CNN)

Convolutional neural network (CNN) [LeCun et al., 1995, 1998] is a neural network architecture designed for data that has a grid-structure topology [Goodfellow et al., 2016].

The development of CNNs is claimed to be originally inspired by biological neural processes. Neurophysiologists [Hubel and Wiesel, 1962] explored how neurons in the cat's brain responded to images on a screen in front of the cat. In this experiment they found out that in the primary visual cortex of the brain, some part of cells (so-called, *simple cells*) strongly respond to specific edge-like patterns, but almost do not respond to other patterns. Another part of cells (*complex cells*) have wider *receptive fields* (sub-regions of the visual field, to which the cells are sensitive) and are locally invariant to small changes in the position of the pattern. This phenomenon inspired the development of some pooling strategies in CNNs, such as, maxout units [Goodfellow et al., 2013] and different models, such as, *NeoCognitron* [Fukushima, 1988], *HMAX* [Serre and Riesenhuber, 2004; Serre et al., 2007], *LeNet* [LeCun et al., 1998].

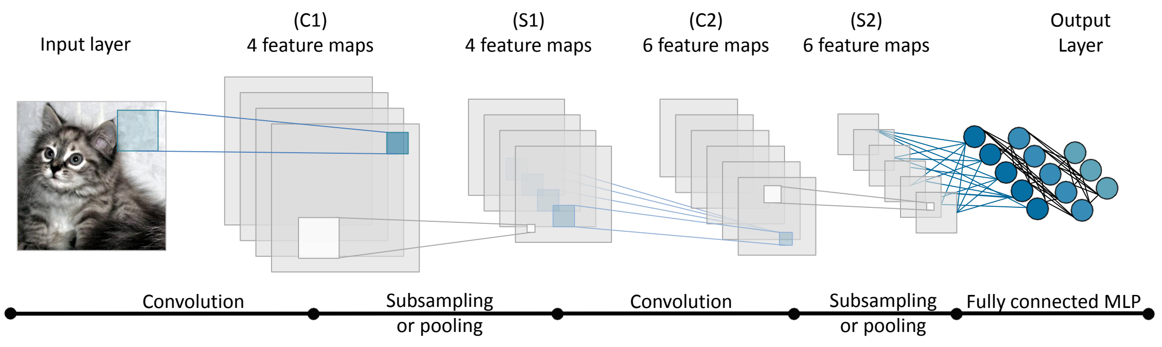


Figure 2.2 Example of the convolutional neural network for image processing. The figure was adapted with some modifications based on the idea from [LeCun et al., 1995].

The modern CNNs were proposed in [LeCun et al., 1995, 1998, 1989] for image processing. The following ideas were introduced into the neural network architecture [LeCun et al., 1998]: *convolution*, *local receptive fields*, *shared weights* and spatial or temporal *subsampling* (or *pooling*) in order to achieve a certain degree of invariance to scale, shift and distortions of the input [LeCun et al., 1998].

A typical example of a CNN architecture is depicted in Figure 2.2. It contains two *convolutional* layers (C1 and C2), two *subsampling* (or *pooling*) layers (S1 and S2), and several fully-connected layers. The lower-layers are composed of convolution and subsampling (or *max-pooling* layers). The upper-layers are fully-connected and correspond to a simple MLP with several hidden layers. Units in a layer are organized in planes, within which all the units share the same set of weights. The set of outputs from units in a plane is called a *feature map*. A convolutional layer is composed of several feature maps with different sets of weights and biases. Different feature maps extract different types of local features from the input. The input to the first fully-connected layer is the set of all features maps from the lower layer.

In general, a CNN can be described by the following operations:

- *Convolution*. The convolution operation, denoted as

$$c(t) = (x * k)(t) = \int x(\tau)k(t-\tau)d\tau, \tag{2.29}$$

can be considered in the discrete case as multiplication of *input* x by a matrix *kernal* (also called *filter*) k. The output is referred to as the *feature map*. In practice input x is usually a multidimensional array of data and kernel k is a multidimensional array of parameters that are trained by the learning algorithm. The kernel typically has a much smaller size than the input.

- *Pooling*. The pooling operation maps the output at a certain region to a summary statistic of neighboring units in this region. Examples of pooling include: the *max-pooling* [Riesenhuber and Poggio, 1999; Zhou and Chellappa, 1988], average of a rectangular neighborhood, the L_p-norm of a rectangular neighborhood [Boureau et al., 2010], a weighted average based on the distance from the central point, and others [Jia et al., 2012; Swietojanski and Renals, 2016]. Pooling helps to make the representation be approximately invariant to small changes of the input. Also for many tasks, pooling is important for handling inputs of different sizes.

These operations introduce several useful properties into the neural network architecture, that can help to improve a machine learning system: *sparse interactions* and *parameter sharing* [Goodfellow et al., 2016]. Also they allow a neural network to work with inputs of variable size.

Convolutional networks play an important role in the history of neural networks. In the past they were applied to many tasks including handwriting recognition [Bottou et al., 1994], on-line handwritten word recognition [Bengio et al., 1995], face recognition [Lawrence et al., 1997] and others. In image processing, a deep CNN was used to win the *ImageNet* image classification challenge [Krizhevsky et al., 2012], and more recently, the 152-layer *residual network* (*ResNet*)[1] has been introduced [He et al., 2016] to significantly reduce the ImageNet classification error rate.

In recent years, CNNs have become widely used in various state-of-the-art applications, for example, in computer vision: [Jia et al., 2014; LeCun et al., 2010; Szegedy et al., 2015], in speech technology: [Chen et al., 2016; Li et al., 2016; Manenti et al., 2016], and particularly, in ASR: [Abdel-Hamid et al., 2013, 2012; Deng et al., 2013a; Ghahremani et al., 2016; Sainath et al., 2013a,b; Suzuki et al., 2016; Swietojanski et al., 2014; Zhang et al., 2016b], CNNs with a very deep *VGG*[2] network architecture [Sercu et al., 2016; Yu et al., 2016]. In ASR systems, for acoustic modeling, the convolution can be done either in time [Yoshioka et al., 2015] or in frequency domain [Abdel-Hamid et al., 2014], or in both [Mitra and Franco, 2015]. Applying convolution in the time domain allows a neural network to be invariant to

[1] *Residual networks (ResNets)* are the type of NNs, where a residual function of the input is learned using skip connections. ResNet framework was designed [He et al., 2016] to train very deep neural networks, which are easy to optimize and which avoid poor optimization or generalization problems.

[2] The *VGG* (Visual Geometry Group) network architecture was introduced in [Simonyan and Zisserman, 2014] for *ImageNet Challenge 2014* submission.

shifts in time. Using convolution in the frequency domain makes a neural network invariant to frequency.

2.4.2 Time-delay neural network (TDNN)

Time-delay neural networks (TDNNs) belong to a particular type of CNNs that share weights along a single temporal dimension. Initially TDNN models have been proposed for the phoneme recognition task in [Lang and Hinton, 1988; Waibel et al., 1989], and later they were applied for spoken word recognition [Bottou et al., 1990] and for on-line handwriting recognition [Guyon et al., 1991] tasks. TDNNs allow the acoustic model to learn the temporal dynamics of the speech signal using short term acoustic feature vectors. Recently AMs with the TDNN topology have been shown to outperform state-of-the-art DNN-HMM ASR systems for many tasks [Peddinti et al., 2015; Povey et al., 2016].

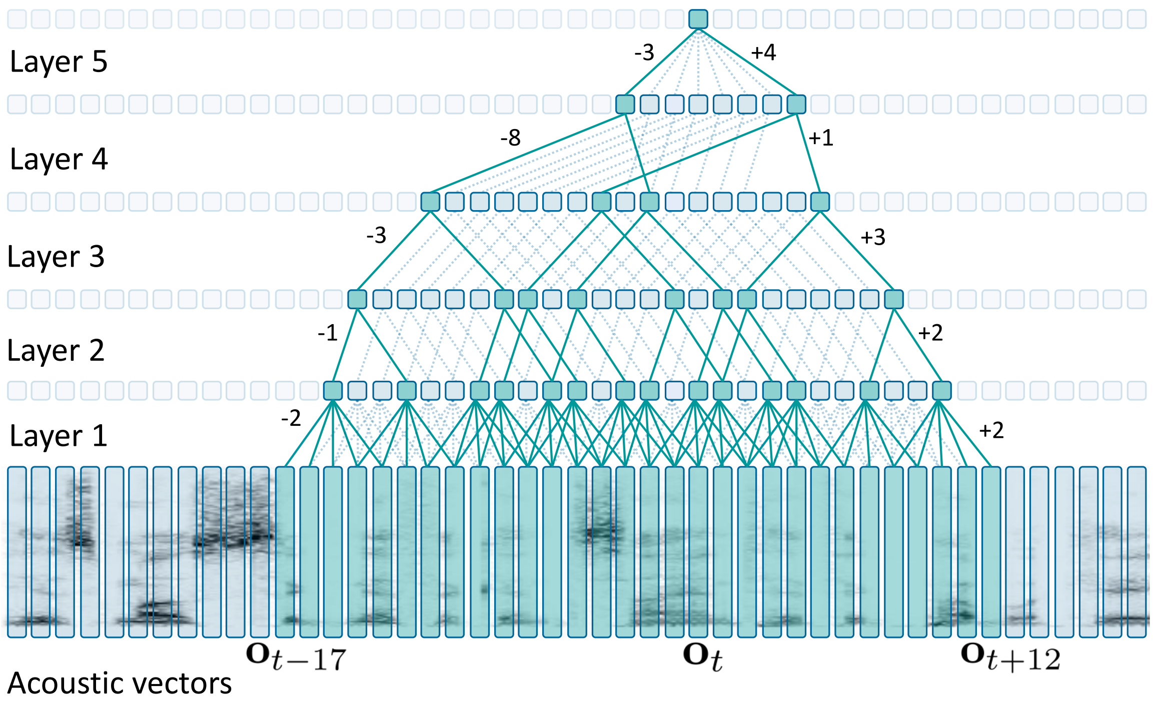


Figure 2.3 Example of feature expansion and subsampling in the time-delay neural network. The figure was adapted with some modifications based on the idea from [Peddinti et al., 2015].

Figure 2.3 shows an example of layer-wise context expansion scheme for a TDNN model. Each frame of the higher level corresponds to a longer context than the lower layers. This hierarchical temporal context expansion is different from the using of a wide contextual window of acoustic feature vectors in the input layer, as it is done in a DNN model. Different layers in the TDNN model correspond to different levels of abstraction in information, extracted from the speech signal: local patterns in speech signal can be captured by the lower layers, while more complex structures can be learned by higher levels.

In the given example (Figure 2.3), each layer has its own context extension and sub-sampling characteristics. For example, the first layer operates on the window of 5 frames $\{\mathbf{o}_{t-2}, \mathbf{o}_{t-1}, \mathbf{o}_t, \mathbf{o}_{t+1}, \mathbf{o}_{t+2}\}$ of the input features. We will denote it as $\{-2,-1,0,1,2\}$ (or simply, $[-2,2]$) following the notations from [Peddinti et al., 2015; Povey et al., 2011b]. Layer 2 operates on the window of 4 frames of the Layer 1: $[-1,2]$. In addition, Layer 2 has a sub-sampling, so it utilizes only boundary frames $\{-1,2\}$ from context window $[-1,2]$. The frame connections that are used in TDNN after sub-sampling are shown in the figure with solid lines (the light dotted lines correspond to frame connections without subsampling). Finally, we can see, that the top layer has an indirect connection to the input acoustic vectors layer by means of context window $[-17,12]$.

This type of neural network will be used for AMs in our experiments. The layer-wise context expansion in time dimension has been explored also in [Amodei et al., 2015; Yu et al., 2016]. Sub-sampling and context extension in the network was also used in stacked bottle-neck networks [Grézl et al., 2014], but in that setup both neural networks were trained separately.

2.4.3 Recurrent neural network (RNN)

Recurrent neural networks (RNNs) is a type of neural networks developed to process sequential data. RNNs provide a powerful extension of feed-forward DNN models by adding connections between different type of units, regardless from their positions within the neural network topology. Possible connections include backward connections to previous layers, self-recurrent loops of units and others. A directed cycle is a basic type of connections between different units in RNN models. The use of recurrence over the temporal dimension allows RNNs to model the dynamic temporal behavior.

Early important types of RNNs, suggested in the literature, include:

- *Jordan* RNN [Jordan, 1986, 1997]: contains recurrent connections that allow the network's hidden units to see its own previous output, so that the subsequent behavior can be formed by the previous responses (Figure 2.4).

- *Elman* RNN or *simple recurrent networks* (SRNs) [Elman, 1990]: has additional *context units* that come from the hidden layer and are augmented with the input layer (Figure 2.5).

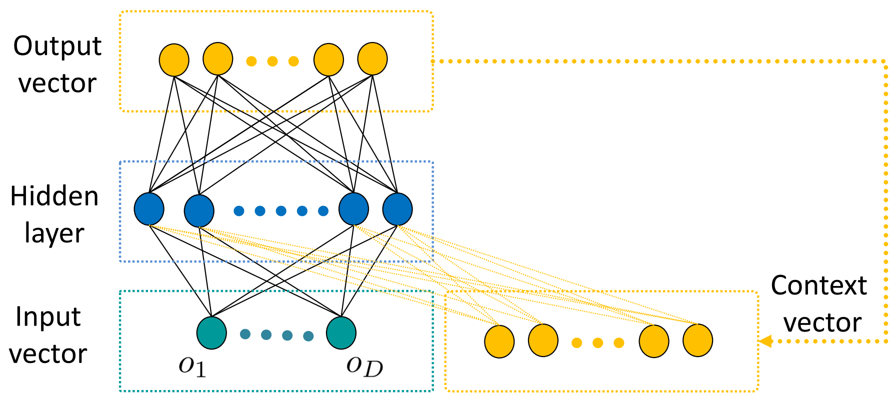


Figure 2.4 *Jordan* recurrent neural network

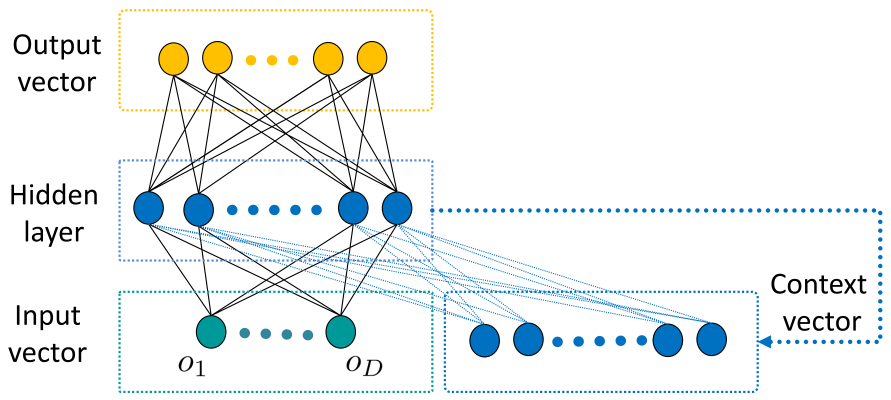


Figure 2.5 *Elman* recurrent neural network

The main difference of RNNs from DNNs consists in the fact that RNNs operate not only on input vectors, but also on internal states of the model. The internal states of the RNN encode the information about the temporal sequence of the past process that was already processed by this RNN. A simple RNN with a single hidden layer can be described with the observation and state equations as follows:

$$\begin{cases} \mathbf{h}_t = f(\mathbf{W}_{oh}\mathbf{o}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{yh}\mathbf{y}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t = g(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y), \end{cases} \tag{2.30}$$

where

$$\begin{cases}
\mathbf{o}_t \in \mathbb{R}^D \text{ – vector of inputs,} \\
\mathbf{h}_t \in \mathbb{R}^N \text{ – vector of hidden state values,} \\
\mathbf{y}_t \in \mathbb{R}^L \text{ – vector of outputs,} \\
\mathbf{W}_{hy} \in \mathbb{R}^{L\times N} \text{ – weight matrix connecting N hidden units to L outputs,} \\
\mathbf{W}_{oh} \in \mathbb{R}^{N\times D} \text{ – weight matrix connecting D inputs to N hidden units,} \\
\mathbf{W}_{hh} \in \mathbb{R}^{N\times N} \text{ – weight matrix connecting N hidden units from time } t-1 \text{ to time } t, \\
\mathbf{W}_{yh} \in \mathbb{R}^{N\times L} \text{ – weight matrix connecting the output layer at time } t-1 \text{ to the hidden layer,} \\
\mathbf{b}_h \text{ – bias vector for hidden states,} \\
\mathbf{b}_y \text{ – bias vector for outputs,} \\
f(\cdot) \text{ – hidden layer activation function,} \\
g(\cdot) \text{ – output layer activation function.}
\end{cases}$$

The term $\mathbf{W}_{yh}\mathbf{y}_{t-1}$ in Formula (2.30) is often omitted. For training RNN models, a *back-propagation-through-time* (BPTT) learning algorithm is typically used [Werbos, 1990]. However, in practice, training RNNs to learn long-term temporal dependencies can be difficult due to the vanishing and exploding gradient problems [Bengio et al., 1994].

2.4.4 Bidirectional recurrent neural network (BRNN)

In many applications the output prediction of $\mathbf{y}_t$ may depend not only on the information about the past sequence $\{\mathbf{o}_1,\ldots,\mathbf{o}_{t-1}\}$, but also on the future sequence $\{\mathbf{o}_{t+1},\ldots,\mathbf{o}_T\}$. In order to capture information from the whole input sequence, the *bidirectional* RNN (BRNN) architecture was proposed [Schuster and Paliwal, 1997]. In BRNNs, data are processed in two directions with two hidden layers, which are then inputted further to the same output layer. As shown in Figure 2.6, a recurrent *forward* hidden layer of BRNN $\overrightarrow{\mathbf{h}}$ computes sequence of hidden outputs for $t=1,\ldots,T$, and an additional recurrent layer $\overleftarrow{\mathbf{h}}$ computes the *backward* sequence of hidden outputs for $t=T,\ldots,1$:

$$\begin{cases}
\overrightarrow{\mathbf{h}}_t = f(\mathbf{W}_{o\overrightarrow{h}}\mathbf{o}_t + \mathbf{W}_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\overrightarrow{h}}), \\
\overleftarrow{\mathbf{h}}_t = f(\mathbf{W}_{o\overleftarrow{h}}\mathbf{o}_t + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\mathbf{h}}_{t-1} + \mathbf{b}_{\overleftarrow{h}}), \\
\mathbf{y}_t = g(\mathbf{W}_{\overrightarrow{h}y}\overrightarrow{\mathbf{h}}_t + \mathbf{W}_{\overleftarrow{h}y}\overleftarrow{\mathbf{h}}_t + \mathbf{b}_y).
\end{cases} \tag{2.31}$$

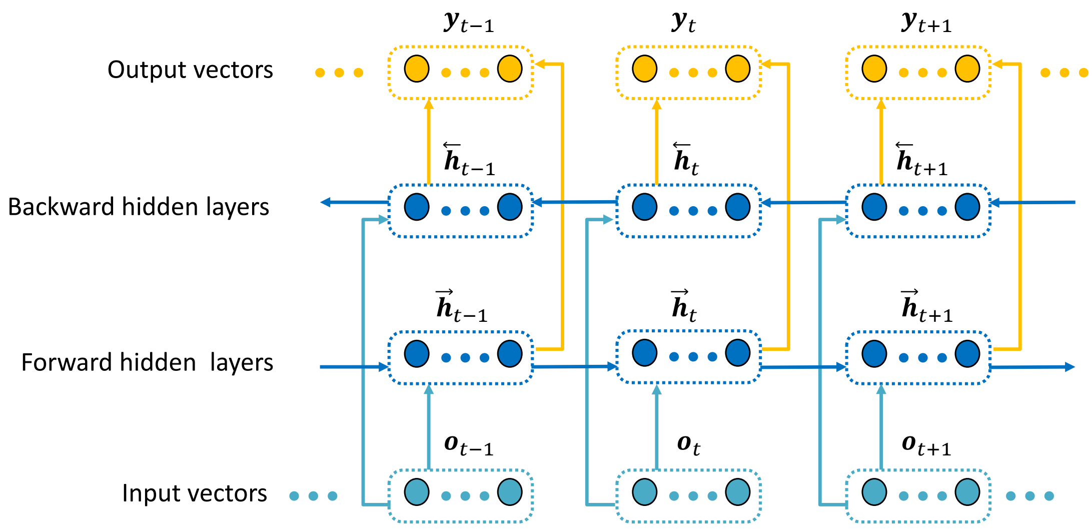


Figure 2.6 Bidirectional recurrent neural network

BRNNs become very successful in many applications [Graves, 2012], such as handwriting recognition [Graves and Schmidhuber, 2009] and ASR [Graves et al., 2013]. In ASR domain, BRNN architecture underlies many *end-to-end* systems [Graves and Jaitly, 2014; Graves et al., 2013; Hannun et al., 2014; Miao et al., 2015a].

2.4.5 Long short term memory (LSTM) network

Long short term memory (LSTM) neural networks represent a special type of RNNs, which are able to learn long-term dependencies. They were introduced in [Hochreiter and Schmidhuber, 1997] and explicitly designed to avoid the long-term dependency problem. The LSTM models have shown themselves to be extremely effective in many tasks, such as handwriting recognition [Graves et al., 2009; Pham et al., 2014]; machine translation [Sutskever et al., 2014]; speech synthesis [Fan et al., 2014]; visual recognition and description [Donahue et al., 2015]; speech recognition [Graves et al., 2013; Sak et al., 2014] and many others [Greff et al., 2016].

A schematic LSTM block diagram is illustrated in Figure 2.7. Here c_t is the memory *cell state* at time moment t. The LSTM has the ability to remove or add information to the cell state, regulated by structures called *gates*. Gates provide a mechanism to optionally let information through. They are composed of a sigmoid neural net layer and a pointwise multiplication operation. To protect and control the cell state, the LSTM has three gates:

- *Forget gate* $\mathbf{f}_t$ decides what information is thrown away from the cell state.

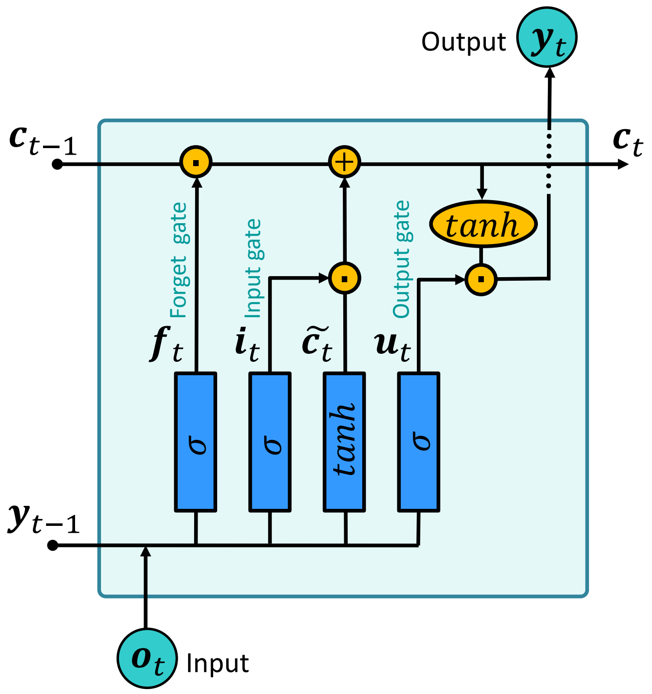


Figure 2.7 Example of the LSTM memory cell

- *Input gate* $\mathbf{i}_t$ – decides which values are updated.
- *Output gate* $\mathbf{u}_t$ – decides what parts of the cell state are outputted.

The computations at time t are described as follows:

$$\begin{cases} \mathbf{i}_t = \sigma\left(\mathbf{W}_{io}\mathbf{o}_t + \mathbf{W}_{iy}\mathbf{y}_{t-1} + \mathbf{b}_i\right), \\ \mathbf{f}_t = \sigma\left(\mathbf{W}_{fo}\mathbf{o}_t + \mathbf{W}_{fy}\mathbf{y}_{t-1} + \mathbf{b}_f\right), \\ \tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_{co}\mathbf{o}_t + \mathbf{W}_{cy}\mathbf{y}_{t-1} + \mathbf{b}_c\right), \\ \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \\ \mathbf{u}_t = \sigma\left(\mathbf{W}_{uo}\mathbf{o}_t + \mathbf{W}_{uy}\mathbf{y}_{t-1} + \mathbf{b}_u\right), \\ \mathbf{y}_t = \mathbf{u}_t \odot \tanh(\mathbf{c}_t), \end{cases} \tag{2.32}$$

where $\mathbf{W}_{.o}$ terms denote the weight matrices connecting the inputs with the units; $\mathbf{W}_{.y}$ terms denote the weight matrices connecting the memory cell outputs from the previous time moment $t-1$ with the units. The operation $\odot$ denotes the element-wise multiplication of two vectors.

There are many variants of the original LSTM model, which were proposed in the literature for different tasks [Grézl et al., 2007; Sak et al., 2014; Yao et al., 2015]. In one popular LSTM variant, introduced in [Gers and Schmidhuber, 2000], peephole connections from the cell to the gates were added to the neural network architecture. This allows the neural network to learn exact timings. Another alternative structure is the *gated recurrent unit* (GRU), proposed in [Cho et al., 2014]. It consists in simplification of the LSTM architecture. In GRU, the forget and input gates are combined into a single *update gate*. In addition, the cell state and hidden state are merged, and some other modifications are made.

Chapter 3

Overview of automatic speech recognition

This chapter provides an overview of ASR systems with the focus on hidden Markov model based ASR systems and describes their principal concepts.

3.1 Introduction to automatic speech recognition

The development of ASR has a long history — since 1960 till this moment – and during this period has achieved a great progress. The success in ASR research has led to the increase of a variety of real-world applications, such as interactive spoken dialog systems, information extraction and retrieval, automatic closed captioning, dictation, transcription of recorded speech, language learning systems, etc.

The goal of ASR consists in finding a text representation for an input speech signal. In a *statistical speech recognition paradigm*, which is the most common one for this problem nowadays [Benesty et al., 2007; Rabiner, 1989], an ASR system aims to find word sequence $\mathbf{W} = (w_1, \dots, w_N)$, which is the most probable according to the trained model, for given acoustic observation feature vectors $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]$.

The principal components of an ASR system with an example of speech recognition analysis are illustrated in Figure 3.1. First, an input speech waveform is converted into a sequence of fixed-size acoustic feature vectors $\mathbf{O} = [\mathbf{o}_1, \dots, \mathbf{o}_T]$ in a process called *feature extraction*. Then, the *decoder* tries to find the most likely sequence of words for the obtained feature vectors. The ASR task can be formulated as finding word sequence $\mathbf{W}^*$ out of all possible word hypotheses, that maximizes the posterior probability of $\mathbf{O}$:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{O}). \tag{3.1}$$

However, since $P(\mathbf{W}|\mathbf{O})$ is difficult to model directly, Bayes theorem is usually applied [Jelinek, 1976]:

$$P(\mathbf{W}|\mathbf{O}) = \frac{P(\mathbf{O}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{O})} \propto P(\mathbf{O}|\mathbf{W})P(\mathbf{W}) \tag{3.2}$$

to transform into the equivalent problem :

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{O}|\mathbf{W})P(\mathbf{W}). \tag{3.3}$$

The likelihood $P(\mathbf{O}|\mathbf{W})$ is determined by an *acoustic model* and the prior $P(\mathbf{W})$ – by a *language model*.

As shown in Figure 3.1 each word w_n in $\mathbf{W}$ is represented as a sequence of basic sounds called *phones*. To take into account different pronunciation variations (*phonetic transcriptions* $\mathbf{Q}$), the likelihood $P(\mathbf{O}|\mathbf{W})$ can be estimated as

$$P(\mathbf{O}|\mathbf{W}) = \sum_{\mathbf{Q}} P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q}|\mathbf{W}), \tag{3.4}$$

where $\mathbf{Q}$ is a sequence of word pronunciations $\mathbf{Q} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_N)$ for the word sequence $\mathbf{W}$, and each word pronunciation is a sequence of phones: $\mathbf{Q}_n = (q_1^{(n)}, q_2^{(n)} \ldots)$, $1 \leq n \leq N$. Then

$$P(\mathbf{Q}|\mathbf{W}) = \prod_{n=1}^{N} P(\mathbf{Q}_n|w_n), \tag{3.5}$$

where $P(\mathbf{Q}_n|w_n)$ is the probability that the word w_n is pronounced with phonetic transcription $\mathbf{Q}_n$. Hence $P(\mathbf{Q}|\mathbf{W})$ corresponds to *pronunciation model* and gives the probability of the phonetic sequence given the sequence of words. The pronunciation model is also called *(phonetic) dictionary*, or *lexicon*. Taking into account all three factors (acoustic, phonetic and language) and the corresponding models, Formula (3.3) can be rewritten as follows:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \sum_{\mathbf{Q}} P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q}|\mathbf{W})P(\mathbf{W}). \tag{3.6}$$

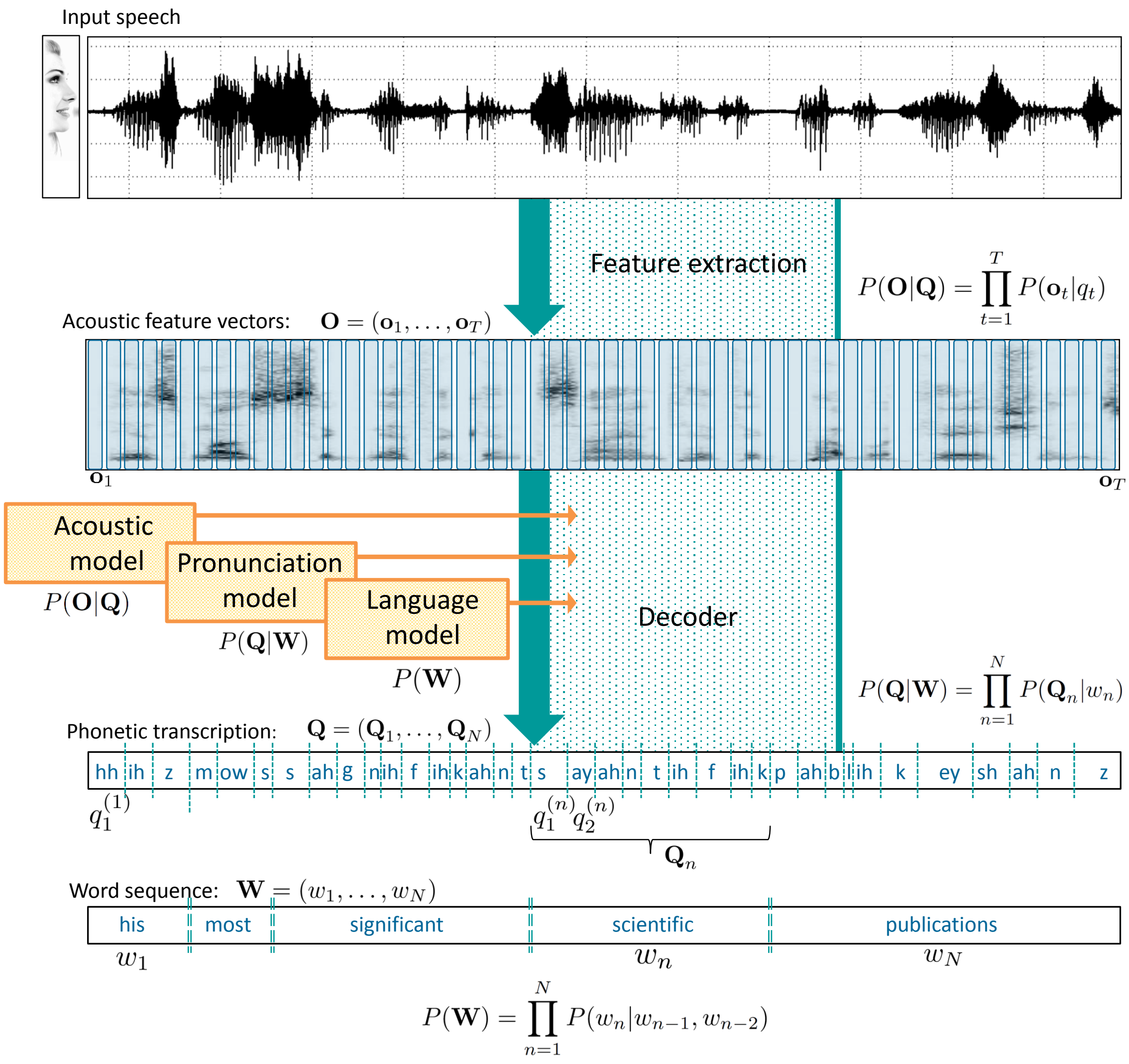

Input speech
Feature extraction
$P(\mathbf{O}|\mathbf{Q}) = \prod_{t=1}^{T} P(\mathbf{o}_t|q_t)$
Acoustic feature vectors: $\mathbf{O} = (\mathbf{o}_1, \ldots, \mathbf{o}_T)$
$\mathbf{o}_1$
$\mathbf{o}_T$
Acoustic model
$P(\mathbf{O}|\mathbf{Q})$
Pronunciation model
$P(\mathbf{Q}|\mathbf{W})$
Language model
$P(\mathbf{W})$
Decoder
$P(\mathbf{Q}|\mathbf{W}) = \prod_{n=1}^{N} P(\mathbf{Q}_n|w_n)$
Phonetic transcription: $\mathbf{Q} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_N)$
hh ih z m ow s s ah g n ih f ih k ah n t s ay ah n t ih f ih k p ah b l ih k ey sh ah n z
$q_1^{(1)}$
$q_1^{(n)} q_2^{(n)}$
$\mathbf{Q}_n$
Word sequence: $\mathbf{W} = (w_1, \ldots, w_N)$
his
most
significant
scientific
publications
w_1
w_n
w_N
$P(\mathbf{W}) = \prod_{n=1}^{N} P(w_n|w_{n-1}, w_{n-2})$

Figure 3.1 Scheme of an ASR system and analysis of an example phrase *"his most significant scientific publications"*

In our example presented in Figure 3.1 the phrase **W**= *(his, most, significant, scientific, publications)* corresponds to the phonetic transcription **Q**=*((hh ih z), (m ow s), (s ah g n ih f ih k ah n t), (s ay ah n t ih f ih k), (p ah b l ih k ey sh ah n z))*.

3.2 Feature extraction

A feature extraction (also referred to as acoustic *front-end analysis*) module is an important component of any ASR system. The purpose of the feature extraction stage is to provide a compact encoding of the speech waveform signal to be used further in the ASR system.

Speech data contains information about different aspects of the signal besides the pronounced word sequence. These aspects include language and dialect, environmental characteristics (recording channel, background noises), speaker's personal characteristics (gender, emotional state, age) and others. One of the main goals of the feature extraction module is to remove the information irrelevant for ASR from the speech signal while preserving the essential information about the spoken content. Another goal is to reduce the dimension of the obtained feature vectors and provide the optimal representation of information for the given *acoustic model* (AM).

Typically, the waveform file format is used for storing speech data. The continuous speech signal is transformed into a sequence of samples (or the discrete-time signal). Signal sampling can be performed at different sampling rates (usually 8-44 kHz) depending on the conditions, task and recording channel. Speech signal is divided into overlapping segments and feature vectors are computed usually every 10-15 ms using an overlapping analysis window of about 25 ms. Speech signal is supposed to be stationary in these segments (*speech frames*). Usually each speech frame is represented by a single parameter feature vector.

One of the simplest and most widely used speech encoding techniques is based on Mel-frequency cepstral coefficients (MFCCs) extraction [Davis and Mermelstein, 1980]. Also there have been various other types of features proposed for the speech recognition problem, which aim to incorporate new information into speech recognition system: perceptual linear prediction (PLP) [Hermansky, 1990], temporal patterns (TRAPS) features [Hermansky and Sharma, 1999], discriminatively trained features [Povey et al., 2005] with minimum phone error objective function (fMPE features), and many others. Features generated by neural networks, such as tandem features [Hermansky et al., 2000], features with split left and right temporal contexts (LC-RC) [Schwarz, 2009], and bottleneck (BN) features [Grézl et al., 2007; Yu and Seltzer, 2011] have shown to be very effective in many ASR systems.

Usually ASR systems with GMM-HMM acoustic models do not use filter-bank coefficients (*fbanks*) as the input representation because they are strongly correlated, and to model them well many parameters (a lot of diagonal Gaussians or full covariance Gaussians) are required [Hinton et al., 2012a]. However, for neural network acoustic models, raw filter-bank features sometimes are more preferable than MFCCs or others [Deng et al., 2013b; Mohamed et al., 2012].

Features, obtained with a GMM were used in [Pinto and Hermansky, 2008] for training a phoneme multilayer perceptron recognizer. This type of features is close to the one which will be used in this thesis. Augmenting the basic input features with some additional features, for example, pitch features [Ghahremani et al., 2014; Metze et al., 2013] is a common technique to improve ASR performance.

In the next two sections, we describe the most commonly used in ASR types of features from two categories: spectral features and neural network-based features.

3.2.1 Spectral features

The general scheme for feature extraction can be described as follows.

- *Spectrum computation.* Short time fast Fourier transform (FFT) is calculated for each speech frame.

- *Auditory-like modifications.* Modifications, which are motivated by psycho-physical acoustic phenomena and human perception characteristics [Moore, 1997], are performed for the obtained spectra.

  For example, in case of MFCC features, to take into account different sensitivity of human hearing system to different frequencies, the powers of the obtained spectra are mapped into the *Mel scale*. In addition, since the human perception of loudness is logarithmic, all the Mel power-spectra values are transformed into logarithmic scale. The obtained features are referred to as *mel-filter bank (fbank)* features, and sometimes are used in this thesis to train DNN AMs.

- *Decorrelation.* In case of MFCCs, decorrelation of the obtained features is done by *discrete cosine transform* (DCT).

- *Derivatives.* The resulting feature vectors are often appended with their first and second order temporal derivatives (delta (Δ) and acceleration ($\Delta\Delta$) coefficients).

MFCCs are one of the most widely used input features for ASR systems. Another popular features are PLP [Hermansky, 1990]. In PLP feature extraction, the *Bark scale* is used (instead of the Mel scale) to compute the filter-bank filters. Then this process is followed by a linear predictive analysis, from which a cepstral representation is derived. Both, MFCC and PLP features are calculated with Δ and $\Delta\Delta$ derivatives and usually have the same dimension (39). It was shown [Dave, 2013; Milner, 2002; Psutka et al., 2001] that PLP features can provide comparable results to MFCCs, and the superiority of one feature type over another depends on the database characteristics, environmental conditions, or system configurations.

3.2.2 Neural network-based features

Neural network models are often used as feature extractors in ASR systems. Features generated by these models can further be used to train other classifiers for ASR, such as GMMs or DNNs. Two important types of neural network-based features, tandem and bottle-neck features, are described below.

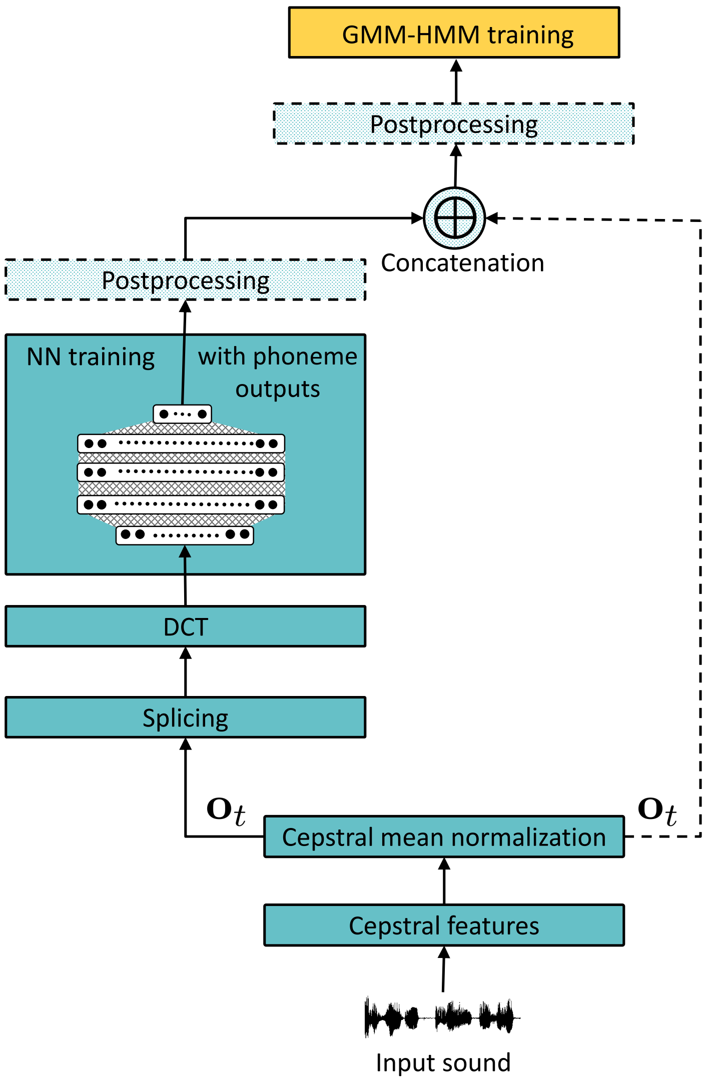


Figure 3.2 Example of the tandem speech recognition system

3.2.2.1 Tandem features

In the *tandem approach* [Ellis et al., 2001; Hermansky et al., 2000; Morgan, 2012] a neural network, trained with phones as targets, is used to generate input features fed to a conventional GMM-HMM model. These features can be fed directly to a GMM model after some post-processing steps (decorrelation and dimensionality reduction, for example, using principal component analysis (PCA), DCT, heteroscedastic linear discriminant analysis (HLDA) [Kumar and Andreou, 1998]) as in the originally proposed approach [Hermansky et al., 2000], or appended with other features, for example, with some spectral features before training a GMM model. An example of the tandem speech recognition system is illustrated in Figure 3.2. The dashed lines in the figure correspond to the stages of the feature extraction process that can be omitted.

In this thesis we propose and investigate the approach which is in some sense inverse to the tandem approach — features, derived from GMM models, are used as input representation for training DNN models.

3.2.2.2 Bottleneck features

Bottleneck (BN) features [Grézl et al., 2007; Yu and Seltzer, 2011] have shown to be effective in improving the accuracy of ASR systems for both DNN-HMM and GMM-HMM acoustic models. Conventionally, bottleneck features are extracted from a DNN, which is trained to predict context-independent monophone states or context-dependent triphone states. In [Yu and Seltzer, 2011] it was shown that the use of context-dependent triphone states gives better accuracy for the final ASR system.

One of the hidden layers in a DNN, trained for BN feature extraction, has a relatively small dimension (35-80 units). This layer is referred to as a BN layer. The part of the DNN, following the BN layer, is removed from the DNN structure after the training, and the outputs from the BN layer are used as features directly or with some post processing (splicing, concatenation with other features and other modifications) to train DNN-HMM or GMM-HMM models. An example of using BN features for training a GMM-HMM acoustic model is shown in Figure 3.3.

An important property of bottleneck features — the capability of transferring the learned representation to unseen languages [Vu et al., 2014] — makes them (and some of their advanced versions, such as *stacked BN* (SBN) features [Grézl et al., 2014]) very popular in multilingual training architectures, especially for low-resource ASR systems [Bell et al., 2014; Li et al., 2014b; Müller et al., 2014].

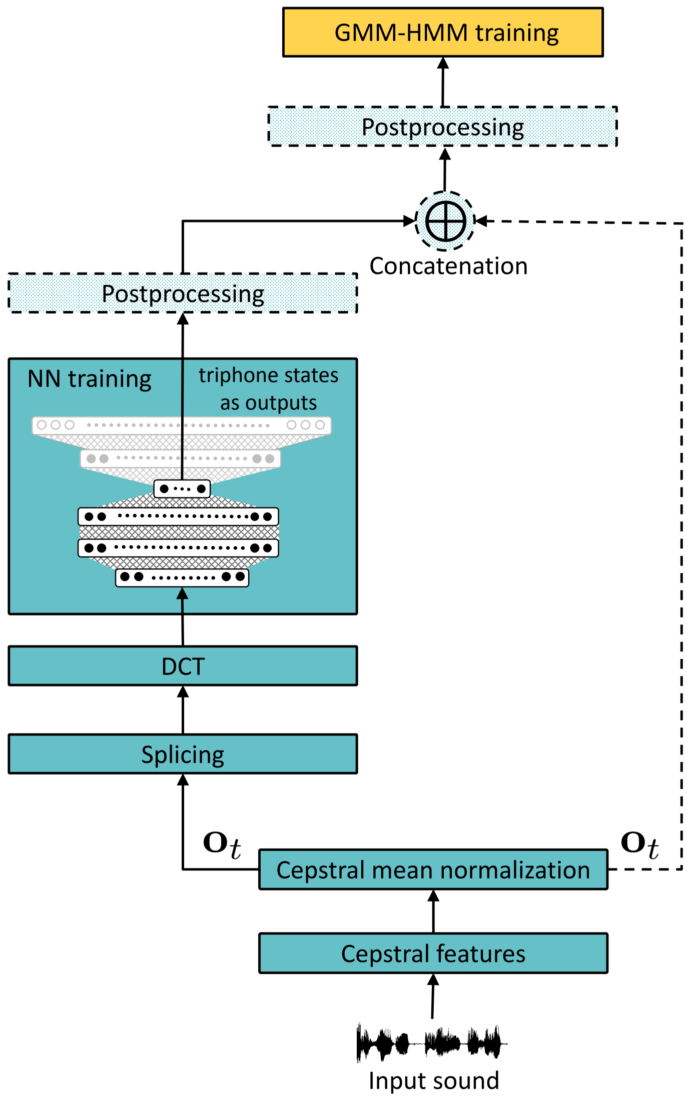


Figure 3.3 Bottleneck (BN) features for GMM-HMM speech recognition system

3.3 Acoustic modeling

Statistical approach with hidden Markov models (HMMs) [Rabiner, 1989] is conventional in state-of-the-art ASR systems. As mentioned above (see Figure 3.1), any spoken word w is represented with a sequences of basic sounds (units). Each basic unit is modeled with a corresponding HMM.

The simplest example of a unit is a phoneme. The same phonemes may have different pronunciations depending on their context. To take into account the effects caused by coarticalation and reduction, usually *context-dependent* (CD) *triphone* units (phonemes with given left and right neighbor phonemes) are used for acoustic modeling. When phones are modeled without considering the context they are referred to as *context-independent* (CI) or *monophone* units.

Introducing triphones into the ASR system significantly increases the number of parameters to be estimated for this system. Some triphones may be missing in the training data sets (so-called *unseen triphones*) or have insufficient number of realizations to model them accurately. To avoid this data sparsity problem, clustering algorithms, such as *data-driven clustering* and *tree-based clustering*, are commonly used [Hwang et al., 1996; Young et al., 2002, 1994] for triphone state clustering. They tie the parameters of those triphone states that are close acoustically.

3.3.1 Hidden Markov models

An HMM is a stochastic Markov process with hidden (unobserved) states. The hidden states can be observed indirectly through another stochastic process (or several stochastic processes) that outputs the sequences of observed symbols. A detailed introduction to HMMs for speech processing is given in [Rabiner and Juang, 1986].

In ASR systems, each unit (triphone or monophone) is modeled with a continuous density HMM, as illustrated in Figure 3.4 for a sequence of observation vectors $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots)$. An HMM model is characterized by the following set of parameters:

- *Number of states* N;
- *State transition probability distributions* a_{ij}, denoted by matrix $\mathbf{A} = \{a_{ij}\}$:

  $$a_{ij} = P(s_t = j | s_{t-1} = i), \tag{3.7}$$

  where

  $$\sum_{j=1}^{N} a_{ij} = 1, \forall i = 1, \ldots, N. \tag{3.8}$$

- *State output (emission) probability distributions* $\{b_j(.)\}$, denoted by $\mathbf{B} = \{b_j(.)\}$, where $b_j(.)$ is a probability distribution of state j, and $b_j(\mathbf{o}_t)$ denotes the likelihood of state j generating observation vector $\mathbf{o}_t$ at time t:

  $$b_j(\mathbf{o}_t) = P(\mathbf{o}_t | s_t = j) \tag{3.9}$$

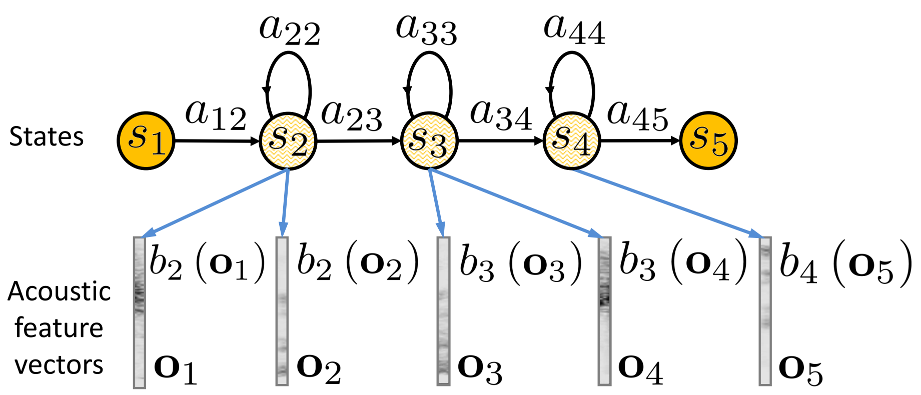


Figure 3.4 HMM-based phone model with five states and left-to-right topology. The first and the final states are *non-emitting*, other states have output distributions $b_j(.)$. Transition parameters a_{ij} model the transition probabilities between hidden states i and j.

- *Initial state probability distribution*. The initial probability π_i for state s_i is $\pi_i = P(s_0 = i)$, and $\sum_{i=1}^{N} \pi_i = 1$. In practice and further in this thesis, initial distribution is excluded from the analysis.

Hence an HMM is represented with model parameters $\mathbf{\Lambda} = \{\mathbf{A}, \mathbf{B}\} = \left\{\{a_{ij}\}, \{b_j(.)\}\right\}$. Modeling of output probability distributions $b_j(.)$ depends on the type of acoustic model (see Section 3.3.2).

Due to computational reasons, two important assumptions for HMM in ASR systems are usually made (except for some particular examples, such as graphical models [Bilmes and Bartels, 2005]):

1. *A first order Markov process assumption*, which means that the stochastic process satisfies the Markov property, so that its state s_t at time t depends only on the state s_{t-1} at the previous moment of time:

$$P(s_t|s_{t-1}, s_{t-2}, \ldots, s_1) = P(s_t|s_{t-1}). \tag{3.10}$$

2. *Observation independence assumption*, in which observation feature vectors $\mathbf{o}_t$ are assumed to be conditionally independent of the previous observations and states, given state s_t:

$$P(\mathbf{o}_t|\mathbf{o}_{t-1}, \ldots, \mathbf{o}_1; s_t, \ldots, s_1) = P(\mathbf{o}_t|s_t). \tag{3.11}$$

The analysis of conditions where the standard model assumptions deviate from real speech data is given in [Gillick et al., 2011]. These assumptions allow us to simplify the calculation of acoustic likelihood for a given sequence of observations $\mathbf{O}$ and states $\mathbf{S}$ as

$$P(\mathbf{O}|w;\mathbf{\Lambda}) = \sum_{\mathbf{S}\in w}\prod_{t=1}^{T} P(\mathbf{o}_t|s_t;\mathbf{\Lambda})P(s_t|s_{t-1};\mathbf{\Lambda}) = \sum_{\mathbf{S}\in w} a_{q_0,q_1}\prod_{t=1}^{T} b_{q_t}(\mathbf{o}_t)a_{q_t,q_{t+1}}. \tag{3.12}$$

In practice, the calculation of the likelihoods directly from Formula (3.12) is too computationally expensive and requires $O(N^T)$ multiplications and summations, hence special algorithms were developed for this purpose. The *Baum-Welch forward-backward algorithm* [Baum et al., 1967] reduces the complexity to $O(N^2T)$ steps. Another widely-used solution is the *Viterbi algorithm* [Viterbi, 1967], which finds only the most likely state sequence (using $\max_{\mathbf{S}\in w}$ operation instead of $\sum_{\mathbf{S}\in w}$ in Formula (3.12)).

3.3.2 State distribution modeling

In this section we consider some of the most common approaches for modeling state probability distributions in HMMs.

3.3.2.1 GMM-HMM

The *Gaussian mixture models* (GMMs) approach has been one of the most common technique to model state probability distributions (see Formula (3.9)) in ASR systems for many decades. In a GMM-HMM model, each state j is modeled as a weighted sum of M_j Gaussians:

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t|j) = \sum_{m=1}^{M_j} \omega_{jm}\mathcal{N}\left(\mathbf{o}_t;\boldsymbol{\mu}_{jm},\mathbf{\Sigma}_{jm}\right), \tag{3.13}$$

where ω_{jm} is the weight of the m'th component in the mixture for state j such that:

$$\omega_{jm} \geq 0 \text{ and } \sum_{m=1}^{M_j} \omega_{jm} = 1. \tag{3.14}$$

In Formula (3.13), $\mathcal{N}(.;\boldsymbol{\mu},\mathbf{\Sigma})$ is a multivariative Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$:

$$\mathcal{N}(\mathbf{o};\boldsymbol{\mu},\mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\mathbf{\Sigma}|}}\exp\left\{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{o}-\boldsymbol{\mu})\right\}, \tag{3.15}$$

where d is the dimensionality of acoustic feature vector: $\mathbf{o} \in \mathbb{R}^d$.

3.3.2.2 Subspace Gaussian mixture models

Subspace Gaussian mixture models (SGMMs) have been proposed in [Burget et al., 2010; Povey et al., 2010, 2011a] as an alternative approach to standard GMM-HMMs for acoustic modeling. All context-dependent HMM states in SGMMs share a common representation, based on the *universal background model* (UBM)[1]. The covariance matrices are shared between states. The mean vectors and mixture weights for each state are specified by a corresponding vector $\mathbf{v}_j \in \mathbb{R}^S$, where S is a subspace dimension (typically $S = 50$). In the simplest form SGMM can be expressed as follows [Povey et al., 2010]:

$$\begin{cases} b_j(\mathbf{o}_t) = P(\mathbf{o}_t|j) = \sum_{i=1}^{I} \omega_{ji} \mathcal{N}\left(\mathbf{o}_t; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_i\right), \\ \boldsymbol{\mu}_{ji} = \mathbf{M}_i \mathbf{v}_j, \\ \omega_{ji} = \frac{\exp \omega_i^T \mathbf{v}_j}{\sum_{i'=1}^{I} \exp \omega_{i'}^T \mathbf{v}_j}, \end{cases} \tag{3.16}$$

where $j \in \{1, \ldots, J\}$ is a context-dependent speech state, modeled with a GMM with I Gaussians (usually $I \in [200, 2000]$) and covariance matrices $\boldsymbol{\Sigma}_i$ which are shared between states, mixture weights ω_{ji} and means $\boldsymbol{\mu}_{ji}$. The description of model extensions and developed training procedures can be found in [Povey et al., 2011a].

The strengths of SGMMs lie in their compactness. These models require less data for training in comparison with GMM models to achieve a comparable accuracy and have been widely used for low-resource and multi-lingual ASR systems [Burget et al., 2010; Lu et al., 2011]. The weakness of SGMMs consists in their complexity of implementation in comparison with conventional GMMs; we will not consider these models further in the thesis.

3.3.2.3 DNN-HMM

The use of neural networks is an alternative approach for AM training, proposed in [Bourlard and Wellekens, 1990; Morgan and Bourlard, 1990], where multilayer perceptions (MLP) (feed-forward *artificial neural networks* (ANN)) were used to estimate emission probabilities

[1] *Universal background model* (UBM) is a GMM model trained on all speech classes pooled together [Povey et al., 2010]. The UBM is a popular approach to alternative hypothesis modeling in speaker recognition domain, where a GMM-UBM represents a large mixture of Gaussians that cover all speech, and in the context of speaker recognition, it is trained using the EM algorithm on data from multiple speakers, and then adapted to each speaker using maximum a posteriori (MAP) estimation [Bimbot et al., 2004; Reynolds et al., 2000].

for HMMs. In the DNN-HMM approach state probabilities in HMMs are estimated with a neural network model.

One of the commonly used approaches in state-of-the-art ASR systems is a *hybrid* DNN-HMM approach, where the dynamics of speech signal is modeled with HMMs, and the state observation probabilities are estimated with DNN models. In the original approach [Bourlard and Wellekens, 1990] the MLP outputs were used directly for HMMs. But later the need for scaling the obtained probabilities with state priors was shown. The output posterior probabilities from a neural network are scaled using prior class probabilities estimated from the training data, to obtain scaled likelihoods [Bourlard et al., 1992].

In a DNN-HMM system, outputs of a DNN are the state posteriors $P(s_i|\mathbf{o}_t)$, which are transformed for decoding into pseudo (or scaled) log-likelihoods as follows:

$$\log P(\mathbf{o}_t \mid s_i) = \log \frac{P(s_i \mid \mathbf{o}_t)P(\mathbf{o}_t)}{P(s_i)} \propto \log P(s_i \mid \mathbf{o}_t) - \log P(s_i), \tag{3.17}$$

where the state prior $P(s_i)$ can be estimated from the state-level forced alignment on the training speech data, and probability $P(\mathbf{o}_t)$ of the acoustic observation vector $\mathbf{o}_t$ is independent on the HMM state and can be omitted during the decoding process.

Typically states of context-dependent (CD) phonemes (triphones) are used as basic classes for DNN outputs. However the earlier hybrid ASR systems use context-independent (CI) phones (monophones) states. The first attempts to incorporate CD triphones into hybrid architectures [Bourlard et al., 1992] were based on modelling the context-dependency using factorization:

$$P(s_i, c_j|\mathbf{o}_t) = P(s_i|\mathbf{o}_t)P(c_i|s_j, \mathbf{o}_t) = P(c_i|\mathbf{o}_t)P(s_i|c_j, \mathbf{o}_t), \tag{3.18}$$

where $c_j \in \{c_1, \dots, c_J\}$ is one of the clustered context classes, and s_j is either a CD phone or a state in a CD phone. ANNs were used to estimate both $P(s_i|\mathbf{o}_t)$ (or $P(c_i|\mathbf{o}_t)$) and $P(c_i|s_j, \mathbf{o}_t)$ (or $P(s_i|c_j, \mathbf{o}_t)$) probabilities. These types of CD DNN-HMM models provided comparatively small improvement over the GMM-HMM models.

The more efficient technique for CD state modeling [Dahl et al., 2012] uses a DNN model to directly estimate the posterior distribution of tied triphone HMM states. These states are usually obtained from a conventional GMM-HMM system through a state clustering procedure, however there are some works that propose a GMM-free training procedure [Bacchiani et al., 2014; Senior et al., 2014; Zhu et al., 2015]. A DNN-HMM model is illustrated in Figure 3.5.

The recent technical progress in computational hardware allowed to overcome some limitations of the first hybrid ASR systems. The shallow neural networks have been replaced with the deep neural architectures, with many hidden layers. This type of acoustic models is referred to as CD-DNN-HMM models in the literature. Modern DNN-HMM AMs have been demonstrated to significantly outperform GMM-HMM systems on various ASR tasks [Dahl et al., 2011, 2012; Hinton et al., 2012a; Kingsbury et al., 2012; Seide et al., 2011b; Yu et al., 2010]. A review of recent developments in DNN-HMM framework and in deep learning approach to ASR is given in [Li et al., 2015a; Trentin and Gori, 2001; Yu and Deng, 2014].

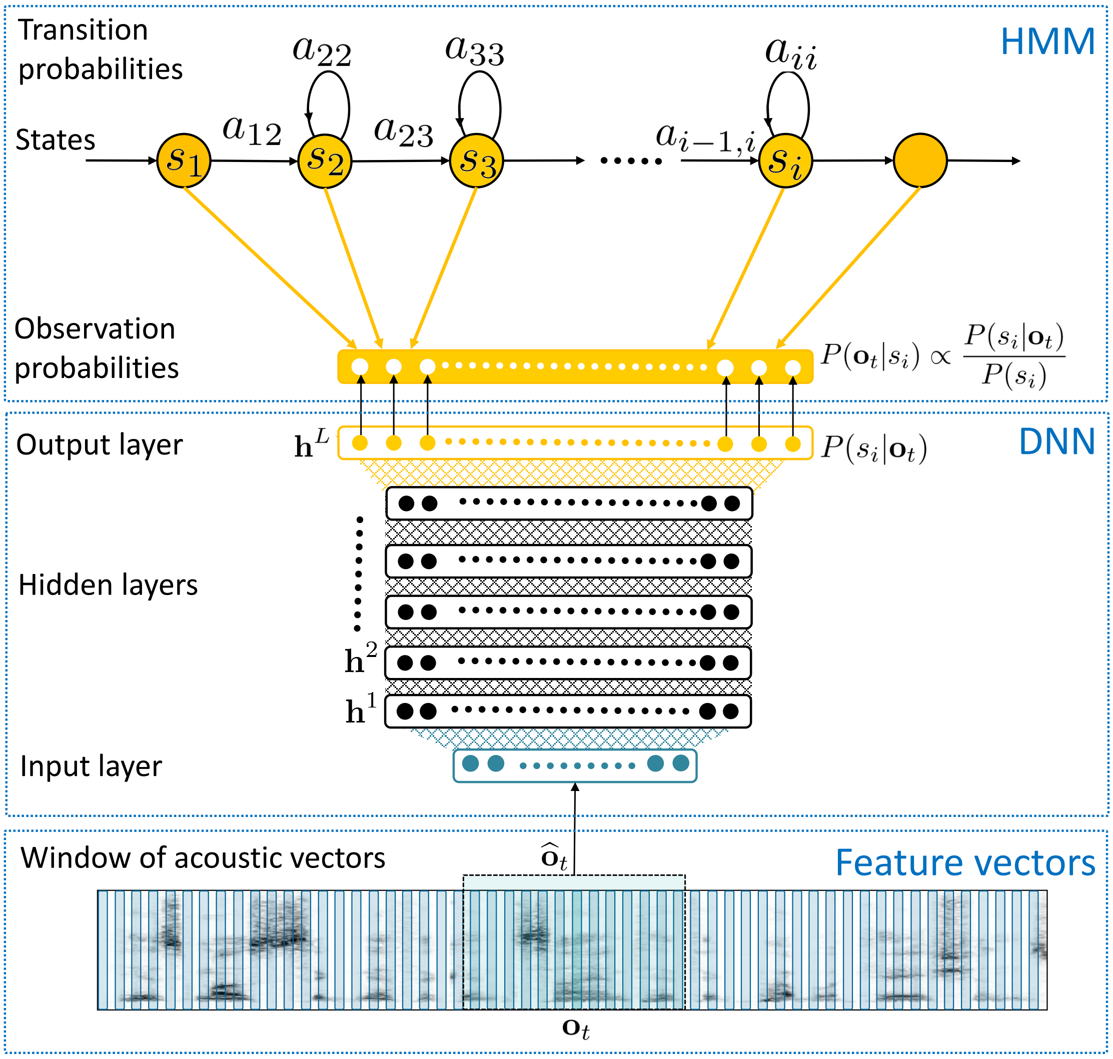


Figure 3.5 Example of the hybrid acoustic model using a DNN to estimate posterior probabilities for HMM states

3.3.3 HMM training and parameter estimation

3.3.3.1 GMM-HMM

The acoustic model parameters $\mathbf{\Lambda} = \big\{\{a_{ij}\},\{b_j(.)\}\big\}$ can be estimated from a training corpus using *expectation maximization* (EM) algorithm [Dempster et al., 1977]. In a general case, HMMs are trained using a chosen *training criterion*, which corresponds to a particular *objective function*. An objective function $\mathcal{F}(\mathbf{\Lambda}) = \mathcal{F}(\mathbf{\Lambda}; \mathcal{O})$ is optimized (minimized or maximized, depending on the training criterion) during the training. Here $\mathcal{O} = (\mathcal{O}_1, \dots, \mathcal{O}_R)$ is the set of training sentences (acoustic sequences).

One of the standard ways to train HMM models is by optimizing *maximum likelihood* (ML) objective function:

$$\mathcal{F}_{ML}(\mathbf{\Lambda}) = \sum_{r=1}^{R} \log p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi_r), \tag{3.19}$$

where $\phi_r = \phi(\mathbf{W}_r)$ is a composite HMM model, corresponding to the (correct) transcription of the training sentence $\mathcal{O}_r$ and $\mathbf{W}_r$ is a sequence of words in this transcription. This objective function can be maximized by using the EM algorithm [Dempster et al., 1977], known as the Baum-Welch algorithm [Baum et al., 1967; Juang et al., 1986].

One of the main problem with the ML training criterion is that it fits to the training data, but does not take into account the ability of the model to discriminate. To eliminate this problem, alternative approaches with *discriminative training criteria* [Schlüter et al., 2001; Vertanen, 2004] have been proposed.

In one of the first approaches, developed in the discriminative training framework, parameters are trained to *maximize the mutual information* (MMI) between an acoustic observation sequence and the corresponding word sequence [Bahl et al., 1986; Povey, 2004; Povey and Woodland, 2001; Schluter and Macherey, 1998; Schlüter et al., 1999] with the following objective function:

$$\mathcal{F}_{MMI}(\mathbf{\Lambda}) = \sum_{r=1}^{R} \log \frac{p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi_r)P(\phi_r)}{\sum_{\phi} p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi)P(\phi)}, \tag{3.20}$$

where $P(\phi)$ is the language model probability for sentence $\mathcal{O}$. Here the numerator corresponds to the data given the correct word sequence $\mathbf{W}_r$, and the dominator corresponds to the total likelihood of the data given all possible word sequences $\mathbf{W}$. In practice, $\mathbf{W}$ is obtained from an ASR system (in the form of N-best lists or word lattices). Hence, the MMI criterion attempts not only to make the correct hypothesis more probable, but also to decrease the probability of incorrect alternatives.

Other approaches include *minimum phone error* (MPE) training [Povey and Woodland, 2002], which attempts to maximize the a posteriori sentence probability, scaled by the raw phone accuracy, and a *state-level minimum Bayes risk* (sMBR) criterion [Veselý et al., 2013]. These two criteria can be expressed as

$$\mathcal{F}_{MBR}(\mathbf{\Lambda}) = \sum_{r=1}^{R} \frac{\sum_{\phi} p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi)^k P(\phi)^k A(\phi,\phi_r)}{\sum_{\phi'} p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi)^k P(\phi')^k}, \tag{3.21}$$

where k is a scaling factor, and $A(\phi,\phi_r)$ is the raw accuracy of the sentence $\mathbf{W}$ with respect to the reference $\mathbf{W}_r$. Hence, $A(\phi,\phi_r)$ corresponds to the number of correct phone labels (in MPE) or state labels (in sMBR).

There are also many others [Kaiser et al., 2000; Povey et al., 2008] criteria, for example, *minimum classification error* (MCE) [Chou et al., 1993; Juang and Katagiri, 1992], which focuses on sentence-level accuracy, or *boosted MMI* (BMMI) criterion [Povey et al., 2008]. In the BMMI criterion [Povey et al., 2008] the MMI objective function (3.20) is modified to boost the likelihoods of paths that contain more errors, as follows:

$$\mathcal{F}_{BMMI}(\mathbf{\Lambda}) = \sum_{r=1}^{R} \log \frac{p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi_r)P(\phi_r)}{\sum_{\phi} p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi)P(\phi)e^{-bA(\phi,\phi_r)}}, \tag{3.22}$$

where b is the *boosting factor*.

In this work for training GMM-HMM models we will mostly use the ML and BMMI criterion. For DNN-HMM models we will apply the cross-entropy (CE) criterion (see Section 2.3) and, for some models, perform additional training with the sMBR criterion.

3.3.3.2 DNN-HMM

DNNs usually use a contextual window of several $(2T+1)$ frames as an input vector. As shown in Figure 3.5, the input vector for the DNN is vector $\widehat{\mathbf{o}}_t = [\mathbf{o}_{t-T},\ldots,\mathbf{o}_t,\ldots,\mathbf{o}_{t+T}]$.

In a CD-DNN-HMM system, a single DNN is used to model the conditional state posterior probability $P(s_i|\mathbf{o}_t)$ for all states, in contrast to a GMM-HMM system, where each state is modeled by a separate GMM model.

DNNs for AMs can be trained with a CE criterion (Formula (2.15)) using SGD algorithm, as described in Section 2.3. To obtain targets for each observation vector, the force-alignment of the training corpus can be produced by a GMM-HMM system, or with DNN alignments as in a GMM-free training procedure [Bacchiani et al., 2014; Senior et al., 2014].

The CE criterion operates on the frame level, so it is possible to improve AMs using discriminative criteria [Veselỳ et al., 2013], which operate on the sentence level, as described in Section 3.3.3.1. Similar to training GMM-HMM systems, before the sequence-discriminative training of DNNs the numerator and denominator lattices have to be generated. In [Povey et al., 2016] a sequence-discriminative training of neural network AMs using lattice-free version of the (MMI) criterion is performed without frame-level CE pre-training. In recent times, connectionist temporal classification (CTC) criterion [Graves et al., 2006], which allows to train a DNN without pre-generated frame-level alignment, has attracted a lot of interest in ASR research. We will consider this approach in Section 3.7.

3.4 Pronunciation modeling

The *pronunciation model* (or *lexicon*) defines the list of words with their phonetic transcriptions in terms of a set of context-independent phonemes. Some words may have multiple pronunciations and prior probabilities associated with their occurrence frequencies.

A pronunciation system usually contains a static word pronunciation dictionary, which can be created by experts or generated automatically. However, such a static dictionary may not cover all possible words in a language, that are required for a given ASR system. Hence, a dictionary is often supplemented with a *grapheme-to-phoneme* (G2P) converter. Many approaches for building a lexicon (or G2P converters) can be broadly classified into two classes [Bisani and Ney, 2008; Jouvet et al., 2012; Wester, 2003]:

- *Knowledge-based approach*, where different pronunciation variants are generated by using phonological rules of the given language [Kaplan and Kay, 1994]. Such rules are developed by experts and based on linguistic knowledge. They have to take into account many pronunciation exceptions that can occur in a language.

- *Data-driven approach*, in which phonetic pronunciations are automatically trained from a corpus of pronunciation examples. Different algorithms have been developed to train such G2P converters, including joint-sequence models [Bisani and Ney, 2008], conditional random field (CRF) models [Illina et al., 2011; Wang and King, 2011], the statistical machine translation approach [Laurent et al., 2009], approaches based on using neural networks [Rao et al., 2015; Yao and Zweig, 2015] and many others.

Often, and in this thesis also, all pronunciations of a given word are assumed to have equal prior probabilities.

3.5 Language modeling

Statistical language modeling aims to learn the joint probability function of word sequences in a given language. The *language model* (LM), which corresponds to $P(\mathbf{W})$ in Formula (3.3), estimates a probability distribution over the sequence of words: $\mathbf{W} = (w_1, \ldots, w_M)$. The prior probability of $\mathbf{W}$ is estimated as

$$P(\mathbf{W}) = \prod_{m=1}^{M} P(w_m | w_{m-1}, \ldots, w_1). \tag{3.23}$$

For *large vocabulary continuous speech recognition* (LVCSR) systems, the common technique is based on using n-gram LMs [Damerau, 1971] and approximates this probability as:

$$P(\mathbf{W}) = \prod_{m=1}^{M} P(w_m | w_{m-1}, \ldots, w_{m-n+1}). \tag{3.24}$$

In Formula (3.24), n is the n-gram order of the given LM. It determines the length of the conditioning word history. Usually n is in the range 2–4.

LMs are trained on text corpora, by estimating n-gram probabilities of word n-gram frequencies to *maximize the likelihood* (ML) of the training data [Dempster et al., 1977]. Let denote $C(w_m, \ldots, w_{m+k})$ the number of occurrences of the $k+1$ words $w_m, \ldots, w_{m+k}$ in the training corpus, then

$$P(w_m | w_{m-1}, w_{m-2}) \approx \frac{C(w_{m-2}, w_{m-1}, w_m)}{C(w_{m-2}, w_{m-1})}. \tag{3.25}$$

The major problem with this ML estimation approach is data sparsity. There are several approaches to robust estimation of LMs:

1. *Katz smoothing* [Katz, 1987] consists in combination of *discounting* and *backing-off*:

$$P(w_m | w_{m-1}, w_{m-2}) = \begin{cases} \dfrac{C(w_{m-2}, w_{m-1}, w_m)}{C(w_{m-2}, w_{m-1})}, \text{ if } C^* < C, \\ d\dfrac{C(w_{m-2}, w_{m-1}, w_m)}{C(w_{m-2}, w_{m-1})}, \text{ if } 0 < C \leq C^*, \\ \alpha(w_{m-1}, w_{m-2}) P(w_m | w_{m-1}), \; C = 0, \end{cases} \tag{3.26}$$

where C^* is a count threshold, d is a discount coefficient and α is a normalization constant. That means, that when there is enough statistic for some n-grams, the ML estimate is used. When the amount of available statistic is not sufficient according to the chosen threshold, then the same ML estimated is used, but discounted. The "discounted probability mass" is then distributed over unseen n-grams, which are estimated with the weighted version of their $(n-1)$-grams. In our example (Formula (3.26)) $n = 2$. The discounting coefficient d is based on Good-Turing estimates [Good, 1953; Katz, 1987]: $d = (r+1)n_{r+1}/(rn_r)$, where n_r is the number of n-grams that occur r times in the training corpus.

2. *Kneser-Ney smoothing* [Chen and Goodman, 1996; Kneser and Ney, 1995; Ney et al., 1994] is an improved smoothing algorithm.

3. *Class-based models* [Brown et al., 1992; Martin et al., 1998] is an alternative approach to robust LM estimation, where every word w_m has a corresponding class c_m, to which this word belongs. Then the probability of the word sequence is estimated using class n-gram probabilities:

$$P(\mathbf{W}) = \prod_{m=1}^{M} P(w_m|c_m)P(c_m|c_{m-1},\dots,c_{m-n+1}). \tag{3.27}$$

   As for the word n-grams, class n-grams are estimated via ML, and since the number of classes (typically several hundreds) is much smaller, than the number of words, this approach helps to overcome the data sparsity problem. The words are grouped together into the classes according to the underlying nature of these classes. It can be syntactical, semantical or statistical word classes. In the statistical approach for classes, the word classes are chosen to optimize the likelihood of the training data.

4. *Interpolation* of LMs relies on linearly weighted combination of several LMs with the same or different n-gram order.

Various other LM approaches have been proposed for ASR systems, for example, random forest LMs [Xu and Mangu, 2005], structured LMs [Chelba and Jelinek, 2000; Emami et al., 2003], LMs based on constraint dependency grammars (CDGs) [Wang et al., 2004] and others [Schwenk, 2007]. In recent years neural networks have become a popular approach for language modeling [Bengio et al., 2003; Le et al., 2011; Xu and Rudnicky, 2000]. The idea in [Bengio et al., 2003] is based on mapping a discrete n-gram word distribution to a continuous representation. Since the obtained distributions are smooth functions of the word representation, this approach allows to better generalize the unseen n-grams. In [Mikolov et al., 2010] recurrent neural network based LMs have been proposed. The long short-term memory (LSTM) neural network architecture for LMs has been explored in [Sundermeyer et al., 2012]. Neural network based LMs are usually used for ASR hypotheses rescoring in a post-processing step.

An evaluation metric for LMs is *perplexity* estimated on a test dataset as:

$$PPL = 2^{-\frac{1}{M}\sum_{m=1}^{M}\log_2(P(w_m|w_{m-1},\dots,w_1))}. \tag{3.28}$$

The lower the perplexity, the better the LM generalizes to predict unseen data.

In this thesis we will use 2-gram, 3-gram and 4-gram LMs.

3.6 Speech decoding techniques

After the acoustic, pronunciation and language models are estimated, it is possible to perform decoding, based on Formula (3.6). Existing decoding strategies can be characterized by the following aspects [Aubert, 2002]:

- *Network expansion*:
  - *Static* expansion of the search space, where the entire decoding network is generated before the decoding [Saon et al., 2005]. In this case, any search algorithm may be applied, though in practice, a *time-synchronous* Viterbi search is usually used [Forney, 1973; Viterbi, 1967] with implementation based on *weighted finite-state transducers* (WFSTs) [Mohri et al., 2002].
  - *Dynamic* (or *on-the-fly*) expansion of the search space [Rybach et al., 2013, 2011]. In this case, the required parts of the network are generated on demand during the search. Hence, decoders with the dynamic network expansion have much lower memory consumptions, but requires additional computations during the search process, that can slow down the decoding process. There are several decoding strategies, that can be implemented in this framework, for example, the *history conditioned lexical tree* (HCLT) [Ney et al., 1992] or WFST-based decoding with *on-the-fly transducer compositions* [Mohri et al., 2005] and *on-the-fly hypothesis rescoring* [Hori et al., 2007].
- *Search*:

– *Time-synchronous decoding*, where all hypotheses are evaluated in parallel, frequently referred to as Viterbi decoding, which approximates the solution to Formula (3.6) by searching only for the most probable state sequence. It relies on *beam pruning* to restrict the number of hypotheses, developed in parallel.

– *Asynchronous decoding* [Frankel and King, 2007; Picheny, 1999] (for example, *stack decoders* [Paul, 1992]), where the most promising hypothesis is followed first until the end of the speech signal is reached, thus, the search is carried out asynchronously with respect to time.

For the experiments reported in this thesis, we will apply decoding with a static network expansion and the Viterbi search with an WFST-based implementation. In the WFST-based approach, all components of the ASR system (acoustic, phonetic and language) are represented with individual WFSTs, which are then composed into a single WFST used for decoding.

In practice, to give more strength to a LM in comparison with an AM, the LM probabilities are scaled in the log-domain[2] with factor k, and Formula (3.6), is replaced with:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \sum_{\mathbf{Q}} P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q}|\mathbf{W})P(\mathbf{W})^k, \tag{3.29}$$

where the optimal value for k is usually estimated empirically on the development set.

Additional improvements in ASR accuracy can be obtained by performing multiple passes over the speech data. For this purpose the decoder can generate and save multiple recognition hypotheses in the form of *word lattices* [Richardson et al., 1995]. A word lattice consists of a set of nodes representing points in time and a set of directed arcs representing word hypotheses. Each arc can contain information about a word identity and corresponding acoustic and language model scores. Lattices can be rescored by using a higher-order (or another more advanced) language model. Also they can be transformed into another efficient representation called a *confusion network* [Mangu et al., 2000].

3.7 End-to-end ASR

One important trend in current ASR technology is training deep *end-to-end* ASR systems [Audhkhasi et al., 2017; Bahdanau et al., 2016; Chan et al., 2016; Chorowski et al., 2014; Collobert et al., 2016; Fritz and Burshtein, 2017; Graves and Jaitly, 2014; Miao et al., 2015a;

[2]Decoders typically work in the log-domain.

Ochiai et al., 2017; Yi et al., 2016; Zhang et al., 2016a, 2017; Zhu et al., 2016] that attempt to map an acoustic signal to a words sequence directly by means of neural network models. Some of these works aim to be exempted from the need of an HMM part, or presegmented training data, but still use lexicons, phoneme representations, LMs, or separate feature extractors, so not all of them are completely end-to-end.

One of the first major steps in this direction was introduced in [Graves et al., 2013] where, for the phoneme recognition task, a deep bidirectional LSTM RNN model was trained to map directly acoustic sequences to phonetics ones. This was done by using the *connectionist temporal classification* (CTC) objective function [Graves et al., 2006]. In the CTC approach, the alignment between the inputs and target labels is unknown. CTC can be implemented with a softmax output layer using an additional unit for the blank label $\emptyset$. The symbol $\emptyset$ corresponds to no output and is used to estimate the probability of outputting no label at a given time. The network is trained to optimize the total log-probability of all *valid* label sequences for training data. A set of valid label sequences for an input sequence is defined as the set of all possible label sequences of the input with the target labels in the correct order with repetitions and with label $\emptyset$ allowed between any labels. Targets for CTC training can be computed using finite state transducers (FSTs) [Sak et al., 2015], and the forward-backward algorithm can be used to calculate the CTC loss function.

State transition probability distribution and state priors are not required for CTC approach, in contrast to the hybrid DNN-HMM system. Several types of output units for CTC training have been explored in the literature, such as phones (or graphemes) [Miao et al., 2015a], words [Audhkhasi et al., 2017] or *grams* [Liu et al., 2017]. Due to the large number of word outputs in *acoustic-to-word* CTC models, they require significantly more training data in comparison with traditional ASR systems [Audhkhasi et al., 2017]. A maximum a-posteriori (MAP) training criterion instead of CTC was used in [Fritz and Burshtein, 2017] to train an end-to-end ASR system.

Most of the works devoted to end-to-end technology do not use any speaker adaptation techniques. In this thesis, we will implement our proposed adaptation techniques to one end-to-end system, described in [Miao et al., 2015a], to explore how the end-to-end technology can benefit from speaker adaptation (Chapter 9).

3.8 Speech recognition performance evaluation

Performance evaluation is characterized by three elements: a *criterion*, a *measure* and a *method* [Hirschman and Thompson, 1997]. In ASR the criterion is *recognition accuracy*, one of the most popular measures is *word error rate* (WER), and the method is comparing the sequence of recognized words with what was actually spoken and establishing alignment at the word level between these sequences by using a *dynamic programming* (DP) algorithm.

3.8.1 General evaluation scheme

LVCSR evaluation procedure contains two important steps:

1. Alignment of reference and recognized sequences of words. It is assumed that we have the text that was actually spoken (a reference). Alignment consists in establishing agreement at the word level between the original and recognized texts. The standard approach to alignment is the classical *Levenshtein (edit distance)* algorithm. If some additional information (such as time labels or phonological features of words) about reference is available, then it may be used in alignment process.

2. Computing the evaluation measure. After the alignment is processed, each word in reference and recognized sentences gets a status: *correct*, *substitution*, *deletion* (for a reference word) or *insertion* (for a recognized word). The evaluation measure is usually based on counts of substitution, deletion, insertion errors and correct words.

3.8.2 Alignment algorithms

Alignment of the original and recognized texts is usually performed by computing the Levenshtein distance using the DP algorithm. The Levenshtein distance is the minimal number (or weighted sum) of insertion, deletion and substitution word operations needed to transform the original text into the recognized one.

Existing alignment algorithms differ from each other by the way they compute weights for word operations. Let *rec* be a recognized word, *ref* – a word from the reference. There are two basic types of alignment [Fiscus and et al, 1998].

1. *Standard Levenshtein alignment (SLA).* This approach is considered to be the standard method for LVCSR performance evaluation tasks. In general, SLA can be characterized by the loss function:

$$\mathcal{F} = w_{cor}N_{cor} + w_{ins}N_{ins} + w_{del}N_{del} + w_{sub}N_{sub}, \tag{3.30}$$

   where N_{sub} , N_{ins} and N_{del} denote the total number of word substitutions, insertions and deletions respectively; N_{cor} — the number of correctly recognized words; w_{cor},

w_{ins}, w_{del} , w_{sub} — the constant weights assigned for operations of each type. Optimal weights (in the sense of minimizing WER) are: $w_{cor} = 0$, $w_{ins} = w_{del} = w_{sub} = 1$. It was shown in [Morris et al., 2004] that any choice of other weights requires special justification. This type of alignment (with some variations of weights) is the most commonly used. For example, NIST used it in Speech Recognition Scoring Toolkit for Speech-To-Text evaluation programs but sets up weights that equal (0,3,3,4) [Fiscus, 2009]; in the HTK toolkit (for building HMMs) weights are (0,7,7,10) [Young et al., 2006].

2. *Time-mediated alignment* (TMA) [Fiscus and et al, 1998]. TMA is a variation of DP alignment where word-to-word distances are based on the time of occurrence for individual words. TMA is computed by replacing the standard word-to-word distance weights with measures based on the times of beginnings and endings of words. The formulas for time-mediated word-to-word distances are[3]:

$$\begin{cases} D(correct) = |B(ref) - B(rec)| + |E(ref) - E(rec)| \\ D(insertion) = E(rec) - B(rec) \\ D(deletion) = E(ref) - B(ref) \\ D(substitution) = |B(ref) - B(rec)| + |E(ref) - E(rec)| + 0.001, \end{cases} \tag{3.31}$$

where $B(x)$ and $E(x)$ are the beginning and ending time marks of word x.

Other types of alignment reported in literature, such as word-weight-mediated, phonological [Fisher and Fiscus, 1993], alignment based on multiple dimension Levenshtein edit distance [Fiscus et al., 2006], modified TMA [Khokhlov and Tomashenko, 2011] and others are developed for particular applications and are not considered in this thesis.

3.8.3 Measures for evaluation

Most of the measures for LVCSR are based on magnitudes: N_{sub} , N_{ins} , N_{del} and N_{cor}. The common measure is WER, which is defined as the percentage ratio of the total number of word errors made by an ASR system to the total number of words in the reference:

$$\text{WER} = \frac{N_{sub} + N_{ins} + N_{del}}{N_{ref}} \times 100\%, \tag{3.32}$$

[3] http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm

where N_{ref} is the total number of words in the reference text. WER has been the most popular measure in continuous speech recognition assessment since the 1980s. NIST used WER as the primary measure in comparative technology evaluation programmes for ASR [Fiscus, 2009]. Other measures were suggested in literature, such as: weighted WER [Fiscus and et al, 1998], *sentence error rate* (SER); *word correct or incorrect* (WCI) and sentence correct or incorrect (SCI) [Strik et al., 2000]; *number of errors per sentence* (NES) [Strik et al., 2000]; *match error rate* (MER) [Morris et al., 2004]; *relative information lost* (RIL) [Maier, 2002; Morris et al., 2004]; *word information lost* (WIL) and *word information preserved* (WIP) measures [Morris, 2002]; *individual WER* (IWER) [Goldwater et al., 2010] and others. In this thesis we will consider WER as a basic metric for evaluation.

3.8.4 Absolute and relative improvement

When several ASR systems are compared it is a common practice to report the absolute and relative change in WER values, calculated between two systems with WERs: WER_1 and WER_2, as follows:

- **Absolute WER reduction**

$$\Delta_{abs}\text{WER} = \text{WER}_1 - \text{WER}_2 \tag{3.33}$$

- **Relative WER reduction**

$$\Delta_{rel}\text{WER} = \frac{\text{WER}_1 - \text{WER}_2}{\text{WER}_1} \times 100\%. \tag{3.34}$$

3.8.5 Statistical significance tests and confidence intervals

In the ASR system development, it is often important to know whether the observed difference in the performance is statistically significant or not. For this purpose several approaches have been proposed in the literature [Bisani and Ney, 2004; Gillick and Cox, 1989; Pallet et al., 1990]. In paper [Gillick and Cox, 1989] a *matched-pairs test* for comparing outputs of two ASR systems is described, which is applicable under the assumptions that the output of an ASR system can be divided into segments, in which the errors are independent of the errors in other segments. In paper [Bisani and Ney, 2004] *a bootstrap method* for significance analysis is presented. The idea is based on creating the replications of a statistic by random sampling from the test dataset with replacement (so-called Monte Carlo simulation).

In this thesis, if the results in terms of WER are reported to be *significantly different*, that means that the hypothesis is tested with *significance level* $\alpha = 5\%$. In addition, as it is done for example in [Bell, 2010; Povey, 2004; Świętojański, 2016] for baseline systems, we approximate the required WER change in terms of *confidence intervals* (CIs), calculated with $\alpha = 5\%$. For clarity, we report these CIs in some experiments for baseline systems (not for all systems to avoid redundancy).

Chapter 4

Acoustic adaptation

This chapter provides a review of adaptation techniques for both GMM and DNN acoustic models.

4.1 Speech variability and adaptation

Differences between training and testing conditions may significantly degrade recognition accuracy in ASR systems. The source of these differences may be of various nature [Benzeghiba et al., 2007] – speaker, channel and environment. Speech variations may occur both across different speakers (*across-speaker variability*) and within one speaker (*within-speaker variability*).

The across-speaker variability is connected with such factors as age, gender, foreign or regional accents, etc. In [Huang et al., 2001], with the use of statistical analysis methods, it was demonstrated that the two principal components in variation correspond to the gender and accent. From the physiological point of view, another speaker-dependent factor is the shape of the vocal tract, that determines the potential range within which parameters of voice for a given speaker may vary.

The within-speaker variability [Karlsson et al., 1998] appears, for example, when a speaker speaks at different rates. In the past it was shown that variations in speaking rate can degrade recognition performance in ASR systems [Mirghafori et al., 1996]. This effect is typically more significant for fast speech than for slow speech, that is, the higher the speaking rate, the higher the error rate in ASR systems. Fast speech is different from normal or slow speech in several aspects, such as acoustic-phonetic and phonological characteristics. Many methods for compensating the effects of speaking rate variability were proposed in the literature [Ban and Kim, 2012; Mirghafori et al., 1996; Nanjo et al., 2001; Siegler, 1995; Tomashenko and Khokhlov, 2014c].

Another important factor, which can be attributed to within-speaker variability, is the speech style. Reading, dictation or spontaneous speech may have different acoustic characteristics for the same speaker. In spontaneous speech, reductions of some phonemes or syllables often happen. Also it usually contains various disfluencies, such as repetitions, hesitations, filled pauses, false starts and others [Liu et al., 2006], which should be taken into account when building an ASR system. A speaker can speak with different volume – louder, quieter, in a whisper [Jou et al., 2004], or shout. Emotional state of a speaker also has an impact on the speech signal [Scherer, 2003].

Adaptation is an efficient way to reduce mismatches between the models and the data from a particular speaker, channel or another factor.

Three types of acoustic models (AMs) can be used in ASR systems:

- *Speaker independent* (SI) AM is trained on a training set of acoustic data from multiple speakers.
- *Speaker dependent* (SD) AM is trained on data from a single target speaker. Given enough training data, a SD AM can provide WERs several times lower than a SI AM [Lee et al., 1991].
- *Speaker adapted* (SA) AM is initially trained on data from multiple speakers, and then is adapted using a comparatively small amount of data from a target speaker.

The aim of AM adaptation is to reduce mismatches between training and testing acoustic conditions and improve the accuracy of the ASR system for a target speaker or channel using a limited amount of adaptation data from the target acoustic source.

4.1.1 Adaptation types and modes

Acoustic adaptation algorithms can be divided in two types:

- *Model-based adaptation* transforms the model parameters in order to optimize a certain criterion, for example, to maximize posterior probability or likelihood on the adaptation data.
- *Feature-space adaptation* transforms the acoustic feature vectors and does not require modification of the parameters of the acoustic model, so usually it is more suitable for real-time on-line ASR systems.

Speaker adaptation techniques can be applied in various modes:

- In the *supervised mode*, the true transcriptions of the adaptation data are supposed to be known.
- In the *unsupervised mode* there are no available (correct) transcriptions of the adaptation data. In this case transcripts for adaptation can be obtained from the first decoding pass by a SI or another model. Since the adaptation with erroneous transcriptions from the first decoding pass may degrade the performance of the unsupervised adaptation compared to the supervised one [Pitz et al., 2000], an important question for some adaptation techniques is how to deal with these transcription errors. One popular solution is the use of *confidence measures*[1] [Anastasakos and Balakrishnan, 1998; Gollan and Bacchiani, 2008; Pitz et al., 2000; Tomashenko et al., 2016b; Uebel and Woodland, 2001; Wang and Narayanan, 2002].

  .

With regard to the use of the accumulated adaptation data, speaker adaptation can also be classified by the following modes [Shinoda, 2011; Woodland, 2001]:

- *Batch / off-line / static adaptation*. This adaptation is performed after all the adaptation data are obtained.
- *Incremental / on-line / dynamic adaptation* [Zhang et al., 2000] is performed each time, a new portion of adaptation data is obtained, so that the system continues to adapt over time.

A speaker diarization system may also be required to detect speaker changes. The choice of the most suitable adaptation mode depends on the application.

4.1.2 Normalization

Normalization attempts to minimize the effects of variations in speech. The most popular normalization techniques include:

- *Cepstral mean and variance normalization* (CMVN). *Cepstral mean normalization* (CMN) subtracts the average (cepstral) feature value. Generally, it reduces the sensitivity to channel variations. Cepstral variance normalization scales each feature

[1]In ASR, *confidence measures* are used in different tasks to evaluate reliability of recognition results. An extensive survey on confidence measures for ASR can be found in [Jiang, 2005].

coefficient to have a unit variance and empirically this has been shown to reduce sensitivity to additive noise [Benesty et al., 2007].

- *Vocal-Tract-Length Normalization* (VTLN) [Lee and Rose, 1996]. Variations in vocal-tract length of different speakers cause variations in formant frequencies. VTLN is usually performed by warping the frequency-axis of the spectra of the given speaker by an corresponding *warp factor* before extracting cepstral features. Warp factors can be found by ML estimation of the warped utterances with respect to a given model and transcriptions.

4.2 Adaptation for GMM models

Many efficient adaptation algorithms have been developed for GMM-HMM AMs (see, for example, reviews in [Lee and Huo, 2000; Shinoda, 2010; Woodland, 2001]). In this section we consider some of the most popular adaptation approaches, such as MAP, MLLR, as well as their modifications, and speaker space methods.

4.2.1 Maximum a posteriori (MAP)

4.2.1.1 Standard MAP approach

One of the most popular approaches for acoustic model adaptation is *maximum a posteriori* (MAP) adaptation [Gauvain and Lee, 1994], also referred to as *Bayesian* adaptation. Given some adaptation data $\mathcal{O} = (\mathcal{O}_1, \dots, \mathcal{O}_R)$, MAP adaptation aims to maximize the following objective function:

$$\mathcal{F}_{MAP}(\mathbf{\Lambda}) = \sum_{r=1}^{R} \log p_{\mathbf{\Lambda}}(\mathcal{O}_r|\phi_r)p_0(\mathbf{\Lambda}), \tag{4.1}$$

where $\phi_r = \phi(\mathbf{W}_r)$, as before in Section 3.3.3, is a composite HMM model, corresponding to the (correct) transcription of the training sentence $\mathcal{O}_r$ and $\mathbf{W}_r$ is a sequence of words in this transcription. In comparison with the ML objective function (see Formula (3.19)), the likelihood in MAP-estimation is weighted by the prior distribution of the parameters $p_0(\mathbf{\Lambda})$.

In MAP adaptation a SI model is used as a prior probability distribution over model parameters. Let m denote the index of a Gaussian component in the SI acoustic model (AM), and $\boldsymbol{\mu}_m$ the mean of this Gaussian. Then the MAP estimation of the mean vector is

$$\widehat{\boldsymbol{\mu}}_m = \frac{\tau\boldsymbol{\mu}_m + \sum_t \gamma_m(t)\mathbf{o}_t}{\tau + \sum_t \gamma_m(t)}, \tag{4.2}$$

where τ is the parameter that controls the balance between the maximum likelihood estimate of the mean and its prior value; $\gamma_m(t)$ is the posterior probability of Gaussian component m at time t. As can be seen from Formula (4.2), the smaller the occupation likelihood of a Gaussian component $\gamma_m = \sum_t \gamma_m(t)$, the closer the mean MAP estimate remains to the SI component mean $\boldsymbol{\mu}_m$. In MAP adaptation, every single mean component in the system is updated with a MAP estimate, based on the prior mean, the weighting, and the adaptation data.

An important advantage of MAP adaptation consists in the fact that, when the amount of adaptation data increases, the parameters asymptotically converge to speaker-dependent performance. On the other hand, one drawback of MAP adaptation is that it requires more adaptation data to be effective when compared to transformation-based techniques, such as *maximum likelihood linear regression* (MLLR) (see Section 4.2.2), because MAP adaptation is defined at the component level and adapts each Gaussian individually. When larger amounts of adaptation data become available, MAP begins to perform better than MLLR. The two adaptation approaches can be combined to improve the performance (see, for example, [Digalakis and Neumeyer, 1996]).

Other approaches developed to make MAP adaptation more robust when only a small adaptation set is available include *vector field smoothing* (VFS) [Ohkura et al., 1992; Takahashi and Sagayama, 1997; Tonomura et al., 1995], *structural maximum a posteriori* (SMAP) [Shinoda and Lee, 1997, 2001], and the combination of MAP estimation and *weighted neighbor regression* (WNR) in the so-called *MAP-WNR adaptation* [He et al., 2000].

4.2.1.2 Vector field smoothing (VFS)

The *vector field smoothing* (VFS) technique [Ohkura et al., 1992; Takahashi and Sagayama, 1997; Tonomura et al., 1995] provides a solution to overcome one limitation of MAP adaptation and improves its performance when only very small amount of adaptation data is available. The VFS technique is based on the assumption that one speaker acoustic feature space can be continuously transfer to another one. Hence, it solves the problem of adapting the parameters in case of unseen adaptation data. The VFS algorithm is applied to the Gaussian mean vectors and consists of three steps [Tonomura et al., 1995]:

- *Estimation of transfer vectors.* The transfer vectors represent the difference of the mean vectors between the initial SI model $\boldsymbol{\mu}_m$ and the target MAP-model $\widehat{\boldsymbol{\mu}}_m$:

$$\mathbf{v}_m = \widehat{\boldsymbol{\mu}}_m - \boldsymbol{\mu}_m, \tag{4.3}$$

where $m \in G_a$ (G_a is the set of Gaussian distributions, which have associated adaptation data).

- *Interpolation of transfer vectors*. Interpolation is applied for those Gaussian components $\boldsymbol{\mu}_n$, that do not have associated adaptation data ($n \in G_{\overline{a}}$):

$$\mathbf{v}_n = \frac{\sum_{k \in N(n)} \lambda_{n,k} \mathbf{v}_k}{\sum_{k \in N(n)} \lambda_{n,k}}, \tag{4.4}$$

where $N(n)$ is the set of nearest neighbors to $\boldsymbol{\mu}_n$. The set of nearest neighbors $N(n)$ is defined based on the Euclidean distance between mean vectors of Gaussian components and $\boldsymbol{\mu}_n$. The weighting coefficient $\lambda_{n,k}$ depends on the distance between $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_k$. Then, $\widehat{\boldsymbol{\mu}}_n$ is estimated with the obtained transfer vector $\mathbf{v}_n$, as follows:

$$\widehat{\boldsymbol{\mu}}_n = \boldsymbol{\mu}_n + \mathbf{v}_n \tag{4.5}$$

- *Smoothing of transfer vectors*. For all transfer vectors $\mathbf{v}_m$ ($m \in G_a$), which Gaussian components are present in the adaptation set, a smoothing is performed as follows:

$$\mathbf{v}_m^s = \frac{\sum_{k \in N(m)} \lambda_{m,k} \mathbf{v}_k}{\sum_{k \in N(m)} \lambda_{m,k}}, \tag{4.6}$$

$$\widehat{\boldsymbol{\mu}}_m^s = \boldsymbol{\mu}_m + \mathbf{v}_m^s. \tag{4.7}$$

4.2.1.3 Structural maximum a posteriori (SMAP)

Structural maximum a posteriori (SMAP) adaptation is another approach to improve the MAP estimates obtained when the amount of adaptation data is small [Shinoda and Lee, 1997, 2001]. In SMAP adaptation, parameters are shared by using a tree structure. First, a tree of Gaussian distributions is build using Kullback-Leibler divergence as a distance between mixture components. The root node corresponds to the whole acoustic space, and each leaf corresponds to a single Gaussian in the AM. Parameters of parent nodes are used as priors, and the MAP estimation is performed from the root to the leaves in a cascade way.

4.2.2 Maximum likelihood linear regression (MLLR)

Maximum likelihood linear regression (MLLR) is an alternative algorithm to AM adaptation, in which a set of linear transforms is estimated to map a SI model to a new adapted model in such a way that the likelihood of the adaptation data is maximized [Gales, 1998; Leggetter and Woodland, 1995], given the transformed Gaussian parameters. The MLLR can be applied either only to the means of Gaussians, or to the both means and variances. The adapted estimate of the mean $\widehat{\boldsymbol{\mu}}$ is calculated as:

$$\widehat{\boldsymbol{\mu}} = \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \tag{4.8}$$

where $\mathbf{A}$ is $d \times d$ *transformation matrix* (d is the dimensionality of acoustic feature vectors) and $\mathbf{b}$ is a *bias vector*. If we denote $\mathbf{W} = [\mathbf{b}\ \mathbf{A}]$ and $\boldsymbol{\xi} = [w\mu_1 \ldots \mu_d]^T$ – the extended mean vector, where w represents a fixed bias offset (which is usually equals to 1) [Young et al., 2006], then Formula (4.8) becomes more compact:

$$\widehat{\boldsymbol{\mu}} = \mathbf{W}\boldsymbol{\xi}. \tag{4.9}$$

The transformation $\mathbf{W}$ is estimated using the expectation maximization (EM) algorithm (the details can be found in [Gales, 1998; Leggetter and Woodland, 1995; Young et al., 2006]).

4.2.2.1 Regression classes

Typically the transformation matrix is tied over a large number of Gaussians, that allows the system to adapt all these Gaussians using a small amount of adaptation data. In general, the MLLR adaptation is very flexible. If only a small amount of adaptation data is available, then a single global transformation for all Gaussians in a GMM-HMM model can be applied. However, increasing the amount of adaptation data, it is possible to improve MLLR adaptation with the use of several more specific transformation matrices, each of which is applied to a certain group of Gaussians. Gaussians can be grouped according to the phoneme classes to which they belong. A popular approach for using such classes is based on the construction of a *regression class tree.*

The regression class tree is built to cluster together those Gaussians, that are close in acoustic space, so that similar Gaussians can be transformed in a similar way. The tree can be built using a SI model and a centroid splitting algorithm with the Euclidean distance measure. The tree topology makes it possible to adapt those Gaussian components, for which there is no data in the adaptation set. Also, it allows the transformations to be applied dynamically, depending on the amount of adaptation data for a particular regression class.

4.2.2.2 Constrained MLLR (CMLLR) and feature-space MLLR (fMLLR)

There are two possible ways of transforming covariance matrices: unconstrained [Gales and Woodland, 1996] and constrained [Digalakis et al., 1995]. In the unconstrained MLLR, transformations for means and variances of Gaussians are estimated independently. On the contrary, in the CMLLR means and variances are adapted using the same transformation:

$$\begin{cases} \widehat{\boldsymbol{\mu}} = \mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \\ \widehat{\boldsymbol{\Sigma}} = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{T}. \end{cases} \tag{4.10}$$

In practice, CMLLR is used more often as an adaptation technique for features, than for acoustic models. In this case typically one regression class is used and the adaptation method is referred as *feature-space MLLR* (fMLLR). For fMLLR, Formula (4.10) can be rewritten for features as follows:

$$\widehat{\mathbf{o}}_t = \mathbf{A}^{-1}\mathbf{o}_t - \mathbf{A}^{-1}\mathbf{b}. \tag{4.11}$$

MLLR can be combined with the MAP adaptation techniques. In [Neumeyer et al., 1995] various ML transformation-based techniques, as well as their combination with MAP, are compared. In [Digalakis and Neumeyer, 1996], the mean vectors, obtained by MLLR, are used as priors for mean vectors in MAP adaptation. Transformation-based methods work well when only a small amount of adaptation data is available. Bayesian methods perform better when the amount of adaptation data increases. Hence, their combination inherits the advantages of these two approaches. Maximum a posteriori linear regression (MAPLR) was proposed in [Chesta et al., 1999; Chou, 1999], where a standard MLLR approach was reformulated under MAP framework. A combination of SMAP (Section 4.2.1.3) and MLLR was investigated in [Siohan et al., 2002] in the method called SMAPLR.

4.2.3 Speaker space methods

4.2.3.1 Cluster adaptive training (CAT)

The basic idea of speaker clustering is to find in the training dataset a cluster of those speakers who are acoustically close to a given test speaker. Then this cluster can be used directly for speech recognition, in case when we have an HMM model corresponding to this cluster. A simple example of this approach is gender-dependent AMs. Also each of these training speakers can be adapted to the test speaker, and the resulting adapted data are used to estimate the new parameters of the model [Padmanabhan et al., 1998].

Speaker clustering is performed according to the chosen distance metric between speakers. Popular metrics include Bhatacharyya distance [Kosaka and Sagayama, 1994], likelihood distance measure [Gales, 2000] and others [Hazen and Glass, 1997; Yoshizawa et al., 2001].

Cluster adaptive training (CAT) [Gales, 2000] can be considered as an extension of speaker clustering. Instead of a SI model, a set of cluster-specific models is trained on more homogenous datasets. In [Gales, 2000] speaker model's mean parameters are estimated in the form of a linear combination of all the cluster means. The Gaussian component variances and priors are assumed to be the same for all clusters. At recognition time, a linear combination of models is selected where the set of interpolation weights gives a speaker-specific transform.

4.2.3.2 Eigenvoice-based adaptation

Eigenvoice approach, proposed in [Kuhn et al., 2000], also represents the adapted model in the form of a weighted sum of a small number of basis eigenvoice vectors obtained from a set of reference speakers. These eigenvoices are found using principal components analysis (PCA) for set of supervectors obtained from all means in the set of SD HMM systems. The eigenvoices with the largest eigenvalues are selected as a basis set. These vectors are orthogonal to each other and represent the most important components of variation between the reference speakers.

4.2.4 Speaker adaptive training (SAT)

Speaker adaptive training (SAT) aims to provide a more suitable model for speaker adaptation. A method, that explicitly compensates for the inter-speaker variations in the HMM parameter estimation process, is introduced in [Anastasakos et al., 1996]. First an initial SI model is trained. Then for each speaker in the training dataset, MLLR mean (or fMLLR) transformations are estimated. Then using the obtained SD transforms, the parameters of the initial models (means, variances and mixture weights) are re-estimated on the whole training dataset. The SAT procedure tends to result in more compact AMs with higher likelihoods on the training set and smaller variances. Usually SAT AMs provide significantly better results with adaptation, in comparison with SI models.

4.3 Adaptation for DNN models

Adaptation of DNN acoustic models is a rapidly developing research area. In recent years DNNs have replaced conventional GMMs in most state-of-the-art ASR systems, because it has been shown that DNN-HMMs outperform GMM-HMMs in different ASR tasks [Hinton et al., 2012a]. Many adaptation algorithms that have been developed for GMM-HMM systems (see Section 4.2) cannot be easily applied to DNNs because of the different nature of these models. GMM is a generative model and it fits the training data so that the likelihood of the data given the model is maximized. In contrast, DNN is a discriminative model. Since DNN parameter estimation uses discriminative criteria, it is more sensitive to segmentation errors and can be less reliable for unsupervised adaptation. Various adaptation methods have been developed for DNNs. These methods, as for GMM-HMM AMs, can also be categorized in two broad classes, *feature-space* and *model-based*.

Model-based adaptation methods rely on direct modifications of DNN model parameters. In [Swietojanski et al., 2016; Swietojanski and Renals, 2014], *learning speaker-specific hidden unit contributions* (LHUC) was proposed. The main idea of LHUC is to directly parametrize amplitudes of hidden units, using speaker-dependent amplitude parameters. The idea of learning amplitudes of activation functions was proposed earlier in [Trentin, 2001]. Other model-based DNN adaptation techniques include linear transformations (Section 4.3.1), adaptation using regularization techniques (Section 4.3.2), subspace methods (Section 4.3.3) and others.

Feature-space adaptation methods operate in the feature space and can either transform input features for DNNs, as it is done, for example, in fMLLR adaptation [Seide et al., 2011a] or use auxiliary features (Section 4.3.6).

4.3.1 Linear transformation

One of the first adaptation methods developed for neural networks was *linear transformation* that can be applied at different levels of the DNN-HMM system:

- *Input features*, as in the *linear input network* (LIN) transformation [Abrash et al., 1995; Gemello et al., 2006; Li and Sim, 2010; Neto et al., 1995; Trmal et al., 2010] or feature-space discriminative linear regression (fDLR) [Seide et al., 2011a; Yao et al., 2012];

- *Activations of hidden layers*, as in the *linear hidden network* (LHN) transformation [Gemello et al., 2006];

- *Softmax layer*, as in the *linear output network* (LON) transformation [Li and Sim, 2010] or in the *output-feature discriminative linear regression* (oDLR) [Yao et al., 2012].

Wherever the transformation is applied, weights are initialized with an identity matrix and then trained by minimizing the error at the output of the DNN system while keeping the weights of the original DNN fixed. Adaptation of hybrid tied-posterior acoustic models [Stadermann and Rigoll, 2005] can also be considered a linear transform of posteriors. The authors of [Dupont and Cheboub, 2000] describe a method based on linear transformation in the feature space and PCA.

4.3.1.1 Linear input network (LIN)

In the LIN [Abrash et al., 1995; Gemello et al., 2006; Li and Sim, 2010; Neto et al., 1995; Trmal et al., 2010] and fDLR [Seide et al., 2011a; Yao et al., 2012] approaches, the linear transformation is applied to the input features, as shown in Figure 4.1. These adaptation techniques aim to linearly transform the SD features of a particular speaker to better match the SI DNN AM.

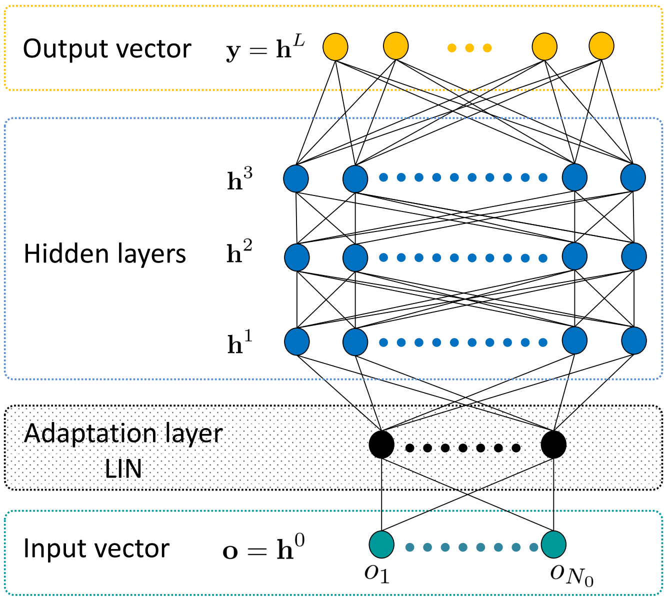


Figure 4.1 Linear input network (LIN) adaptation

In LIN adaptation the input vector $\mathbf{h}^0 \in \mathbb{R}^{N_0}$ is transformed to $\mathbf{h}^0_{\text{LIN}}$ as follows:

$$\mathbf{h}^0_{\text{LIN}} = \mathbf{W}^{\text{LIN}}\mathbf{h}^0 + \mathbf{b}^{\text{LIN}}, \tag{4.12}$$

where N_0 is the size of the input layer, $\mathbf{W}^{\text{LIN}} \in \mathbb{R}^{N_0 \times N_0}$ is the weight matrix and $\mathbf{b}^{\text{LIN}} \in \mathbb{R}^{N_0}$ is the bias vector. Since in ASR the input vector $\mathbf{h}^0$ typically composed of a sequence of several speech frames: $\mathbf{h}^0 = \widehat{\mathbf{o}}_t = [\mathbf{o}_{t-T}, \ldots, \mathbf{o}_{t+T}]$, when the adaptation set is small, it is more preferable [Yu and Deng, 2014] to use per-frame transformation, with smaller number of parameters:

$$\mathbf{o}^{\text{LIN}} = \mathbf{W}_o^{\text{LIN}}\mathbf{o} + \mathbf{b}_o^{\text{LIN}}, \tag{4.13}$$

where $\mathbf{W}_o^{\text{LIN}} \in \mathbb{R}^{D\times D}$ is the weight matrix and $\mathbf{b}_o^{\text{LIN}} \in \mathbb{R}^D$ is the bias vector, D is the dimension of acoustic feature vectors ($N_0 = (2T+1)D$), and the final transformed input feature vector is constructed as: $\mathbf{h}^0_{\text{LIN}} = \widehat{\mathbf{o}}_t^{\text{LIN}} = [\mathbf{o}_{t-T}^{\text{LIN}}, \dots, \mathbf{o}_{t+T}^{\text{LIN}}]$.

Based on the LIN, the adaptation parameter estimation via MAP linear regression was studied in [Huang et al., 2014] and is referred to as *feature-space MAP LIN* (fMAPLIN).

4.3.1.2 Linear output network (LON)

In LON [Li and Sim, 2010] and *output-feature discriminative linear regression* (oDLR) [Yao et al., 2012; Yu et al., 2013] adaptation algorithms, the linear transformation is applied to the softmax layer, as shown in Figure 4.2. The LON adaptation can be applied in two different ways: either before the calculation of the softmax layer weights, or after it. Both approaches lead to very similar linear transforms [Yu and Deng, 2014], but the number of parameters can differ a lot depending on the difference between the hidden and output layer dimensions.

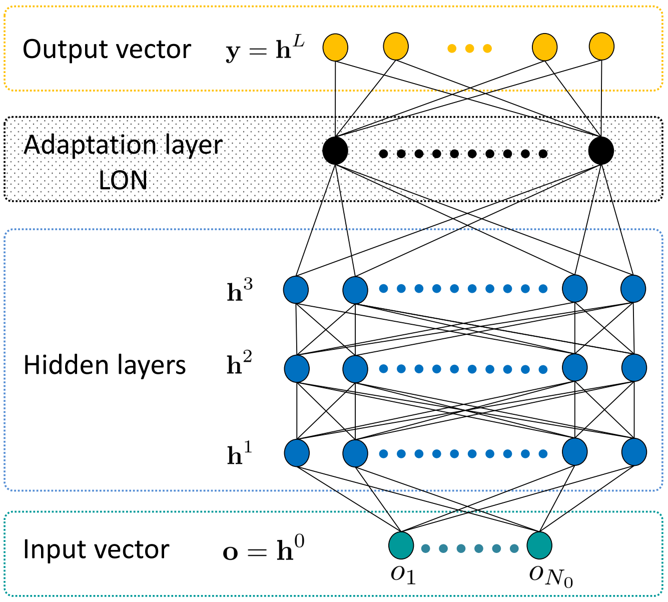


Figure 4.2 Linear output network (LON) adaptation

4.3.1.3 Linear hidden network (LHN)

In a similar manner, linear transformations can also be applied to the hidden layers, as in LHN [Gemello et al., 2006]. Illustration of LHN adaptation is given in Figure 4.3. The

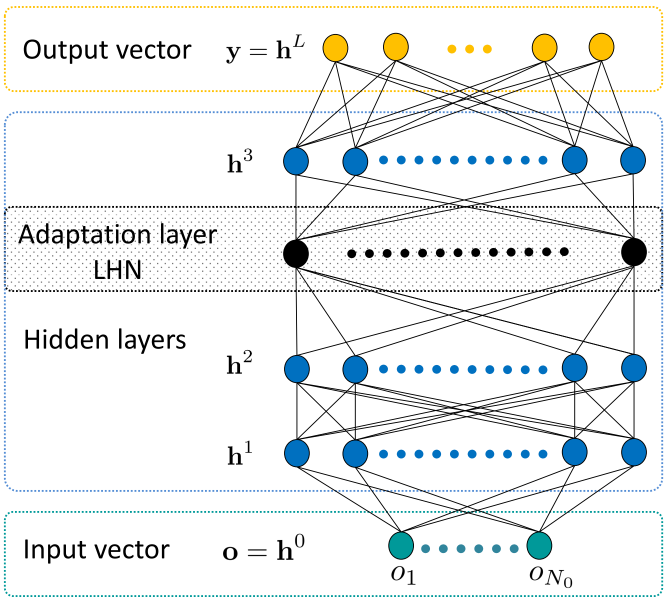


Figure 4.3 Linear hidden network (LHN) adaptation

hierarchical MAP approach applied to the LHN was proposed in [Huang et al., 2015b].

The choice of the appropriate adaptation techniques depends on the task, the number of parameters, and the size of the adaptation set. An experimental comparison of different linear transformation approaches can be found in [Gemello et al., 2006; Huang et al., 2015b; Li and Sim, 2010].

4.3.2 Regularization techniques

In order to improve generalization during the adaptation *regularization techniques*, such as L_2 (or L_2-prior) regularization [Li and Bilmes, 2006; Liao, 2013], Kullback-Leibler divergence (KLD) regularization [Huang and Gong, 2015; Tóth and Gosztolya, 2016; Yu et al., 2013], conservative training [Albesano et al., 2006; Gemello et al., 2006], reduction of the number of adapted parameters [Ochiai et al., 2014; Stadermann and Rigoll, 2005; Xue et al., 2014a] and others [Ochiai et al., 2014; Shen et al., 2016; Xue et al., 2014d] are used.

4.3.2.1 L_2 regularization

The key idea of the DNN adaptation with L_2 regularization [Li and Bilmes, 2006; Liao, 2013; Yu and Deng, 2014] is to update DNN parameters or only part of them by using the modified training objective function $\mathcal{F}_{L_2}(\mathbf{W},\mathbf{b};\mathbb{T})$, which takes into account the difference between the parameters of SI and speaker-adapted models. This difference is calculated using L_2 norm as follows:

$$R_2(\mathbf{W}_{\text{SI}}-\mathbf{W}) = \|\text{vec}(\mathbf{W}_{\text{SI}}-\mathbf{W})\|_2^2 = \sum_{l=1}^{L}\left\|\text{vec}(\mathbf{W}_{\text{SI}}^l-\mathbf{W}^l)\right\|_2^2, \tag{4.14}$$

where $\mathbf{W}_{\text{SI}}$ and $\mathbf{W}$ correspond to the original SI and adapted models respectively, and $\left\|\text{vec}(\mathbf{W}^l)\right\|_2^2$ is the Frobenious norm of matrix $\mathbf{W}^l$. The objective function is modified with this L_2 regularization term:

$$\mathcal{F}_{L_2}(\mathbf{W},\mathbf{b};\mathbb{T}) = \mathcal{F}(\mathbf{W},\mathbf{b};\mathbb{T}) + \lambda R_2(\mathbf{W}_{\text{SI}}-\mathbf{W}), \tag{4.15}$$

where λ is the weight regularization factor, that controls the impact of the original SI model to the adaptation criterion. The larger the regularization term is, the closer the adapted model is forced to be to the SI one.

4.3.2.2 Kullback-Leibler divergence (KLD) regularization

Using the *Kullback-Leibler divergence (KLD) regularization* [Huang and Gong, 2015; Tóth and Gosztolya, 2016; Yu et al., 2013] is another way to adapt the model. The idea of this adaptation technique consists in keeping the senone distribution, estimated from the adapted model, to be close to the distribution estimated from the SI model. To measure the distance between these distributions, KLD is used [Yu et al., 2013]. By adding this KLD as a regularization term to the adaptation criterion and removing the terms unrelated to the model parameters, the objective function with the KLD regularization can be written as follows [Yu and Deng, 2014]:

$$\mathcal{F}_{KLD}(\mathbf{W},\mathbf{b};\mathbb{T}) = (1-\lambda)\mathcal{F}(\mathbf{W},\mathbf{b};\mathbb{T}) + \lambda R_{\text{KLD}}(\mathbf{W}_{\text{SI}},\mathbf{b}_{\text{SI}};\mathbf{W},\mathbf{b};\mathbb{T}), \tag{4.16}$$

where λ is the regularization weight,

$$R_{\text{KLD}}(\mathbf{W}_{\text{SI}},\mathbf{b}_{\text{SI}};\mathbf{W},\mathbf{b};\mathbb{T}) = \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{C} P_{\text{SI}}(i|\mathbf{o}_m;\mathbf{W}_{\text{SI}},\mathbf{b}_{\text{SI}})\log P(i|\mathbf{o}_m;\mathbf{W},\mathbf{b}), \tag{4.17}$$

$P_{\mathrm{SI}}(i|\mathbf{o}_m;\mathbf{W}_{\mathrm{SI}},\mathbf{b}_{\mathrm{SI}})$ and $P(i|\mathbf{o}_m;\mathbf{W},\mathbf{b})$ are the probabilities that the observation vector $\mathbf{o}_m$ belongs to class i, estimated with the SI and adapted models respectively. Unlike in L_2-regularization approach to adaptation, which imposes the constraints to the DNN parameters, in the KLD-regularization approach the constraints are applied to the output probabilities.

The regularization weight $\lambda \in [0,1]$ can be optimized on the development set. The large values of λ provide small impact from the adaptation data to the final model.

4.3.2.3 Reducing the number of adapted parameters

Alternative regularization technique for preventing overfitting during the adaptation is to adapt not the entire DNN model, but only a subset of its parameters. In [Stadermann and Rigoll, 2005] only a sub-set of the hidden units with maximum variance, computed on the adaptation data, is retrained.

The number of speaker-specific parameters is reduced in [Xue et al., 2014a; Yu and Deng, 2014] through factorization based on singular value decomposition (SVD). One approach proposed in [Xue et al., 2014a] is referred to as *SVD bottleneck adaptation*. It is based on applying the SVD to the fully connected weight matrix $\mathbf{W}_{m\times n} \in \mathbb{R}^{m\times n}$ (where $m \geq n$) :

$$\mathbf{W}_{m\times n} = \mathbf{W}^1_{m\times n}\mathbf{\Sigma}_{n\times n}(\mathbf{W}^2_{n\times n})^T, \tag{4.18}$$

where $\mathbf{\Sigma}_{n\times n}$ is a diagonal matrix with singular values of matrix $\mathbf{W}_{m\times n}$ on the diagonal. If $\mathbf{W}_{m\times n}$ is a sparse matrix with rank $r \ll n$, then Formula (4.18) can be rewritten as

$$\mathbf{W}_{m\times n} \approx \mathbf{W}^3_{m\times r}\mathbf{\Sigma}_{r\times r}(\mathbf{W}^4_{n\times r})^T = \mathbf{W}^3_{m\times r}\mathbf{W}^5_{r\times n}. \tag{4.19}$$

Hence, the original matrix $\mathbf{W}_{m\times n}$ can be approximated with two smaller matrices: $\mathbf{W}^3_{m\times r}$ and $\mathbf{W}^5_{r\times n}$, as shown in Figure (4.4). To apply an SVD bottleneck adaptation, an additional r-dimensional linear layer is added to the BN layer:

$$\mathbf{W}_{m\times n} \approx \mathbf{W}^3_{m\times r}\mathbf{S}_{r\times r}\mathbf{W}^5_{r\times n}, \tag{4.20}$$

where $\mathbf{S}_{r\times r}$ is initialized with the identity matrix for SI model. During the adaptation, this matrix $\mathbf{S}_{r\times r}$ is adapted to a particular speaker, while keeping matrices $\mathbf{W}^3_{m\times r}$ and $\mathbf{W}^5_{r\times n}$ fixed for every layer. Since in this case only $r \times r$ parameters have to be adapted instead of $m \times n$, as r can be much smaller then m and n, this approach allows to adapt all layers of the DNN by adapting only small-sized matrices for each layer. Thus, such technique gives the possibility to decrease the required amount of adaptation data.

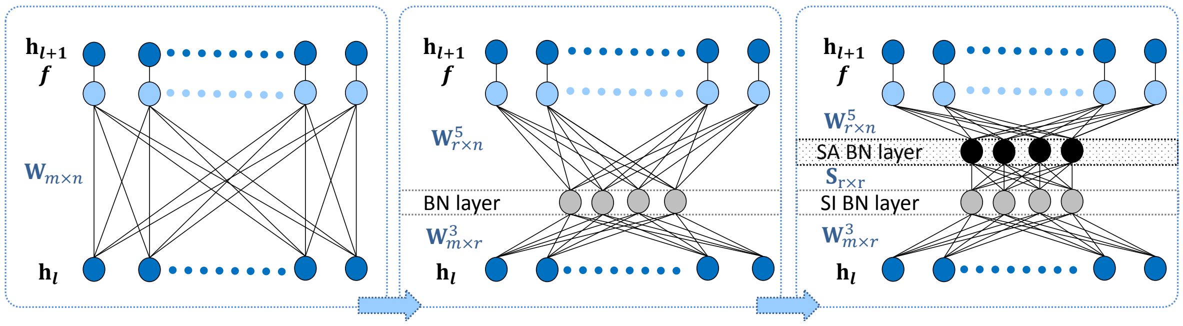


Figure 4.4 SVD-BN adaptation: f corresponds to a nonlinear activation function; $\mathbf{h}_l$, $\mathbf{h}_{l+1}$ are the layers of the NN between which a SI and SA BN layers are inserted.

In [Xue et al., 2014d] the SVD also is applied on the weight matrices $\mathbf{W}_{m\times n}$ in the trained SI DNNs (Formula (4.18)), and then only the diagonal matrices with singular values $\mathbf{\Sigma}_{n\times n}$ are fine-tuned with the adaptation data. Since during the adaptation the weight matrices can be modified only by changing the singular values, this techniques proposes the solution for the over-fitting problem. Regularized SAT of subsets of DNN parameters is explored in [Ochiai et al., 2014].

4.3.2.4 Conservative training

When a neural network trained with a large set of parameters has to be adapted with new data that does not appropriately represent the original training data, then a problem, addressed in the literature as *catastrophic forgetting* [French, 1999], occurs. In particular, this problem appears when some of the targets of the original NN are not present in the adaptations set. In this situation, the previously learned information in the NN can be distorted or forgotten during the adaptation.

The *conservative training* [Albesano et al., 2006; Gemello et al., 2006] proposes a solution to eliminate this problem. During the adaptation, the conservative training does not assigns zero value to the targets of the missing units, using instead as target values the outputs computed by the original network.

4.3.3 Subspace methods

Subspace adaptation methods aim to find a speaker subspace and then construct the adapted DNN parameters as a point in the subspace.

In [Dupont and Cheboub, 2000] an approach similar to the eigenvoice technique [Kuhn et al., 2000], was proposed for the fast speaker adaptation of NN AMs. It is based on using the affine transformation in the feature space. In contrast to the standard approach, where the whole transformation is adapted, in [Dupont and Cheboub, 2000] adaptation to a new speaker is performed using PCA by optimizing several (K) principal components, which represent the most meaningful speaker specific variations. A new speaker is represented by the parameter space described by the first K eigenvoices. Adaptation is performed by estimating the corresponding K eigenvoices coefficients. This allows to reduce the required amount of adaptation data.

As noticed in [Yu and Deng, 2014], this idea can be extended to a more general case and to other adaptation techniques, for example, to any other linear transformations (LON, LHN, etc). Given the set of S speakers, for each speaker an adaptation matrix $\mathbf{W}_{SA}$ can be estimated (by any adaptation technique, considered before). If $\mathbf{a} = vec(\mathbf{W}_{\mathrm{SA}})$ is the vectorization of the matrix $\mathbf{W}_{\mathrm{SA}}$, then PCA can be done for the set of S vectors in the speaker space in order to obtain the eigenvectors, that define the *principal adaptation matrices*. It is assumed [Dupont and Cheboub, 2000; Yu and Deng, 2014] that the number of speakers S in the training set is big enough, so that the new speaker can be represented as a point in the space, defined by these S speakers. A new speaker is defined as a linear combination of the eigenvectors:

$$\mathbf{a} = \bar{\mathbf{a}} + \mathbf{U}\mathbf{g}, \tag{4.21}$$

where $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_S)$ is the eigenvectors matrix; $\mathbf{g}$ is the projection of the adaptation vector onto principal directions; $\bar{\mathbf{a}}$ is the mean of the adaptation parameters of all speakers. The eigenvectors corresponding to small variances in the PCA are discarded after the PCA procedure, and the dimensionality of the speaker space can be reduced to $K < S$. In the reduced speaker space the adaptation vector $\mathbf{a}$ can be approximated as

$$\mathbf{a} \approx \bar{\mathbf{a}} + \widetilde{\mathbf{U}}\widetilde{\mathbf{g}}, \tag{4.22}$$

where $\widetilde{\mathbf{U}} = (\mathbf{u}_1, \ldots, \mathbf{u}_K)$ is the reduced eigenvectors matrix. In Formula (4.22), $\bar{\mathbf{a}}$ and $\widetilde{\mathbf{U}}$ are estimated on the training set using data of S speakers; and $\widetilde{\mathbf{g}}$ is estimated for a new speaker on the adaptation set.

In [Wu and Gales, 2015] a *multi-basis adaptive neural network* is proposed, where a traditional DNN topology is modified and a set of sub-networks, referred as *bases* were introduced. This DNN has a common input layer and a common output layer for all the bases. Each basis has several fully-connected hidden layers and there is no connections between neurons from different bases. The outputs of bases are combined by linear interpolation using a set of adaptive weights. The adaptation to a given speaker can be performed through optimization of interpolation weights for this speaker. The idea of this approach was motivated by the CAT, developed for GMM AMs (Section 4.2.3.1). Paper [Tan et al., 2015] also investigates the CAT framework for DNNs. A subspace learning speaker-specific hidden unit contributions (LHUC) adaptation was proposed in [Samarakoon and Sim, 2016c].

4.3.4 Factorized adaptation

Factorized adaptation [Li et al., 2014a; Qian et al., 2016; Samarakoon and Sim, 2016b; Tran et al., 2016; Yu et al., 2012] takes into account different factors that influence the speech signal. These factors can have different nature (speaker, channel, background noise conditions and others) and can be modeled explicitly before incorporating them into the DNN structure, for example, in the form of auxiliary features [Li et al., 2014a] (see also Section 4.3.6), such as i-vectors, or can be learnt jointly with the neural network AM [Tang et al., 2017]. The first case, when factors, such as noise or speaker information, are estimated explicitly from the training and testing data, and are then fed to the DNN AM, is also known as *noise-aware* or *speaker-aware training* correspondingly [Yu and Deng, 2014], or, in general case, as *context-aware training* [Tang et al., 2017].

Paper [Li et al., 2014a] proposed to incorporate the auxiliary features, consisting of average noise estimates, computed for each utterance, and noise-distorted speech, into the DNN AM. These features correspond to the two factors and are included directly to the input of the softmax layer. The general scheme of this approach is shown in Figure 4.5. This approach is claimed to be related [Li et al., 2014a] to the vector Taylor series (VTS) based adaptation [Moreno, 1996] technique, developed for noise-robust ASR. Also this approach can be considered from the joint factor analysis (JFA) point of view [Kenny et al., 2007].

Auxiliary features, representing different factors, can also be included into the other parts of the DNN structure. For example, they can be appended to the input feature vectors. We will consider this case in more detail in Section 4.3.6.

In paper [Yu et al., 2012] two types of factorized DNNs were introduced: *joint and disjoint models*. In the first model, hidden factors (speaker and environment conditions) and triphone tied states are modeled jointly. On the contrary, in the disjoint factorized model, factors and triphone tied states are estimated separately using different DNNs. However, as it was noticed in [Yu and Deng, 2014], the total number of parameters in such networks is typically too large to use them in real-world applications.

In [Tran et al., 2016] an extension of the LIN adaptation (see Section 4.3.1.1), so-called *factorized LIN* (FLIN), has been investigated for the case when adaptation data for a given

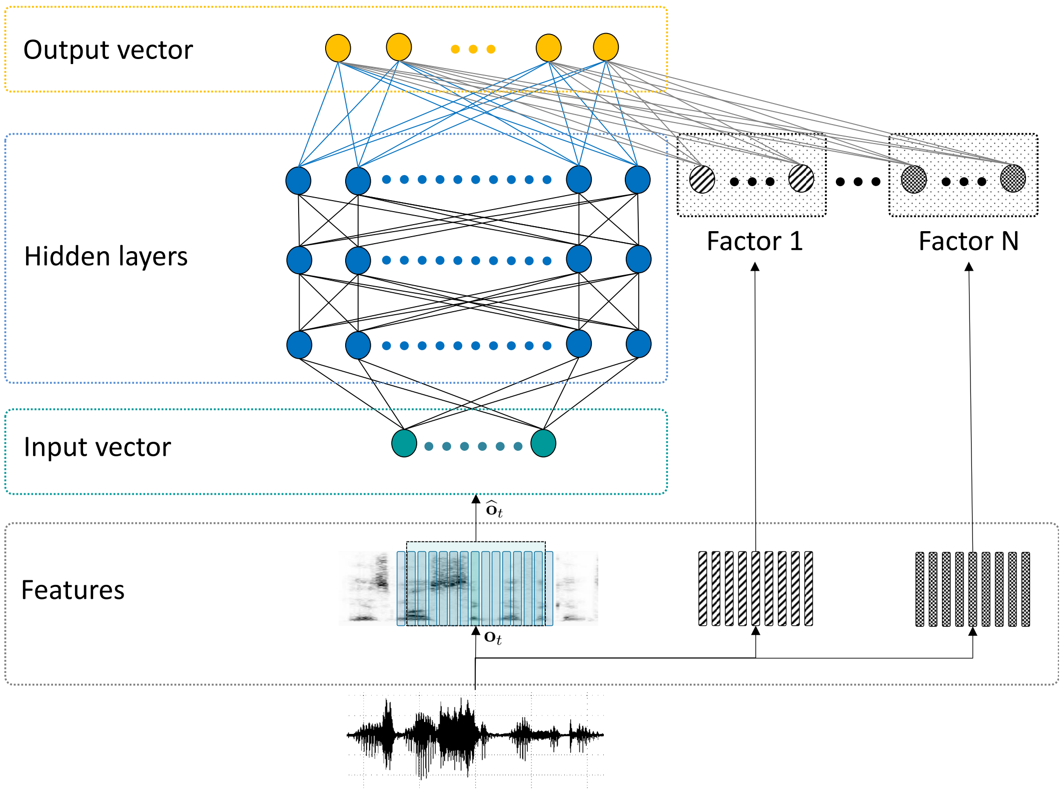


Figure 4.5 Factorized adaptation

speaker include multiple acoustic conditions. The feature transformations are represented as weighted combinations of affine transformations of the enhanced input features. Each set of LIN transformations can represent a class of noise-environment conditions. The weights in the combination are obtained from a vector characterizing the noise context conditions for the given utterance. The noise context features are obtained using an auxiliary neural network from the enhanced and noisy features. This auxiliary network is jointly trained with the LIN transformations.

In [Delcroix et al., 2015] a *context adaptive DNN* is proposed. This DNN contains one or several *factorized hidden layers*. Each factorized hidden layer contains several sub-layers, which represent different acoustic conditions. The output of the factorized layer is a weighted averaging over the outputs of all sub-layers, where weights are the posterior probabilities of the factor classes. Factorized hidden layer adaptation was also studied in [Samarakoon and Sim, 2016a,b].

Multi-factor aware joint DNN training was proposed in [Qian et al., 2016], where factors are dynamically estimated using a DNN. The input to the factor extractor networks are noisy corrupted features. Each factor is modeled using several-layer DNN with a BN layer in the

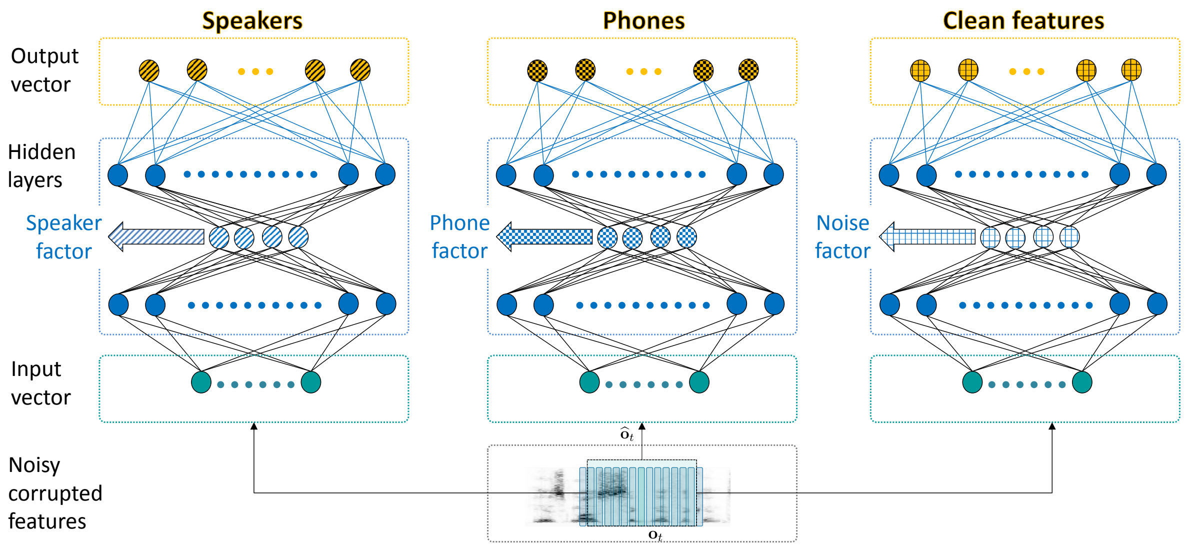


Figure 4.6 Factor extraction using BN features

middle, and factor-specific output layer, as shown in Figure 4.6. Three types of outputs were considered: speakers, context-independent (CI) phones and clean features. The output of a BN layer represents a corresponding factor. These factors are integrated into the main DNN AM, which is trained to classify triphone tied states, and are trained jointly with it.

A different approach called *collaborative joint learning* is proposed in [Tang et al., 2017] where the two models – for ASR and for speaker recognition task are trained in a collaborative manner, mutually improving the performance of each other.

4.3.5 Multi-task learning

The concept of *multi-task learning* (MTL) has recently been applied to the task of speaker adaptation in several works [Huang et al., 2015a; Li et al., 2015b; Swietojanski et al., 2015] and has been shown to improve the performance of different model-based DNN adaptation techniques, such as LHN [Huang et al., 2015a] and LHUC [Swietojanski et al., 2015].

The basic idea of MTL framework, used in [Bell et al., 2017; Huang et al., 2015a; Swietojanski et al., 2015] consists in adding to the neural network structure an additional auxiliary output layer or several layers, as shown in Figure 4.7. Each output layer is associated with a specific task. Typically, a main task is used during an ASR process, but a DNN can be trained or adapted using auxiliary (secondary) tasks. This auxiliary tasks (layers) correspond to different sets of targets and usually have a lower dimension than the main task. For example, the main task can represent the senones, which correspond to a large number of *context-dependent* (CD) triphone tied states, and the auxiliary task can represent CI monophone states or clusters of senones from the main task.

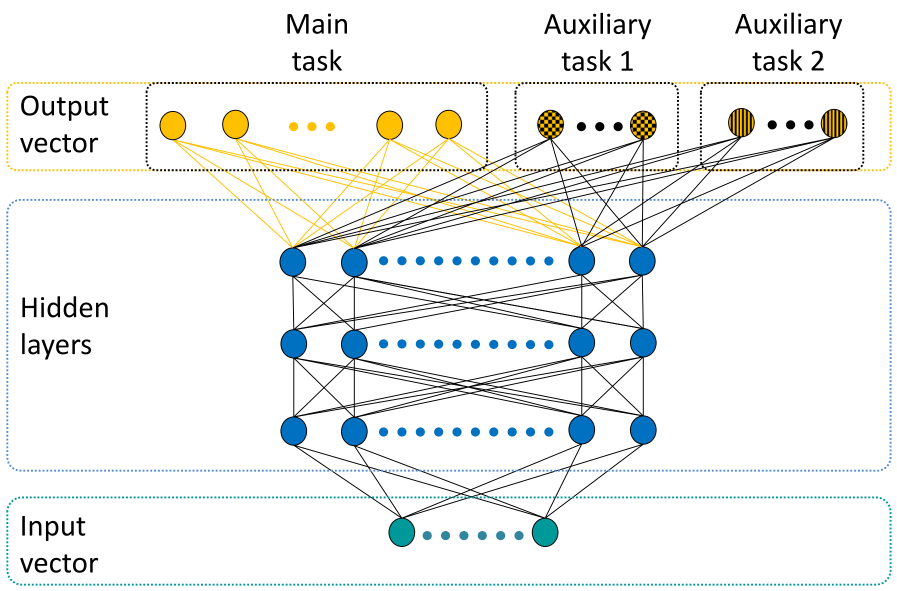


Figure 4.7 Multi-task learning (MTL) neural architecture for adaptation

This type of the DNN architecture can be trained using a MTL approach. MTL is a machine learning technique which allowed a classifier to learn several related tasks in parallel, using a shared representation [Caruana, 1998]. The principal goal of MTL is to improve generalization by leveraging the information, contained in auxiliary tasks. In MTL DNN training, the error vector from each output layer is back-propagated to the same last HL. Then, the error vectors, corresponding to different output layers, are combined together in a linear combination. The combined error vector is further back-propagated to the previous HLs. During the adaptation, the auxiliary tasks can be used to deal with the data sparsity problem and unseen senones.

In [Pironkov et al., 2016b] a speaker classification task was used as an auxiliary task to train RNN-LTSM AM. In [Pironkov et al., 2016a], in addition to the speaker classification task, the i-vector extraction auxiliary task was implemented.

A slightly different idea was proposed earlier in [Price et al., 2014] in the form of special hierarchy of output layers, where tied triphone state layer are followed by monophone state layer. Figure 4.8 illustrates the DNN with an auxiliary output layer, added on the top of the original main output layer, as proposed in [Price et al., 2014]. During adaptation, the DNN parameters are updated to predict posterior probabilities for the over HMM states, corresponding to the auxiliary monophone output layer, which has much lower dimension then the main triphone output layer. The errors are back-propagated through the original softmax layer to the rest of the DNN. Before adaptation, the weights of the hierarchy output layer are trained using SI data, keeping fixed the layers below. During adaptation, the weights of the original network are updated, using the full hierarchical structure, but keeping the weights in the hierarchy output layer fixed. During decoding, only the main original output triphone layer is used.

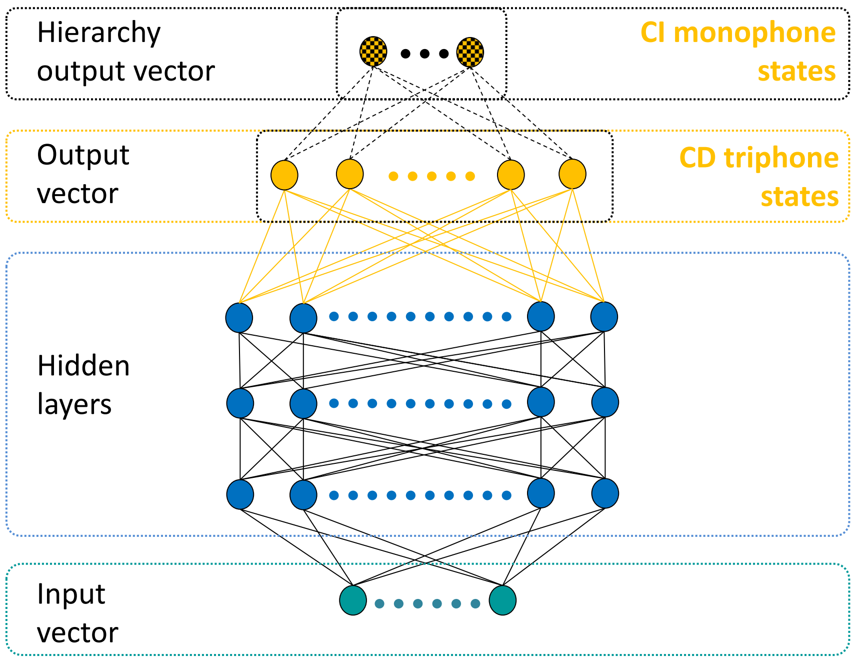


Figure 4.8 DNN architecture for adaptation with hierarchy output layer

A different multi-task learning approach, called *collaborative joint learning*, is proposed in [Tang et al., 2017]. It relies on inter-task information propagation, which can improve the performance of each task by the auxiliary information derived from other tasks.

4.3.6 Auxiliary features

Using *auxiliary features*, such as *i-vectors* [Gupta et al., 2014; Karanasou et al., 2014; Saon et al., 2013; Senior and Lopez-Moreno, 2014], is another widely used approach in which the acoustic feature vectors are augmented with additional speaker-specific or channel-specific features computed for each speaker or utterance at both training and test stages. Another example of auxiliary features is the use of speaker-dependent BN features obtained from a speaker aware DNN used in a far field speech recognition task [Liu et al., 2014]. Alternative method is adaptation with *speaker codes* [Abdel-Hamid and Jiang, 2013; Huang et al., 2016;

Xue et al., 2014c]. In [Murali Karthick et al., 2015] speaker-specific subspace vectors, obtained by the SGMM model, are used as auxiliary features to adapt CNN AMs.

4.3.6.1 I-vectors

The use of *speaker identity vectors* (or *i-vectors*) has become a popular approach for speaker adaptation of DNN AMs [Gupta et al., 2014; Karanasou et al., 2014; Miao et al., 2014; Saon et al., 2013; Senior and Lopez-Moreno, 2014]. Originally i-vectors were developed for speaker verification and speaker recognition tasks [Dehak et al., 2011], and nowadays they have become a very common technique in these domains. I-vectors can capture the relevant information about the speaker's identity in a low-dimensional fixed-length representation [Dehak et al., 2011; Saon et al., 2013].

I-vector extraction

The acoustic feature vector $\mathbf{o}_t \in \mathbb{R}^D$ can be considered as a sample, generated with a *universal background model* (UBM), represented as a GMM with K diagonal covariance Gaussians [Dehak et al., 2011; Saon et al., 2013]:

$$\mathbf{o}_t \sim \sum_{k=1}^{K} c_k \mathcal{N}(\cdot; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.23}$$

where c_k are the mixture weights, $\boldsymbol{\mu}_k$ are means and $\boldsymbol{\Sigma}_k$ are diagonal covariances. The acoustic feature vector $\mathbf{o}_t(s)$, belonging to a given speaker s is described with the distribution:

$$\mathbf{o}_t(s) \sim \sum_{k=1}^{K} c_k \mathcal{N}(\cdot; \boldsymbol{\mu}_k(s), \boldsymbol{\Sigma}_k), \tag{4.24}$$

where $\boldsymbol{\mu}_k(s)$ are the means of the GMM, adapted to the speaker s. It is assumed that there is a linear dependence between the SD means $\boldsymbol{\mu}_k(s)$ and the SI means $\boldsymbol{\mu}_k$, which can be expressed in the form:

$$\boldsymbol{\mu}_k(s) = \boldsymbol{\mu}_k + \mathbf{T}_k \mathbf{w}(s), \quad k = 1, \dots, K, \tag{4.25}$$

where $\mathbf{T}_k \in \mathbb{R}^{D \times M}$ is a *factor loading matrix*, corresponding to component k and $\mathbf{w}(s)$ is the i-vector, corresponding to speaker s[2]. Each $\mathbf{T}_k$ contains M bases, that span the subspace of the important variability in the component mean vector space, corresponding to component k.

[2]More strictly, i-vector is estimated as the mean of the distribution of random variable $\mathbf{w}(s)$.

The detailed description of how to estimate the factor loading matrix, given the training data $\{\mathbf{o}_t\}$, and how to estimate i-vectors $\mathbf{w}(s)$, given $\mathbf{T}_k$ and speaker data $\{\mathbf{o}_t(s)\}$, can be found, for example, in [Dehak et al., 2011; Saon et al., 2013].

Integration of i-vectors into a DNN model

Various methods of i-vector integration into a DNN AM have been proposed in the literature.

The most common approach [Gupta et al., 2014; Saon et al., 2013; Senior and Lopez-Moreno, 2014] is to estimate i-vectors for each speaker (or utterance), and then to concatenate it with acoustic feature vectors, belonging to a corresponding speaker (or utterance). The obtained concatenated vectors are introduced to a DNN for training, as shown in Figure 4.9. In the test stage i-vectors for test speakers also have to be estimated, and input in a DNN in the same manner.

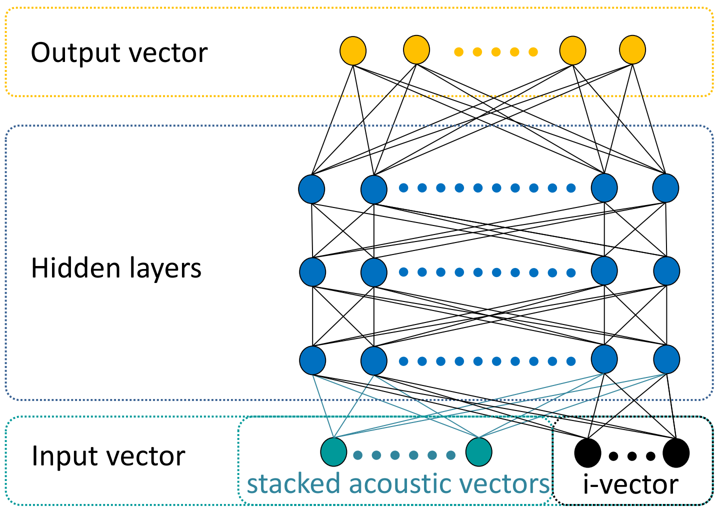


Figure 4.9 Using i-vectors for DNN adaptation: concatenation of i-vectors with input features

Unlike acoustic feature vectors, which are specific for each frame, an i-vector is the same for a chosen group of acoustic features, to which it is appended. For example, i-vector can be calculated for each utterance, as in [Senior and Lopez-Moreno, 2014], or estimated using all the data of a given speaker, as in [Saon et al., 2013]. I-vectors encode those effects in the acoustic signal, to which an ASR system is desired to be invariant: speaker, channel and background noise. Providing to the input of a DNN the information about these factors makes it possible for a DNN to normalize the acoustic signal with respect to them.

An alternative approach of i-vector integration into the DNN topology is presented in [Miao et al., 2014, 2015b], where an input acoustic feature vector is normalized though

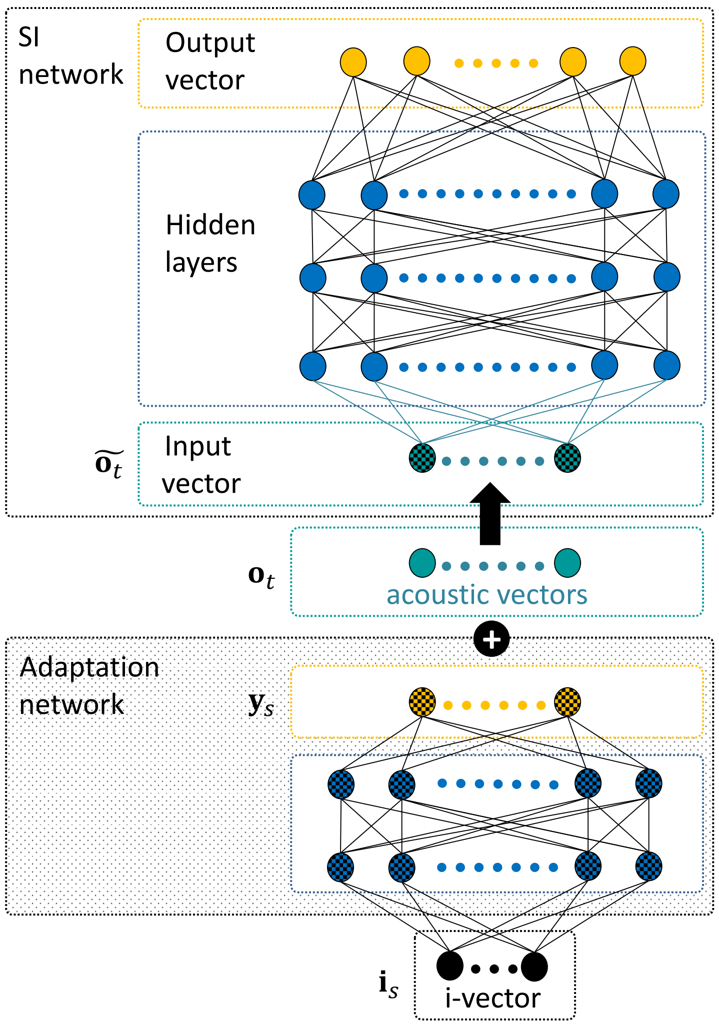


Figure 4.10 Using i-vectors for DNN adaptation: adaptation network

a linear combination of it with a speaker-specific normalization vector obtained from an i-vector, as shown in Figure 4.10. Firstly, an original SI DNN is built. Secondly, a small adaptation network is built, keeping a SI DNN fixed. This adaptation network is learnt to take i-vectors $\mathbf{i}_s$ as input and generate SD linear shifts $\mathbf{y}_s$ to the original DNN feature vectors. After the adaptation network is trained, the SD output vector $\mathbf{y}_s$ is added to every original feature vector $\mathbf{o}_t$ from speaker s:

$$\widetilde{\mathbf{o}_t} = \mathbf{o}_t \oplus \mathbf{y}_s, \tag{4.26}$$

where $\oplus$ denotes element-wise addition. Finally, the parameters of the original DNN model are updated in the new feature space, while keeping the adaptation network unchanged. This gives a SAT DNN model. Similar approaches have been studied in [Goo et al., 2016; Lee et al., 2016]. Also i-vector dependent feature space transformations were proposed in [Li and Wu, 2015a].

In paper [Dimitriadis et al., 2016] two different i-vector representations are presented: (1) *noise i-vectors*, estimated only from the noise component of the noisy speech signal; and (2) *noisy i-vectors*, estimated from noisy speech signal. Another type of factorized i-vectors is described in [Karanasou et al., 2014], where speaker and environmental i-vectors are used together.

In [Garimella et al., 2015] *casual i-vectors* are calculated using all previous utterances of a given speaker for accumulating sufficient statistics. Exponentially decaying weights are applied on previous speech frames to give more importance to the recent speech signal, in order to make i-vectors to be able to capture more recent speaker and channel characteristics.

Attribute-based i-vectors are explored in [Zheng et al., 2016] for accent speech recognition.

4.3.6.2 D-vectors

Deep vectors or *d-vectors* were originally proposed for text-dependent speaker verification [Variani et al., 2014]. They also described speaker information and are derived by a DNN which is trained to predict speakers at frame-level. For extracting the d-vector, firstly, a DNN is trained to classify speakers at the frame-level. Then the averaged output activations of last hidden layer are used as the speaker representation. Paper [Li and Wu, 2015b] explores the idea of augmenting DNN inputs with d-vectors derived by *LSTM projected* (LSTMP) RNNs [Sak et al., 2014]. Both the d-vector extraction and senone prediction are LSTMP based, and the whole network is jointly optimized with multi-task learning.

4.3.6.3 Speaker codes

Adaptation with *speaker codes*, proposed in [Abdel-Hamid and Jiang, 2013; Xue et al., 2014c], relies on a joint training procedure for (1) an adaptation NN from the whole training set and (2) small speaker codes, estimated for each speaker only using data from that speaker, as shown in Figure 4.11, The speaker code is fed to the adaptation NN to form a nonlinear transformation in feature space to normalize speaker variations. This transformation is controlled by the speaker code. During adaptation, a new speaker code for a new speaker is learned, optimizing the performance on the adaptation data.

In [Xue et al., 2014b] instead of stacking an adaptation NN below the initial speaker independent NN and normalizing speakers features with speakers codes, speaker codes are fed directly to the hidden layers and the output layer of the initial NN and the output layer of the initial NN. In [Huang et al., 2016] speaker adaptation with speaker codes was applied for RNN-BLSTM AMs.

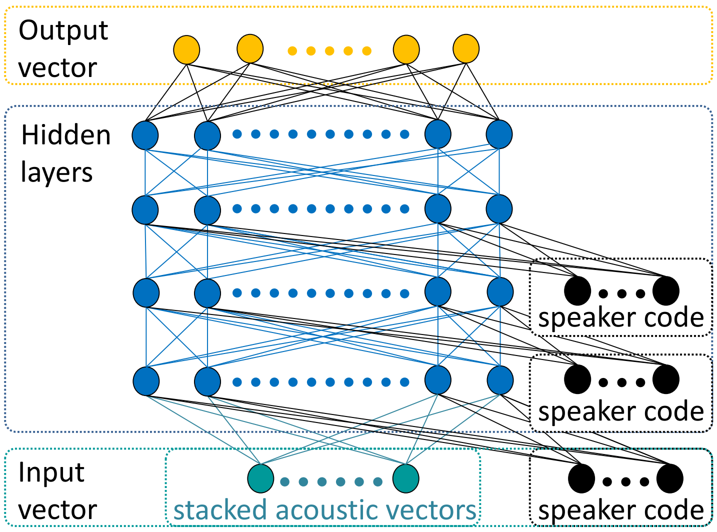


Figure 4.11 Speaker codes for DNN adaptation

4.3.7 Adaptation based on GMMs

The most common way of combining GMM and DNN models for adaptation is using GMM-adapted features, for example fMLLR, as input for DNN training [Kanagawa et al., 2015; Parthasarathi et al., 2015; Rath et al., 2013; Seide et al., 2011a]. In [Lei et al., 2013] likelihood scores from DNN and GMM models, both adapted in the feature space using the same fMLLR transform, are combined at the state level during decoding.

Another method is *temporally varying weight regression* (TVWR) [Liu and Sim, 2014], where DNN posteriors are transformed, using a regression model, into the time-varying scaling factors for the Gaussian weights.

However, none of these approaches can be considered as a universal method for transfer of adaptation algorithms from GMM models to DNNs. Development and investigation of a method, that can solve this problem, led us to *GMM-derived features*, which we use as input to train a DNN [Tomashenko and Khokhlov, 2014b, 2015]. This approach will be presented and extensively explored in Chapters 6–10.

Chapter 5

Speech corpora

This chapter describes speech corpora and language models, used in the experiments throughout this thesis.

We perform a series of experiments to explore the proposed acoustic model adaptation framework using four different corpora, which are suitable for this purpose. This corpora differ among themselves in such characteristics as:

- language;
- training dataset size;
- testing dataset sizes;
- amount of data available for adaptation;
- vocabulary size;
- number of speakers;
- type of speech;
- quality of available text transcriptions.

It is important to explore the adaptation framework in different conditions to get a more complete picture of the approach, its properties, strengths and weaknesses, and boundaries of applicability. We carry out certain types of experiments for particular corpora. Below in this chapter we describe four speech corpora used in experiments.

5.1 Wall Street Journal (WSJ)

In this thesis, the WSJ0 (CSR-1) part of the Wall Street Journal Corpus (WSJ) [Paul and Baker, 1992] was used for experiments. This corpus consists of read speech in English with texts from the Wall Street Journal news. The part of the corpus used in the experiments was recorded with a Sennheiser close-talking microphone, 16 kHz. The phoneme set consists of 39 phonemes[1]. This corpus was used for experiments in Chapter 7.

Training data

For acoustic models training we used 7 138 utterances from 83 speakers (42 male and 41 female) from the standard SI-84 training set, which correspond to approximately 15 hours (12 hours of speech and 3 hours of silence) recordings.

Adaptation data

Dataset *si_et_ad* consists of 320 utterances from 8 speakers (5 male and 3 female). We used this dataset in experiments with semi-supervised adaptation (Section 7.1.2).

Test data and language models

Evaluation was carried out on the two standard WSJ0 evaluation tests:

1. Test *si_et_05* is a November '92 NIST evaluation set with 330 read utterances from 8 speakers[2] (about 40 utterances per speaker). A standard WSJ trigram closed LM without verbalized punctuation (NVP, non-verbalized punctuation) with a 5K word vocabulary was used during recognition.

2. Test *si_et_20* consists of 333 read utterances from 8 speakers. A WSJ trigram open NVP LM with a 20K word vocabulary was used. The *out-of-vocabulary* (OOV) rate is about 1.5%.

Both LMs were pruned as in the Kaldi WSJ recipe with the threshold 10^{-7} [Povey et al., 2011b].

[1]ARPAbet phoneme set is used: https://en.wikipedia.org/wiki/Arpabet
[2]These 8 speakers are the same for *si_et_05*, *si_et_20* and *si_et_ad* datasets

5.2 TED-LIUM

The experiments were conducted on the TED-LIUM corpus [Rousseau et al., 2012, 2014]. The corpus is comprised of TED talks[3] in English. It has been released by LIUM laboratory [Rousseau et al., 2012] within the context of the participation in the International Workshop on Spoken Language Translation (IWSLT) 2011 evaluation campaign, and later extended [Rousseau et al., 2014].

We used the last (second) release of this corpus [Rousseau et al., 2014]. This publicly available dataset contains 1 495 TED talks that amount to 207 hours (141 hours from male and 66 hours from female) speech data from 1 242 speakers, 16kHz. The phoneme set is the same, as for the WSJ corpus (Section 5.1). This corpus is used for experiments in Chapters 8, 9 and 10.

Training and test data

For experiments with SAT and adaptation we removed from the original corpus data for speakers who had less than 5 minutes of data, and from the rest of the corpus we made four datasets: training set, development set and two test sets. Characteristics of the obtained datasets are given in Table 5.1.

Table 5.1 Datasets statistics for the TED-LUM corpus

Characteristic		**Dataset**			
		Training	Development	Test$_1$	Test$_2$
Duration, hours	Total	171.66	3.49	3.49	4.90
	Male	120.50	1.76	1.76	3.51
	Female	51.15	1.73	1.73	1.39
Duration per speaker, minutes	Mean	10.0	15.0	15.0	21.0
	Minimum	5.0	14.4	14.4	18.3
	Maximum	18.3	15.4	15.4	24.9
Number of speakers	Total	1 029	14	14	14
	Male	710	7	7	10
	Female	319	7	7	4
Number of words	Total	-	36 672	35 555	51 452

[3]Technology, Entertainment, Design: http://www.ted.com

The motivation for creating the new test and development datasets was to obtain datasets that are more representative and balanced in characteristics (gender, duration) than the original ones and more suitable for adaptation experiments.

Language models

For evaluation two different LMs are used:

1. *LM-cantab* is the publicly available 3-gram LM *cantab-TEDLIUM-pruned.lm3*[4] with 150K word vocabulary. The same LM is used in the Kaldi *tedlium s5* recipe[5].

2. *LM-lium* is a 4-gram LM from TED-LIUM corpus with 152K word vocabulary. It is similar to the one currently used in the Kaldi *tedlium s5_r2* recipe.[6] We conducted part of the experiments presented here using this LM in order to be compatible with the most recent Kaldi recipe and for comparison purposes with the results of the TDNN acoustic models. The only difference is that we modified a little the training set, removing from it data presented in our test and development datasets.

These two LMs are used in different series of experiments. The ASR results with different LMs are not comparable between each other. Also because of the fact that some data from the development and test sets may be part of the training corpus for *LM-cantab*, this LM can be biased towards the test sets, and more ASR errors are due to acoustics than due to the LM. For this reason, most of the final results are reported for *LM-lium*.

5.3 STC

The STC corpus is a microphone corpus (16 kHz) of read Russian speech collected at Speech Technology Center[7].

Training data

The training set contains approximately 27 hours of speech data from 203 (111 male and 92 female) speakers. An 11k vocabulary without LM was used in evaluation. The phone set consists of 52 phonemes. This corpus is used for experiments with supervised speaker adaptation in Chapter 6.

[4]http://cantabresearch.com/cantab-TEDLIUM.tar.bz2

[5]https://github.com/kaldi-asr/kaldi/tree/master/egs/tedlium/s5; date of access: January, 2016

[6]https://github.com/kaldi-asr/kaldi/tree/master/egs/tedlium/s5_r2; date of access: March, 2017

[7]http://speechpro.com/

Adaptation and test data

The adaptation experiments were conducted on data from 20 (10 male and 10 female) speakers excluded from the training set. For each speaker there are approximately 180 utterances, corresponding to 16-17 minutes of speech data. The amount of adaptation data is 5 minutes for each speaker, and the rest of the data (11-12 minutes for each speaker) is used for testing in all experiments. The test set consists of 2395 utterances. The total number of word occurrences in the test set is approximately 19K.

5.4 Arabic MGB 2016 challenge

The 2016 Multi-Genre Broadcast (MGB-2) Challenge is a controlled evaluation of speech recognition and lightly supervised alignment using Aljazeera Arabic TV channel recordings [Ali et al., 2016]. In Section 8.7 we report experimental results on the MGB-2 Challenge corpus, provided by the organizers to all participants.

Training and development data

The training data for AMs, provided by the organizers, consists of approximately 1 128 hours of Arabic broadcast speech, obtained from more than 2 000 broadcast shows on the Aljazeera Arabic TV channel over a period of 10 years, from 2005 until 2015. According to the MGB-2 organizers [Ali et al., 2016], most of the speech data (around 70%) is Modern Standard Arabic (MSA), and the remaining part contains speech in different Arabic dialects, such as Egyptian, Gulf, Levantine and North African.

The original data has a corresponding time-aligned transcription output from a lightly supervised alignment based on Aljazeera closed-captions, with varying quality of manual transcription. Table 5.2 presents some statistics for the training and development datasets.

For AM training, the LIUM has iteratively selected and aligned 648.3 hours of speech data (see the last line of Table 5.2). The detailed description of LIUM's data selection and alignment strategy is described in [Tomashenko et al., 2016f]. We use this selected data for AM training.

For experiments with adaptation we use grapheme-based lexicon with 40 graphemes. The number of speakers, estimated using meta data, is 14 249. This number is considered when performing speaker adaptation or per-speaker normalization, but this may not correspond to the real number of speakers in the training corpus, as, for example, a speaker presented in different shows, will be counted several times. The estimated number of speakers in

Table 5.2 Statistics for MGB-2016 training and development datasets. The first line of the table corresponds to the training, provided by the MGB organizers, and the last line corresponds to the subset of the training dataset selected for AM training.

Dataset	**Shows**	**Duration, hours**	**Utterances**	**Words**
Training (original)	2 214	1 128.0	376 011	7 815 566
Dev	16	8.5	4 940	57 647
Training (selected for AM training)	-	648.3	398 438	4 422 123

the development set equals to 132. Since for the test data set the original text data was not provided by the MGB-2 organizers, and the evaluation process on the test dataset was performed by them, we do not report statistics about this dataset here.

Language models

The provided language modeling data, according to the organizers, consists of more than 110 million words from the Aljazeera[8] website, collected between 2004 and 2011. The provided normalized and transliterated language modeling data contains 4 717 873 sentences, corresponding to 121 277 408 words. The automatic transcriptions of Arabic broadcast speech were also available for LM training.

We use 2-gram and 4-gram LMs, built using SRILM toolkit [Stolcke et al., 2002] by the LIUM. The description of these LMs are given in [Tomashenko et al., 2016f]. For these LMs, modified Knesser-Ney discounting was applied. The vocabulary size is 300K.

5.5 Summary for all datasets

Summary information for all datasets, used in this thesis, is given in Table 5.3.

[8]http://www.aljazeera.net

Table 5.3 Summary statistics for all datasets

Corpus	**WSJ**		**TED-LIUM**			**STC**		**Arabic MGB-2016**
Language	English		English			Russian		Arabic
Channel	Microphone							
Type	Reading		Lectures			Reading		Multi-genre broadcasts
Training								
Duration (speech + pause), hours	15 (13+2)		172			27		648
Speakers (male + female)	83 (42+41)		1029 (710+319)			203 (111+92)		14 249
Test								
Adaptaion / Dev./ Test sets	si_et_05	si_et_20	Dev.	Test$_1$	Test$_2$	Adapt.	Test	Dev.
Mean duration per speaker, minutes	3.5	3.5	15	15	21	5	11.5	3.8
Speakers (male + female)	8 (5+3)		14 (7+7)	14 (7+7)	14 (10+4))	20 (10+10)		132
Words	5 353	5 645	36 672	35 555	51 452	-	19K	57 647
Vocabulary, words	5K	20K	150K and 152K			-	11K	303K

Chapter 6

GMM framework for neural network AMs

In this chapter we proposed a GMM framework for adaptation of DNN AMs. It is based on paper [Tomashenko and Khokhlov, 2014b].

6.1 Introduction

The main purpose of introducing the GMM framework is to transfer GMM adaptation techniques to DNN AMs.

On the one hand, GMM-HMM AMs have a long history: since 1980s they have been used in speech recognition and, as we saw in Section 4.2, speaker adaptation is a well-studied field of research for these models. Many very effective adaptation algorithms have been developed for GMM AMs. On the other hand, DNNs have achieved big advances in ASR over the past 3-6 years, and nowadays DNNs show higher performance than GMMs for different ASR tasks. Neural networks today are state-of-the-art of acoustic modeling. However, speaker adaptation is still a very challenging task for these models.

One of the motivations for the current research, described in this thesis, is the fact that many adaptation algorithms that work well for GMM systems cannot be easily applied to DNNs. Except for some feature normalization techniques, such as VTLN or CMVN (Section 4.1.2), only fMLLR adaptation technique (Section 4.2.2.2), originally developed for GMM AMs, has also become a widespread adaptation technique for DNN AMs (Section 4.3.7). However, fMLLR adaptation is limited to only one feature-space transformation. Thus, this algorithm performs well, when a small amount of adaptation data is available, but when the amount of adaptation data increases, this method is known to saturate, and does not make use of all the available adaptation data, unlike, for example, Bayesian methods. Therefore, there is no universal method for efficient transfer of all adaptation algorithms from the GMM framework to DNN models.

Another important aspect, that should be taken into account when comparing GMM and neural network AMs, is the different nature of these models. Hence, GMM and DNN models may be complementary, and ASR systems may benefit from their combination. In this thesis, we aim to take advantage of the robust adaptability of GMM AMs and existing adaptation methods developed for them, and apply these methods to DNN AMs.

6.2 Hybrid DNN-HMM systems with GMM-derived features

As we saw in Section 3.3.2.1, in a conventional GMM-HMM ASR system, the state emission log-likelihood of the observation feature vector $\mathbf{o}_t$ for a certain tied state s_i of HMMs is modeled as (see Formula (3.13)):

$$\log P(\mathbf{o}_t \mid s_i) = \log \sum_{m=1}^{M} \omega_{im} \mathcal{N}_{im}(\mathbf{o}_t \mid s_i), \tag{6.1}$$

where M is the number of Gaussian mixtures in the GMM for state s_i and ω_{im} is the mixture weight.

In a DNN-HMM system, outputs of a DNN are the state posteriors $P(s_i|\mathbf{o}_t)$, which are transformed for decoding into pseudo (or scaled) log-likelihoods as follows (see Formula (3.17)):

$$\log P(\mathbf{o}_t \mid s_i) = \log \frac{P(s_i \mid \mathbf{o}_t) P(\mathbf{o}_t)}{P(s_i)} \propto \log P(s_i \mid \mathbf{o}_t) - \log P(s_i), \tag{6.2}$$

where the state prior $P(s_i)$ can be estimated from the state-level forced alignment on the training speech data, and probability $P(\mathbf{o}_t)$ is independent on the HMM state and can be omitted during the decoding process.

We propose to use features, derived from a GMM model, in order to train a DNN model. Further in this thesis, we will refer to a GMM model, used for feature extraction, as an *auxiliary GMM model*, and to the obtained features – as *GMM-derived (GMMD) features*.

As we mentioned before, the construction of GMMD features for DNNs is mainly motivated by two factors. First, in the past it was shown [Ellis and Reyes-Gomez, 2001;

Pinto and Hermansky, 2008; Swietojanski et al., 2013] that NN and GMM models may be complementary and their combination can provide an additional improvement in ASR performance. Secondly, this type of features makes it possible to use various GMM-HMM adaptation algorithms in the DNN framework.

The GMMD features are obtained as follows (see Figure 6.1). Firstly, acoustic feature vectors are extracted from the speech signal. It can be spectral features (Section 3.2.1), such as MFCCs, PLPs, or NN-based features (Section 3.2.2), such as BN features (Section 3.2.2.2) or other types. After that, cepstral mean normalization (CMN) can be applied to the extracted features. Then, an auxiliary HMM-GMM model is used to transform acoustic feature vectors into likelihood-based feature vectors. This auxiliary HMM-GMM can be trained in a standard way with ML objective function (Section 3.3.3) and with triphone or monophone states as basic units.

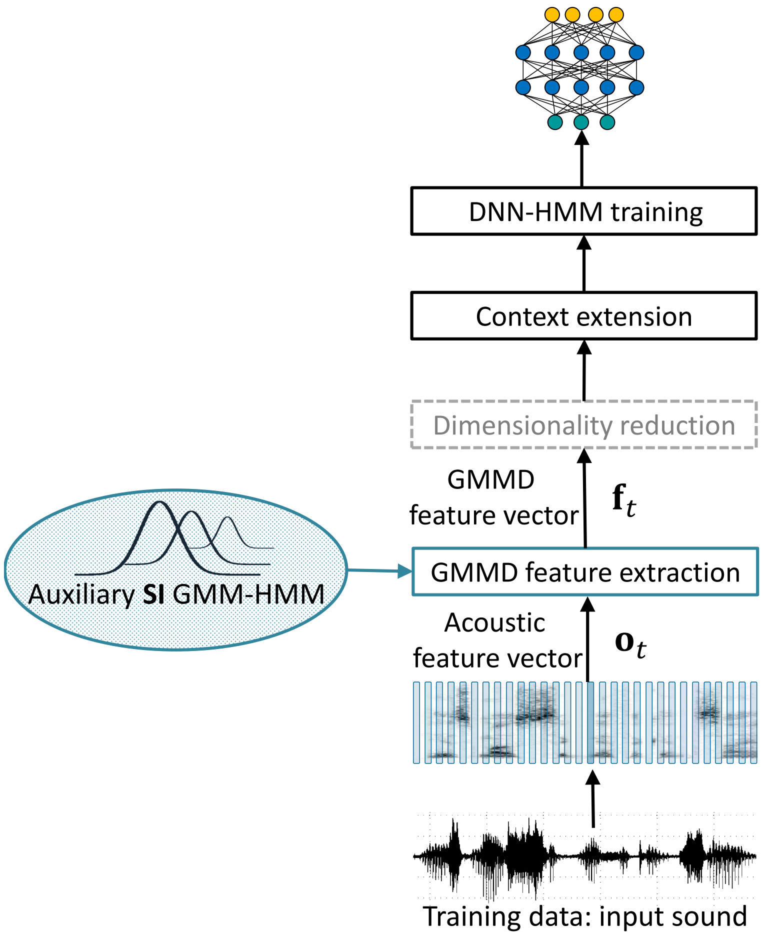


Figure 6.1 Use of GMMD features for training DNN-HMM model

For a given acoustic feature vector, a new GMMD feature vector is obtained by calculating likelihoods across all the states of the auxiliary GMM model on the given vector. Suppose $\mathbf{o}_t$ is the acoustic feature vector at time t, then the new GMMD feature vector $\mathbf{f}_t$ is calculated as follows:

$$\mathbf{f}_t = [p_t^1, \dots, p_t^n], \tag{6.3}$$

where n is the number of states in the auxiliary GMM-HMM model,

$$p_t^i = \varphi(P(\mathbf{o}_t \mid s_t = i)) \tag{6.4}$$

is the function of likelihood estimated using the GMM-HMM. Here s_t denotes the state index at time t. The natural choice of φ is a logarithmic function (as in [Pinto and Hermansky, 2008; Tomashenko and Khokhlov, 2015; Tomashenko et al., 2016b]) or an identity function (as in [Tomashenko and Khokhlov, 2014b]).

The dimension of the obtained likelihood-based feature vector $\mathbf{f}_t$ is determined by the number of states in the auxiliary GMM-HMM AM. At this step, we can also reduce the dimension of $\mathbf{f}_t$ using PCA, LDA, HLDA or other analysis. This step is not necessary, as shown in Figure 6.1. The dimensionality reduction can be required to obtain a sufficient dimension of the input layer of a DNN.

After that, the features are spliced in time by taking a context of several frames (typically $2T+1$ frames (i.e. $[-T..T]$[1]), where $5 \leq T \leq 15$).

The obtained GMMD features are used to train a DNN-HMM AM. A DNN model can be trained either directly on GMMD features, as shown in Figure 6.1, and as it was done in [Tomashenko and Khokhlov, 2014b, 2015], or on combination of GMMD with other conventional features, as in [Tomashenko et al., 2016a,b].

6.3 DNN adaptation using GMMD features

Speaker adaptation of a SI DNN-HMM model built on GMMD features is performed through the adaptation of the auxiliary SI GMM-HMM model, that was used for GMMD feature extraction. The adaptation of the auxiliary SI GMM-HMM model can be done by any adaptation algorithm developed for GMM-HMM AMs (Section 4.2). In this thesis we will investigate the application of the most popular speaker adaptation algorithms – MAP (Section 4.2.1) and MLLR (Section 4.2.2) for DNN AM adaptation.

[1]Notation $[-T..T]$ for feature vector $\mathbf{o}_t$ means, that for this vector a new context vector $\widehat{\mathbf{o}}_t$ is obtained through concatenation of neighboring vectors: $\widehat{\mathbf{o}}_t = [\mathbf{o}_{t-T}, ..., \mathbf{o}_t, ..., \mathbf{o}_{t+T}]$

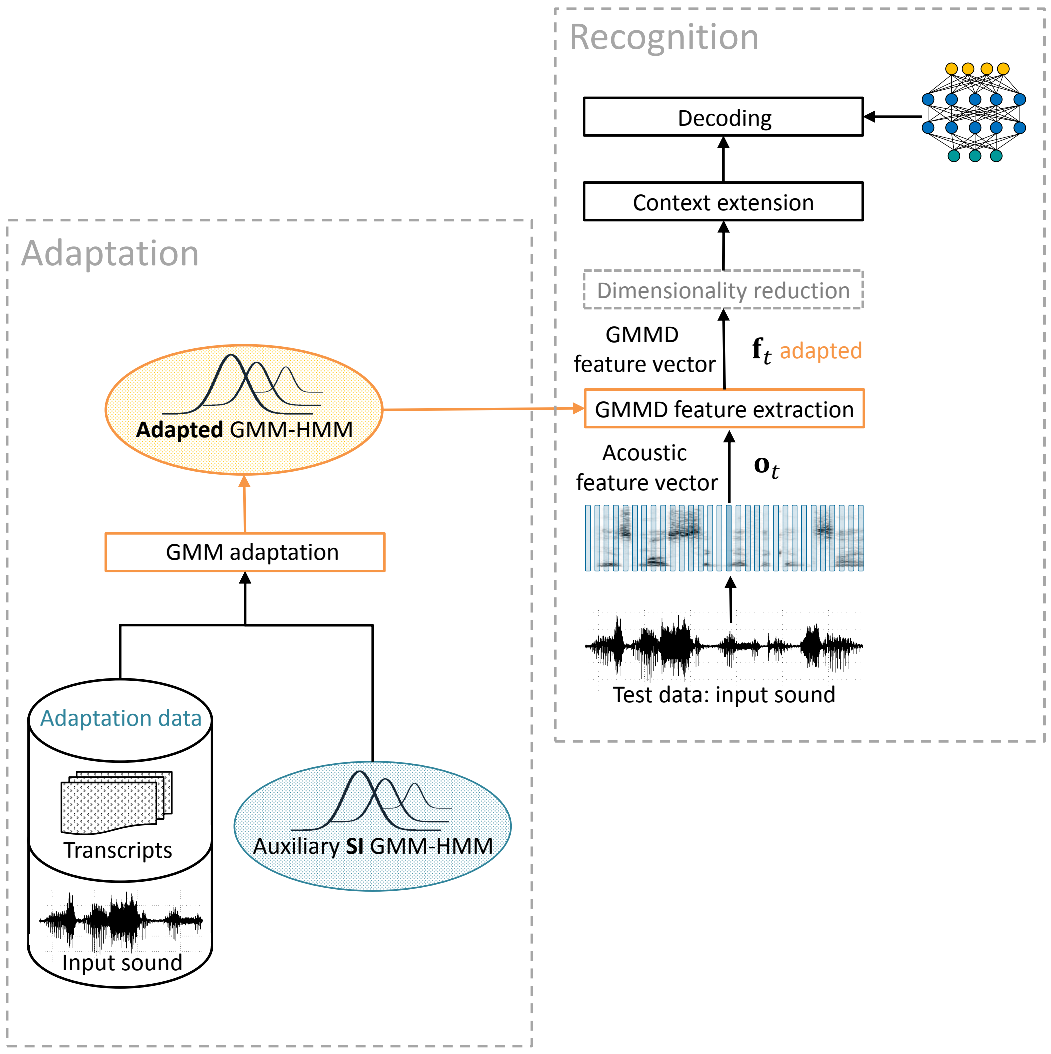


Figure 6.2 Adaptation scheme for a DNN trained on GMMD features. Adaptation of a DNN is performed through the adaptation of an auxiliary GMM.

The proposed adaptation scheme for a DNN model is shown in Figure 6.2. First, adaptation of an auxiliary SI GMM-HMM model is performed and a new speaker-adapted (SA) GMM-HMM model is created. Second, at the recognition stage, GMMD features are calculated using this SA GMM-HMM. The proposed approach can be considered as a feature space transformation technique with respect to DNN-HMMs trained on GMMD features.

6.4 Preliminary experimental results with supervised adaptation

In this section preliminary results for the proposed approach are presented. The adaptation experiments were conducted in a supervised mode (Section 4.1.1).

6.4.1 Baseline system

The experiments were conducted on STC corpus (Section 5.3). We used 11×13MFCC features (13-dimensional MFCC spliced across 11 frames ($[-5..5]$) as baseline features for training baseline DNN.

For GMMD feature extraction we first built an auxiliary speaker-independent (SI) GMM-HMM monophone model. Acoustic features in Figure 6.1 in this series of experiments are 39-dimensional MFCC$+\Delta+\Delta\Delta$ features, extracted from speech signal with a frame shift of 10 ms and window length of 16 ms. Features were normalized using CMN. In the auxiliary HMM-GMM, each of 52 Russian phonemes was modeled using a 3-state left-right context-independent HMM, with 30 Gaussians per state. The silence model was a 1-state HMM. The total number of states in the auxiliary HMM-GMM was equal to 157 – that is the dimension of $\mathbf{f}_t$ (n in Formula (6.3)). The function ϕ from Formula (6.4) was an identity function in these experiments. That means, that the coordinates of feature vector $\mathbf{f}_t$ were likelihoods:

$$p_t^i = P(\mathbf{o}_t \mid s_t = i) \tag{6.5}$$

We extracted two types of GMMD features (with and without dimensionality reduction):

- *11×157GMMD* – 1 727-dimensional features, obtained after splicing $\mathbf{f}_t$ feature vectors with a context of 11 frames: $[-5..5]$.
- *11×55GMMD* – 550-dimensional features. In this case, before splicing, the dimension of $\mathbf{f}_t$ was reduced from 157 to 55 using PCA.

The three SI-DNN models corresponding to these three types of features (11×13MFCC, 11×55GMMD and 11×157GMMD) had identical topology (except for the dimension of the input layer) and were trained on the same training corpus. The auxiliary SI monophone GMM was also trained on the same data.

The SI CD-DNN-HMM systems used 1000-neuron hidden layers, and a 2008-neuron output layer. 2008 neurons in the output layer correspond to context-dependent states determined by tree-based clustering in the CD-GMM-HMM system. ReLU nonlinearities (Formula (2.6)) were used in the hidden layers. The DNN system was trained using the frame-level cross-entropy (CE) criterion and the senone alignment generated from the triphone GMM system. The output layer was a softmax layer. DNNs were trained without layer-by-layer pre-training, and dropout with hidden dropout factor $(HDF = 0.2)$ was applied during training as a regularization (Section 2.3).

In order to explore trends in the behavior of SI and adapted models trained on the proposed GMMD features for DNNs of different depths, two DNN-HMMs were trained for each experiment, with 2 and 4 hidden layers.

An 11k vocabulary was used in evaluating both SI and adapted models. No language model was used. All described experiments were conducted on the same test set, which consists of 2 395 utterances.

The performance results in terms of WER (Formula (3.32)) for SI DNN-HMM models are presented in Table 6.1. We can see that DNN-HMMs trained on 11×13MFCC perform better than DNN-HMMs trained either on 11×55GMMD or 11×157GMMD features. In Section 6.4.3, we explore this issue in more detail and propose ways to improve the quality of SI models.

Table 6.1 Baseline WER results for SI-DNN models with 2 and 4 hidden layers for three types of features on the STC corpus. WERs for baseline systems are given with confidence intervals, corresponding to 5% level of significance.

Features	Number of hidden layers	WER, %
11×13MFCC	2	39.7 ± 0.70
	4	38.0 ± 0.69
11×157GMMD	2	40.8
	4	40.8
11×55GMMD	2	42.0
	4	41.6

6.4.2 Adaptation for DNN

The adaptation experiments were conducted on data from 20 speakers (10 male and 10 female speakers) excluded from the training set (see Section 5.3 for details). The parameter τ in the MAP adaptation formula (4.2) is set to 5. Adaptation experiments are carried out in a supervised mode.

6.4.2.1 Supervised adaptation performance for different DNN-HMMs

The adaptation results for four DNN-HMMs are presented in Table 6.2. We can see that adaptation improves SI models by approximately 14-15% absolute, corresponding to 34-36% relative WER reduction (Δ_{rel}WER) (Formula 3.34). The amount of adaptation data in these experiments was 5 minutes per each speaker.

Table 6.2 Supervised adaptation performance for different DNN-HMMs for the STC corpus. Adaptation is performed using all the available adaptation data (5 minutes per each speaker). Absolute and relative WER reduction (Δ_{abs}WER and Δ_{rel}WER) for adapted AMs is calculated with respect to the corresponding SI AMs.

Features	Number of hidden layers	WER, % for adapted AM	Δ_{abs}WER, %	Δ_{rel}WER, %
11×157GMMD	2	26.4	14.3	35.2
	4	26.9	13.9	34.1
11×55GMMD	2	26.8	15.2	36.2
	4	27.1	14.5	34.8

6.4.2.2 Adaptation performance depending on the amount of the adaptation data

In this experiment we measure the performance of the proposed adaptation method using different amounts of adaptation data. Adaptation sets of different size, from 5 seconds to 5 minutes (for a speaker), are used to adapt a SI 4-hidden-layer DNN-HMM trained on 11×157GMMD features. The results are shown in Figure 6.3. We can see that even with 5 seconds of adaptation data, we can achieve a performance gain of about 2% absolute (5% relative) WER reduction. After using all available 300 seconds of adaptation data, the gain from the adaptation increases up to approximately 14% absolute (34% relative) WER reduction.

6.4.3 Fusion

As we observed in the experiments described above, the proposed adaptation technique provides a significant improvement in the ASR performance. However, from the practical point of view, it can also be useful to have a strong SI baseline for the ASR system , for example, in case we do not have adaptation data for some speakers. Also, a stronger SI model (according to WER) can potentially provide a better result than a weaker model when they are adapted.

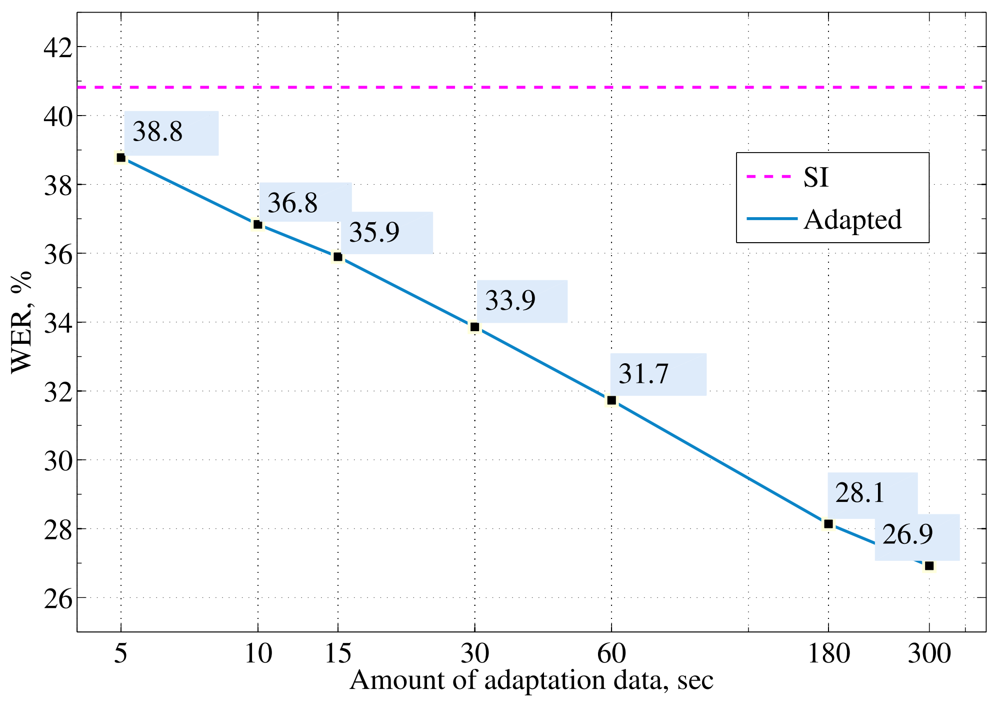


Figure 6.3 Supervised adaptation performance depending on the amount of the adaptation data for the DNN trained on 11×157GMMD features for the STC corpus.

As seen from the results described above (Table 6.1), DNN-HMMs trained on 11×13MFCC perform better than DNN-HMMs trained on either 11×55GMMD or 11×157GMMD features. For the SI 2-hidden-layer DNN, the absolute difference in WER is 2.3% for 11×55GMMD and 1.1% for 11×157GMMD as compared to 11×13MFCC features. As the number of layers in DNNs increases from 2 to 4, the differences in quality between the SI DNN trained on 11×13MFCC features and SI DNNs trained on 11×55GMMD and 11×157GMMD features increase up to 3.6% and 2.8% respectively. The purpose of this set of experiments was to find out whether it is possible to improve the performance of SI systems by applying different multi-stream combination techniques, so-called *late* and *early integration* [Pinto and Hermansky, 2008]. More advanced analysis for fusion techniques and ways of integration of GMMD features into the state-of-the-art ASR system architecture is presented in Chapter 8.

6.4.3.1 Feature-level fusion

In this experiment, 11×13MFCC and 11×55GMMD features are concatenated, resulting in 693-dimension feature vectors, and new DNN SI models are trained. The results with feature concatenation are presented in Table 6.3. We can see that this type of features helps to reduce WER for the SI model compared to the baseline SI model for 11×55GMMD. In addition, the adaptation performance with feature concatenation is slightly better than with 11×55GMMD for the 4-hidden-layer DNN, while for the 2-hidden-layer DNN it remains unchanged. However, the performance of feature concatenation without adaptation is still worse than 11×13MFCC, especially for the deeper network.

Table 6.3 Feature and posterior level fusion results on the STC corpus for 11×13MFCC and 11×55GMMD features. Adaptation is performed in a supervised mode using all the available adaptation data (5 minutes per each speaker).

Fusion type	Number of hidden layers	WER, % for SI AM	WER, % for adapted AM
Features	2	39.8	26.8
	4	39.4	26.3
Posteriors	2	38.1	27.1
	4	37.2	26.6

6.4.3.2 Posterior-level fusion

Finally, we perform state-level system fusion experiments. The outputs of the two classifiers can be combined using various multi-stream combination techniques [Kittler et al., 1998] such as sum (or linear combination), product, maximum, minimum, etc. In this work results are shown for the mean of outputs of two DNNs, as this combination gave the best result in preliminary experiments.

System fusion is performed at the state-level for every frame. The posterior fusion results are summarized in Table 6.3. Combining scores from DNNs trained on 11×13MFCC and 11×55GMMD features achieves 1.6% and 0.8% absolute WER improvement for SI DNNs as compared to DNNs trained on 11×13MFCC features for DNNs with 2 and 4 hidden layers respectively. Adaptation for posterior-level fusion gives about 30-32% relative WER reduction compared to SI DNNs trained on 11×13MFCC features.

Thus we see that posterior-level fusion of the two SI-DNN-HMMs gives better results than the feature concatenation approach. Moreover, posterior-level fusion helps to improve the quality of the baseline SI-system which is used for adaptation. This type of fusion gives an improvement of about 2-4% of relative WER reduction over the single best stream. This result can probably be further improved by using other fusion techniques.

6.5 Conclusions

In this chapter we proposed a novel feature-space transformation method for the adaptation of DNN-HMM models [Tomashenko and Khokhlov, 2014b]. The proposed method is based on a special method of feature processing for DNNs. Input features for DNN are generated from the likelihoods of the GMM model. The described method of feature processing is effective for training DNN-HMMs when used in combination with other conventional features, such as 11×13MFCC. According to the experiments, the best performance is obtained by the posterior-level fusion technique which gives approximately 2-4% relative WER reduction compared to the single best stream (the baseline DNN trained on 11×13MFCC).

The main advantage of these GMMD features is the possibility of performing the adaptation of a DNN-HMM model through the adaptation of the auxiliary GMM-HMM. We investigate the performance of MAP adaptation for this scheme. Experiments demonstrate that MAP adaptation is very effective and, for single-stream systems, gives, on average, approximately 35-36% relative WER reduction for a 5 minute adaptation sample and 5% relative WER reduction for a 5 second adaptation sample, as compared to a SI-DNN model.

If we compare adaptation results in a multi-stream recognition system, we see that the performance improvement from the adaptation for a 5 minute adaptation set is 11% absolute (29% relative) WER as compared to the SI system.

It is worth noting that in the proposed scheme, other methods for the adaptation of the auxiliary GMM-HMM can be used instead of MAP adaptation. Thus, this approach opens new perspectives in the adaptation of DNN-HMMs, since it allows us to use approaches developed for GMM-HMM, as well as SAT (that will be the topic of the following chapter), in DNN-HMM adaptation.

Chapter 7

Speaker adaptive training

In this chapter we extend the scheme for GMM-derived (GMMD) feature extraction and apply the concept of speaker adaptive training (SAT) to DNNs, trained on GMMD features. Also several techniques for adaptation performance improvement are proposed and explored: using lattices scores in maximum a posteriori (MAP) adaptation, data augmentation and data selection techniques. This chapter is based on papers [Tomashenko and Khokhlov, 2015] and [Tomashenko et al., 2016d].

7.1 SAT DNN AM training using GMMD features

7.1.1 Proposed approach to SAT-DNN

In this section we improve the previously proposed scheme for GMMD feature extraction and apply the concept of SAT to DNNs, trained on GMMD features. In SAT, speaker adaptation is performed for the training data, independently for each training speaker, so that all the training data are projected into speaker-adaptive feature space. Then, the parameters of the SAT-DNN AM are estimated in this feature space.

The scheme of SAT-DNN training is shown in Figure 7.1. In contrast to the previous approach, explored above (Section 6.2 and Figure 6.1), for each training speaker we extract speaker-adapted GMMD features, and train SAT-DNN AM on them.

The procedure for building SAT-DNN can be summarized as follows:

1. Train an auxiliary SI GMM-HMM $\mathbf{\Lambda}$ over the training data.

2. Adapt $\mathbf{\Lambda}$ for the training speakers, so that for each speaker S in the training corpus a speaker-adapted GMM-HMM $\mathbf{\Lambda}(S)$ is created.

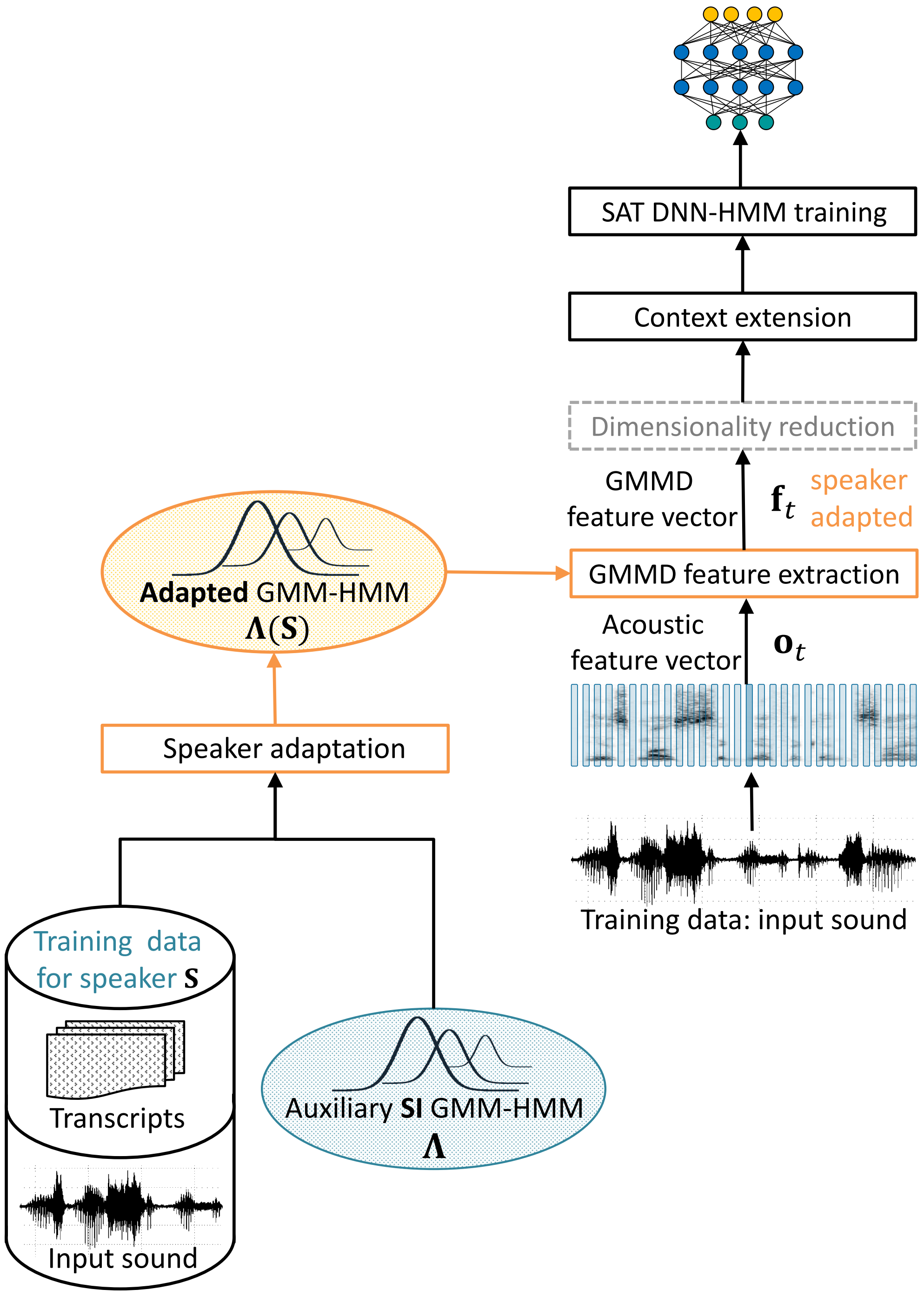


Figure 7.1 SAT scheme for DNN training with GMMD features.

3. For each speaker in the training corpus, transform all acoustic feature vectors $\mathbf{o}_t$ of the training corpus, belonging to speaker S, into likelihood based feature vectors $\mathbf{f}_t$ using speaker-adapted auxiliary AM $\mathbf{\Lambda}(S)$ and Formulas (6.3) and (6.4).

4. Calculate GMMD features from speaker-adapted $\mathbf{f}_t$.

5. Train DNN AM on the obtained speaker-adapted GMMD features.

During the test, the adaptation is performed in the same manner, as described in Section 6.3. In this chapter, and further in the following chapters, we use log-likelihoods in Formula (6.4), so that

$$p_t^i = \log P(\mathbf{o}_t \mid s_t = i) \tag{7.1}$$

7.1.2 Experimental results with SAT and unsupervised MAP and fMLLR adaptation

7.1.2.1 Baseline system

The experiments were conducted on the WSJ0 corpus (Section 5.1).

We used conventional 11×39MFCC features, composed of 39-dimensional MFCC (with CMN) spliced across 11 frames $[-5..5]$, as baseline features and compared them to the proposed GMMD features. The two SI-DNN models corresponding to these two types of features, 11×39MFCC and 11×118GMMD, had identical topology (except for the dimension of the input layer) and were trained on the same training dataset. An auxiliary monophone GMM was also trained on the same data. For training speaker independent (SI) DNN on GMMD features, we applied the scheme shown in Figure 6.1.

The SI CD-DNN-HMM systems used four 1000-neuron hidden layers and an output layer with approximately 2500 neurons. The neurons in the output layer corresponded to the context-dependent states determined by tree-based clustering in CD-GMM-HMM. Rectified linear units (ReLUs) were used in the hidden layers. The DNN system was trained using the frame-level CE criterion and the senone alignment generated from the GMM system. The output layer was a softmax layer. DNNs were trained without layer-by-layer pre-training, and hidden dropout factor ($HDF = 0.2$) was applied during the training as a regularization.

Evaluation was carried out on the two standard WSJ0 evaluation tests: (1) *si_et_05* and (2) *si_et_20*.

7.1.2.2 SAT-DNN AMs

We trained three SAT DNNs on GMMD features, as shown in Figure 7.1. For adapting an auxiliary GMM model we used two different adaptation algorithms: MAP and fMLLR. In addition, we trained a SAT DNN on GMMD features with "fMLLR+MAP" configuration, where MAP adaptation of an auxiliary GMM model was performed after fMLLR adaptation. For comparison purposes we also trained a DNN on conventional 11×39MFCC features with fMLLR. All SAT DNNs had similar topology and were trained as described in Section 7.1.2.1 for SI models. Training GMM-HMMs and fMLLR adaptation was carried out using the Kaldi speech recognition toolkit [Povey et al., 2011b].

7.1.2.3 Adaptation performance for different DNN-HMMs

Unless explicitly stated otherwise, the adaptation experiments were conducted in an unsupervised mode on the test data using transcripts obtained from the first decoding pass. The performance results in terms of word error rate (WER) for SI and SAT DNN-HMM models are presented in Table 7.1. We can see that SI DNN-HMM trained on GMMD features performs slightly better than DNN-HMM trained on 11×39MFCC features. In all experiments we consider SI DNN trained on 11×39MFCC features as the baseline model and compare the performance results of the other models with it.

Table 7.1 WER results for unsupervised adaptation on WSJ0 evaluation sets: *si_et_20* and *si_et_05*. Δ_{rel}WER is a relative WER reduction calculated with respect to the baseline SI model. WER for the baseline system is given with confidence intervals, corresponding to 5% level of significance.

Features	Adaptation (SAT)	*si_et_20*		*si_et_05*	
		WER, %	Δ_{rel}WER, %	WER, %	Δ_{rel}WER, %
11×39MFCC	SI	9.55 ±0.77	baseline	3.23 ±0.47	baseline
	fMLLR	8.03	16.0	2.76	14.5
GMMD	SI	9.28	2.8	3.19	1.2
	MAP	**7.60**	**20.4**	2.69	16.8
	fMLLR	7.85	17.8	**2.52**	**22.0**
	fMLLR+MAP	7.83	18.0	2.78	13.9

The SAT DNN trained on GMMD features with MAP adaptation demonstrates the best result over all cases in the dataset *si_et_20*. It outperforms the baseline SI DNN model trained on 11×39MFCC features and results in 20.4% relative WER reduction (Δ_{rel}WER, see Formula 3.34). In the dataset *si_et_05* the SAT DNN trained on GMMD features with fMLLR adaptation performs better than other models and gives 22.0% of Δ_{rel}WER. Moreover, we can see that in all experiments SAT models trained on GMMD features, with fMLLR as well as with MAP adaptation, perform better in terms of WER than the SAT model trained on 11×39MFCC features with fMLLR adaptation.

From the last row of Table 7.1, which shows the results of combining MAP and fMLLR algorithms, we can conclude that this combination does not lead to additional improvement in performance, and for the test *si_et_05*, it even degrades the result.

The results, presented in Tables 7.2 and 7.3, demonstrate adaptation performance for different speakers from the *si_et_20* and *si_et_05* datasets, correspondingly. The bold figures in the tables indicate the best performance over the three acoustic models. Relative WER reduction (Δ_{rel}WER) is given in comparison to the baseline model, SI DNN built on 11×39MFCC. We can observe that all the three models behave differently depending on the speaker.

Table 7.2 Performance for unsupervised speaker adaptation on the WSJ0 corpus for speakers from *si_et_20* evaluation set

Speaker id	WER, % for SI, 11×39MFCC	Δ_{rel}WER, %		
		11×39MFCC fMLLR	GMMD fMLLR	GMMD MAP
440	8.49	**24.1**	22.4	6.9
441	16.02	24.3	26.2	**36.4**
442	10.59	13.2	**15.8**	14.5
443	9.94	11.3	**14.1**	9.9
444	10.87	11.8	18.4	**26.3**
445	7.25	13.0	**20.4**	18.5
446	7.13	8.5	**17.0**	12.8
447	6.59	16.0	0.0	**26.0**
All	9.55	16.0	17.8	**20.4**

7.1.2.4 Adaptation performance depending on the size of the adaptation dataset

In this experiment we measured the performance of the proposed adaptation method using different amounts of adaptation data. Adaptation sets of different sizes, from 15 to 210 seconds of speech data (per speaker), were used to adapt SAT DNN-HMM models trained on 11×39MFCC and GMMD features.

The results are shown in Figure 7.2. Relative WER reduction is given with respect to the baseline SI DNN trained on 11×39MFCC. We can see that, for adaptation sets of different size, the SAT DNN trained on GMMD features with fMLLR perform better than the SAT DNN trained on 11×39MFCC features with fMLLR.

Table 7.3 Performance for unsupervised speaker adaptation on the WSJ0 corpus for speakers from *si_et_05* evaluation set

Speaker id	WER, % for SI, 11×39MFCC	Δ_{rel}WER, %		
		11×39MFCC fMLLR	GMMD fMLLR	GMMD MAP
440	4.15	29.6	**40.7**	**40.7**
441	4.71	43.3	**56.7**	16.7
442	3.32	0	8.3	**12.5**
443	1.52	-10.0	**0**	-20
444	2.43	**33.3**	22.2	22.2
445	3.16	0.0	5.3	**15.8**
446	1.89	-15.2	-7.6	**23**
447	4.87	6.3	**15.6**	6.3
All	3.23	14.5	**22**	16.8

In contrast, MAP adaptation reaches the performance of fMLLR adaptation (for MFCC) only when using all the adaptation data. However, the gain from MAP adaptation grows monotonically as the sample size increases, while fMLLR adaptation reaches saturation on a small adaptation dataset. The same behavior of MAP and fMLLR adaptation algorithms is known for GMM-HMM AMs.

We conducted an additional experiment with MAP adaptation, marked in Figure 7.2 with the text *"using extra data"*, in which we added the data from the WSJ0 *si_et_ad* dataset to the adaptation data. Hence, in this case we performed the adaptation in a semi-supervised mode: the transcriptions for the *si_et_ad* dataset were supposed to be known and the transcriptions for the *si_et_05* were generated from the first decoding pass. The total duration of the adaptation data was approximately 380 seconds for each speaker. This result (21.4% of relative WER reduction) confirms the suggestion that the performance of MAP adaptation did not reach its maximum in the previous experiments.

7.1.3 Discussion

In this section we improved the previously proposed adaptation algorithm by applying the SAT concept to DNNs built on GMMD features and by using fMLLR-adapted features for training an auxiliary GMM model. Traditional adaptation algorithms, such as MAP and fMLLR were performed for the auxiliary GMM model used in a SAT procedure for a DNN.

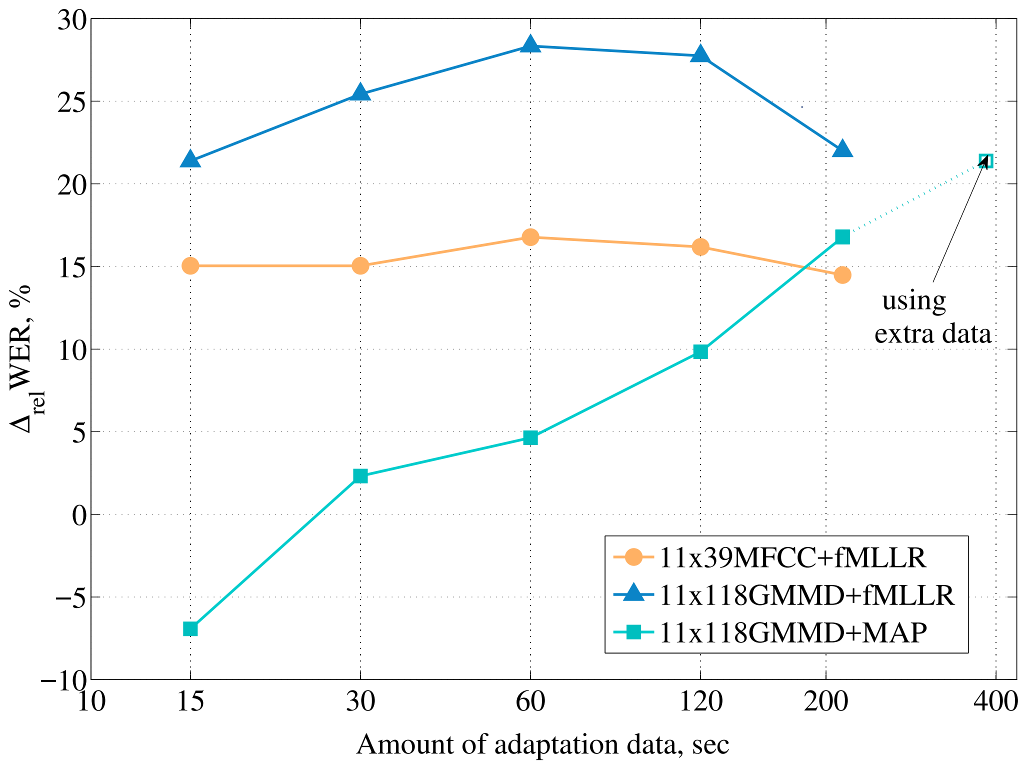


Figure 7.2 Unsupervised adaptation performance on *si_et_05* test depending on the size of the adaptation dataset. Relative WER reduction (Δ_{rel}WER) is given with respect to the baseline SI DNN trained on 11×39MFCC.

Experimental results on the WSJ0 corpus demonstrate that, in an unsupervised adaptation mode, the proposed adaptation technique can provide, approximately, a 17–20% relative WER reduction for MAP and 18–28% relative WER reduction for fMLLR adaptation on different adaptation sets, compared to the SI DNN-HMM system built on conventional 11×39MFCC features. We found that fMLLR adaptation for the SAT DNN trained on GMMD features outperforms fMLLR adaptation for the SAT DNN trained on conventional features by up to 14% of relative WER reduction. It has been shown, that fMLLR adaptation for GMMD features is efficient when using a small amount of adaptation data, while MAP adaptation works better when more adaptation data are used.

It is worth noting that in the proposed scheme, any other methods for the adaptation of the auxiliary GMM can be used instead of MAP or fMLLR adaptation. Thus, this approach provides a general framework for transferring adaptation algorithms developed for GMM-HMMs to DNN adaptation.

7.2 MAP adaptation using lattices scores

The use of lattice-based information and confidence scores [Gollan and Bacchiani, 2008] is a well-known method for improving the performance of unsupervised adaptation. In this work we use the MAP adaptation algorithm for adapting the SI GMM model. Speaker adaptation of a DNN-HMM model built on GMMD features is performed through the MAP adaptation of the auxiliary GMM monophone model, which is used for calculating GMMD features. We modify the traditional MAP adaptation algorithm by using lattices instead of alignment from the first decoding pass as follows.

Let m denote an index of a Gaussian in SI AM, and $\boldsymbol{\mu}_m$ the mean of this Gaussian. Then the MAP estimation of the mean vector is

$$\widehat{\boldsymbol{\mu}}_m = \frac{\tau \boldsymbol{\mu}_m + \sum_t \gamma_m(t) p_s(t) \mathbf{o}_t}{\tau + \sum_t \gamma_m(t) p_s(t)}, \tag{7.2}$$

where τ is the parameter that controls the balance between the maximum likelihood estimate of the mean and its prior value; $\gamma_m(t)$ is the posterior probability of Gaussian component m at time t; and $p_s(t)$ is the confidence score of state s at time t in the lattice obtained from the first decoding pass by calculating arc posteriors probabilities. The forward-backward algorithm is used to calculate these arc posterior probabilities from the lattice as follows [Evermann and Woodland, 2000; Uebel and Woodland, 2001]:

$$P(l|O) = \frac{\sum_{q \in Q_l} P_{ac}(O|q)^{\frac{1}{\lambda}} P_{lm}(w)}{P(O)}, \tag{7.3}$$

where λ is is the scale factor (the optimal value of λ is found empirically by minimizing WER of the *consensus hypothesis* [Mangu et al., 2000]); q is a path through the lattice corresponding to the word sequence w; Q_l is the set of paths passing through arc l; $P_{ac}(O|q)$ is the acoustic likelihood; $P_{lm}(w)$ is the language model probability; and $p(O)$ is the overall likelihood of all paths through the lattice. For the given frame $\mathbf{o}_t$ at time t we the calculate confidence score $p_s(t)$ as follows:

$$p_s(t) = \sum_{l \in S_s(\mathbf{o}_t)} P(l|O), \tag{7.4}$$

where $S_s(\mathbf{o}_t)$ is the set of all arcs corresponding to state s in the lattice at time t. In a particular case, when $p_s(t) = 1$ for all states and t, Formula (7.2) represents the traditional MAP adaptation (Formula (4.2)).

In addition to this frame-level weighting scheme, we apply confidence-based selection scheme, when we use in (7.2) only the observations with confidence scores exceeded the chosen threshold.

7.3 Data augmentation and data selection for SAT

In this work we explore two other approaches to improve the performance of SAT DNN models with MAP adaptation (Figure 7.3): *data augmentation* and *data selection*.

The first approach is based on using different values of parameter τ (in Formula (4.2)) when extracting adapted GMMD features for DNN training. In this approach we extract features for all training corpus several times for a set of τ values: $\{\tau_1, \tau_2, \ldots\}$. Then, the DNN model is trained on the union of the obtained features. The intuition behind this approach is similar to that used in *data augmentation* [Cui et al., 2015].

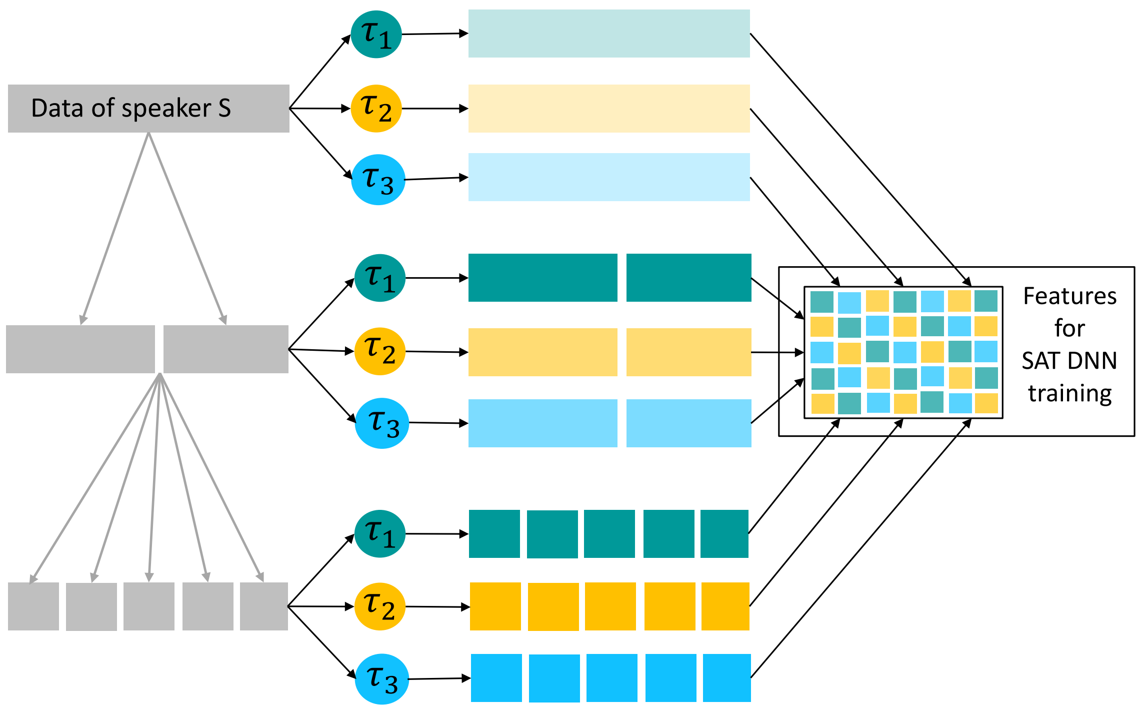


Figure 7.3 Data selection and data augmentation scheme for SAT. For each speaker S in the training corpus, training data of this speaker are divided into several parts (shown in gray). Then MAP adaptation is performed independently for these parts. At this stage, different values of parameter τ: $\{\tau_1, \tau_2, \tau_3\}$ are applied during the GMMD feature extraction for each part. In the shown example, the size of the training corpus (in terms of the features) increases in 9 times (3(selection) $\times$ 3(augmentation)).

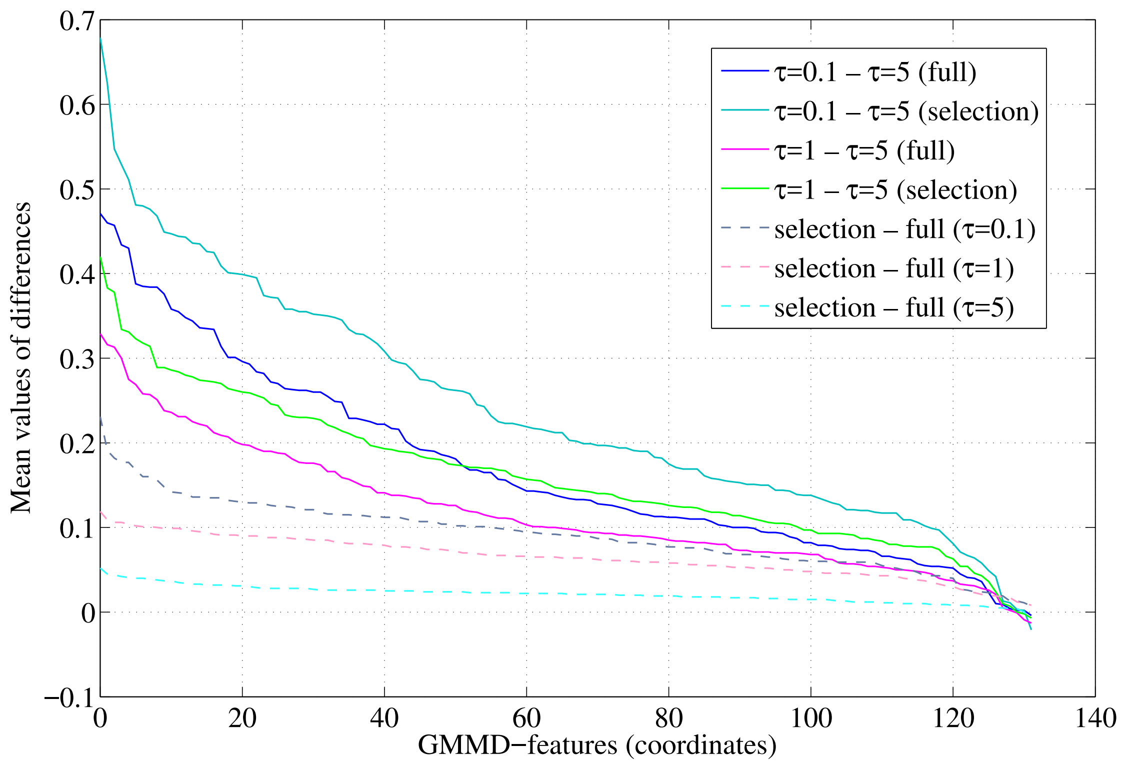
The second approach, which we call *data selection* strategy, consists in splitting training data for each speaker in the training corpus into several parts and then performing MAP adaptation independently on each of the parts. In this chapter we use a simple implementation of this strategy – we randomly separate training data for each speaker into several subsets, so that the total amount of data in each subset is approximately equal to the average amount of data per speaker in the test set. This strategy serves as a regularization and is supposed to make adaptation more robust to the size of the adaptation set.

Hence, the original data from the training corpus are used in AM training several times with different values of τ and inside different subsets of data chosen for adaptation. The motivation for these two approaches lies in obtaining more robust SAT DNN models for MAP adaptation, especially when the training corpus is relatively small.

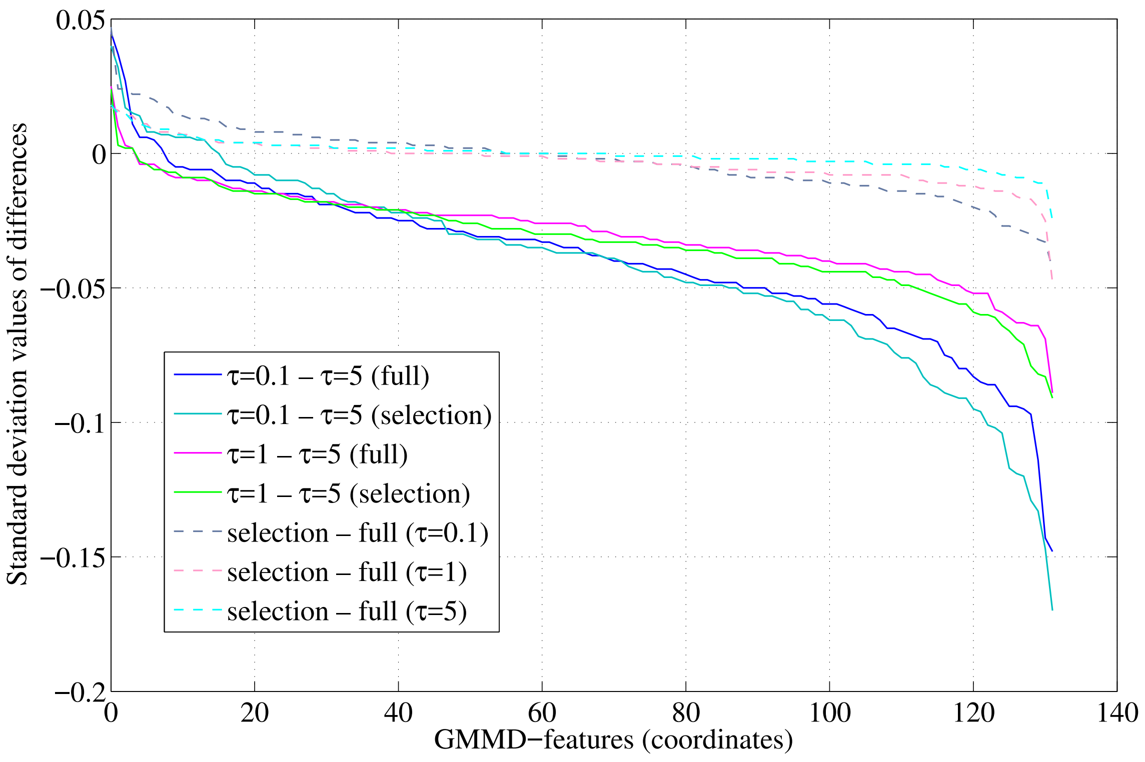
The GMMD feature dynamic in the training corpus for different values of τ and for different data selection strategies is shown in Figure 7.4. In both pictures "full" means that during the SAT training for a given speaker all data of that speaker from the training corpus are used for MAP adaptation, whereas "selection" means that data selection strategy is applied and training data for this speaker is randomly split into two subsets so that MAP adaptation is performed for each subset independently. Let denote T_1 and T_2 two types of features, (or more precisely, GMMD features extracted with different parameters). Every curve in Figure 7.4a and 7.4b, marked as "T_1–T_2", corresponds to the average differences between T_1 and T_2 features and is calculated as follows. First, we subtract coordinate-wise features T_2 from T_1 on the training corpus. Then, we found mean (Figure 7.4a) and standard deviation (Figure 7.4b) values for each feature vector coordinate. Finally, we sort the obtained values for each feature vector dimension by descending order.

For example, in Figure 7.4 the first (blue) line in both pictures corresponds to "τ=0.1 – τ=5 (full)". This means in our notation, that we consider two types of GMMD features: $T_1 : \tau = 0.1$ and $T_2 : \tau = 5$, calculated on the training corpus with different τ values, and "(full)" means that in both cases features are extracted in the standard way without data selection, for each speaker using all data from this speaker available in the training corpus to perform speaker adaptation.

We can see that GMMD features calculated for various τ and with (or without) data selection strategy have different amplitude and dynamic characteristics, therefore they can contain complementary information. Hence data augmentation might improve AM by making them more robust to τ and to the size of the adaptation set.



(a) Mean values



(b) Standard deviation values

Figure 7.4 Differences in GMMD-features depending on τ values. GMMD feature coordinates are sorted by descending order of the corresponding differences. Statistics are calculated on the training dataset of the WSJ0 corpus.

7.4 Experimental results with data augmentation, data selection and lattices scores

The experiments are conducted on the WSJ0 corpus (Section 5.1). Unlike the initial experiments with SAT and GMMD features, described in Section 7.1.2, for this series of experiments we built a novel strong baseline AM using the Kaldi speech recognition toolkit [Povey et al., 2011b], following mostly Kaldi WSJ recipe (except for GMMD-features and adaptation). The new SAT-DNN AM on GMMD features was also trained according to the new recipe.

We use conventional 11×39MFCC features (39-dimensional MFCC (with CMN) spliced across 11 frames (±5)) as baseline features and compare them to the proposed GMMD features. We train four DNN models: SI model on 11×39MFCC; SI and two SAT models on GMMD features. These four DNNs have identical topology (except for the dimension of the input layer) and are trained on the same training dataset. An auxiliary GMM is also trained on the same data.

The first SAT DNN on GMMD features is trained as described in Section 7.1.1 with parameter τ for adaptation equal to 5. The second SAT DNN on GMMD features is trained using data augmentation (with τ equal to 0.1, 1 and 5) and data selection strategy, as described in Section 7.3. For training SI-DNN on GMMD features, we apply the scheme shown in Figure 6.1.

All four CD-DNN-HMM systems had six 2048-neuron hidden layers and a 2355-neuron output layer. The neurons in the output layer correspond to context-dependent states determined by tree-based clustering in CD-GMM-HMM. The DNN is initialized with the stacked restricted Boltzmann machines by using layer by layer generative pre-training. It is trained with an initial learning rate of 0.008 using the CE objective function. After that, five iterations of training with the sMBR criterion (Formula (3.3.3.1)) are performed.

In all experiments further we consider SI DNN trained on 11×39MFCC features as the baseline model and compare the performance results of the other models with it. Evaluation is carried out on the standard WSJ0 evaluation test **si_et_20**. The adaptation experiments are conducted in an unsupervised mode on the test data using transcripts or lattices obtained from the first decoding pass.

For adapting an auxiliary GMM model we use MAP adaptation algorithm. We perform two adaptation experiments:

1. with traditional MAP;

2. with lattice-based MAP using confidence scores, as described in Section 7.2.

For lattice-based MAP the value of confidence threshold is 0.6. The performance results in terms of WER for SI and adapted DNN-HMM models are presented in Table 1. We can see that using confidence scores can give an additional slight improvement in MAP adaptation for DNN models over adaptation, which uses an alignment. The best result is obtained using data augmentation and data selection strategies. For comparison purposes we also train six DNN models with τ values 0.1, 1 and 5 with and without data selection strategies, but in all cases the results are worse than the one obtained combining both strategies, so we do not report other results here.

Table 7.4 Summary of WER (%) results for unsupervised adaptation on WSJ0 evaluation set *si_et_20*. Δ_{rel}WER is a relative WER reduction calculated with respect to the baseline SI AM. The WER for the baseline system is given with a confidence interval, corresponding to 5% level of significance.

Type of Features	Adaptation	WER, %	Δ_{rel}WER, %
11×39MFCC	SI	7.51 ±0.69	baseline
GMMD	SI	7.83	–
	MAP (alignment)	7.09	5.6
	MAP (lattice-based)	6.93	8.4
	MAP (data augmentation & selection)	6.77	9.9

Experimental results demonstrate, that in an unsupervised adaptation mode, the proposed adaptation technique can provide, approximately, up to 9.9% relative WER reduction compared to the SI DNN system built on conventional 11×39MFCC features.

7.5 Conclusions

In this chapter we extended and improved the previously proposed adaptation algorithm by applying the concept SAT to DNNs built on GMM-derived features.

In the first series of experiments (Section 7.1.2), traditional adaptation algorithms, such as MAP and fMLLR were performed for the auxiliary GMM model used in a SAT procedure for a DNN [Tomashenko and Khokhlov, 2015]. Experimental results on the WSJ0 corpus demonstrate that, in an unsupervised adaptation mode, the proposed adaptation technique can provide, approximately, a 17–20% relative WER reduction for MAP and 18–28% relative

WER reduction for fMLLR adaptation on different adaptation sets, compared to the SI DNN-HMM system built on conventional 11$\times$39MFCC features. We found that fMLLR adaptation for the SAT DNN trained on GMM-derived features outperforms fMLLR adaptation for the SAT DNN trained on conventional features by up to 14% of relative WER reduction. It has been shown, that fMLLR adaptation for GMMD features is efficient when using a small amount of adaptation data, while MAP adaptation works better when more adaptation data are used.

Another contribution of this chapter consists in the way to improve the previously proposed adaptation algorithm by using confidences scores in adaptation [Tomashenko et al., 2016d]. In addition, we introduced two approaches, so called data augmentation and data selection strategies, for improving the regularization in MAP adaptation for DNN. The proposed approaches are especially suitable when the training corpus is small, or when the amount of adaptation data is not known in advance and can vary.

The second series of experiments (Section 7.4) was conducted with different DNN training recipes, criteria and different DNN topologies in comparison with the experiments described in the first part of this chapter. Particularly, all DNNs in this second series were trained with much more parameters (6 HLs$\times$2048 neurons vs. 4 HLs$\times$1000 neurons). Also, DNNs in the second series had sigmoid activation functions and were trained with RBM pre-training in contrast to the DNNs in the first series, which used ReLU activation functions and dropout. In addition, DNNs in the second series were trained with sMBR training criterion, while CE criterion was used in the first series of experiments. This resulted in more accurate (in sense of WER) models. We observed that, in these conditions, improvement from MAP adaptation decreased but is still significant. The relative WER reduction for adapted AM equals to approximately 10%, if we compare them with SI AM trained on the same features, and 6% if we compare them with SI AM trained on conventional features. These two values can be further improved by the proposed data augmentation and selection techniques and reach respectively 14% and 10% of relative WER reduction.

Chapter 8

Integration of GMMD features into state-of-the-art ASR systems

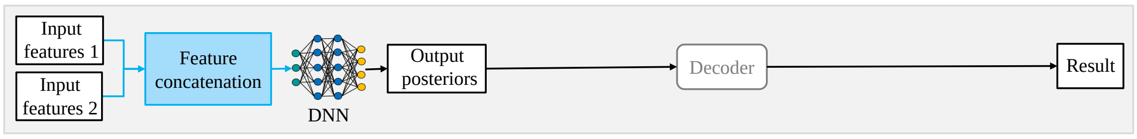
In this chapter we investigate various ways of integrating GMMD features into different neural network architectures (DNN and TDNN). It includes the results from papers [Tomashenko et al., 2016b], [Tomashenko et al., 2016a] and [Tomashenko et al., 2016f].

8.1 System fusion

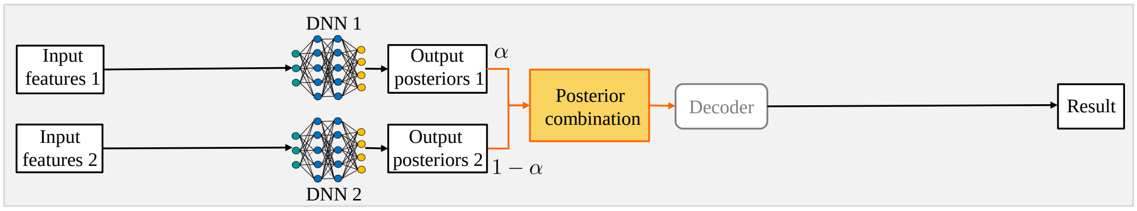
In this section we suggest several types of combination of GMMD features with conventional ones at different levels of DNN architectures. It is known that GMM and DNN models can be complementary and their combination allows to improve the performance of ASR systems [Pinto and Hermansky, 2008; Swietojanski et al., 2013]. Fusion is useful when the individual systems, used in combination, contain complementary information. To obtain better recognition performance we explore the following types of fusion: *feature-level, posterior-level, lattice-level* and others.

8.1.1 Feature-level fusion

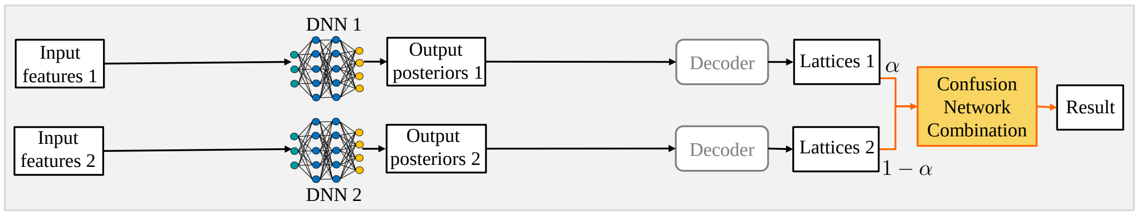
In this type of fusion, also called *early fusion* or *early integration* [Pinto and Hermansky, 2008], input features are combined before performing classification, as shown in Figure 8.1a. In our case, features of different types – GMMD features and cepstral or BN features are simply concatenated and provided as input into the DNN model for training. This type of fusion allows us to combine different adaptation techniques in a single DNN model. For example, MAP-adapted GMMD features can be concatenated with fMLLR-adapted BN



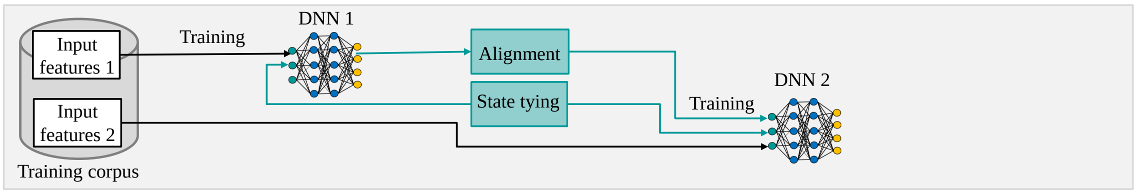
(a) Fusion for training and decoding stages.



(b) Fusion for decoding stage: posterior combination.



(c) Fusion for decoding stage: CNC.



(d) Fusion for training stage.

Figure 8.1 Types of fusion

features or i-vectors, that makes adaptation more efficient for both small and large adaptation sets.

8.1.2 Posterior-level fusion

Posterior-level fusion is also referred to as *late fusion* [Parthasarathi et al., 2015], *late integration* [Pinto and Hermansky, 2008], *state-level score combination* [Lei et al., 2013] or *explicit combination* [Swietojanski et al., 2013]. In this type of fusion (Figure 8.1b) the outputs of two or more DNN models are combined at the state level. The outputs of the two classifiers can be combined using various multi-stream combination techniques such as sum (or linear combination), product, maximum, minimum, etc. In this work we perform frame-synchronous fusion using a linear combination of the observation posteriors of two models (DNN_1 and DNN_2) as follows:

$$P(\mathbf{o}_t \mid s_i) = \alpha P_{\text{DNN}_1}(\mathbf{o}_t \mid s_i) + (1-\alpha) P_{\text{DNN}_2}(\mathbf{o}_t \mid s_i), \tag{8.1}$$

where $\alpha \in [0,1]$ is a weight factor that is optimized on a development set. This approach assumes that both models have the same state tying structure.

8.1.3 Lattice-level fusion

The highest level of fusion operates in the space of generated word hypotheses and tries to rescore or modify recognition hypotheses provided as lattices (Figure 8.1c) or n-best lists. This type of fusion is also referred to as *implicit combination* [Swietojanski et al., 2013].

One of the most common techniques for ASR system combination are *recognizer output voting error reduction* (ROVER) [Fiscus, 1997] and *confusion network combination* (CNC) [Evermann and Woodland, 2000]. In ROVER 1-best word sequences from the different ASR systems are combined into a single *word transition network* (WTN) using a dynamic programming algorithm. Based on this WTN the best scoring word is chosen among the words aligned together. The decision is based either on the voting scheme, or word confidence scores if they are available for all systems.

In CNC, instead of aligning the single best output, confusion networks built from individual lattices are aligned. In this work we experiment with the CNC approach because usually (for example, [Evermann and Woodland, 2000; Xu et al., 2011]) it provides better results than a simple ROVER scheme.

8.1.4 Other types of fusion

There are other possible ways of combining information from different ASR systems than those listed above, that also could be considered as fusion. In this chapter, in addition to those already mentioned, we also implemented the two following approaches.

The first approach is specific to the adaptation task and is related only to the acoustic model adaptation stage. It consists in using the transcripts (or lattices) obtained from the decoding pass of one ASR system in order to adapt another ASR system (*cross-adaptation*).

The second approach (Figure 8.1d) is related to the acoustic model training. It is possible to transfer some information from building one acoustic model to another one. In this work we used phoneme-to-speech alignment obtained by one acoustic DNN model to train another

DNN model. In addition we used state tying from the first DNN model to train the second DNN. This procedure is important when we want to apply *posterior fusion* for two DNNs and need the same state tying for these models. Also, several types of fusion described above can be combined.

8.2 Training DNN AMs with GMMD features

In the previous chapters (6 and 7), an auxiliary GMM model was trained on MFCC features, and then this GMM model was used to extract GMMD features for further training a DNN model. In this chapter, we investigate the effectiveness of the proposed approach on another level of DNN architecture. We use *bottleneck* (BN) features (Section 3.2.2.2) from a DNN to train a GMM model for GMMD feature extraction. The motivation for using BN features in this approach is that for better source features we can obtain better adaptation results. BN features allow us to capture long term spectro-temporal dynamics of the signal with GMMD features and are proven to be effective both for GMM and DNN acoustic model training [Grézl et al., 2007, 2014]. To confirm this suggestion, in Section 8.6.1, we will experimentally demonstrate that BN features are more effective than MFCC for GMMD feature extraction.

The scheme for training DNN models with GMM adaptation framework is shown in Figure 8.2. First, 40-dimensional log-scale filterbank features, concatenated with 3-dimensional pitch-features[1], are spliced across 11 neighboring frames [-5..5], resulting in 473-dimensional (43×11) feature vectors. After that a DCT transform is applied and the dimension is reduced to 258. Then a DNN model for 40-dimensional BN features is trained on these features. An auxiliary triphone or monophone GMM-HMM model is used to transform BN feature vectors into log-likelihoods vectors. At this step, speaker adaptation of the auxiliary speaker-independent (SI) GMM-HMM model is performed for each speaker in the training corpus and a new speaker-adapted (SA) GMM-HMM model is created in order to obtain SA GMMD features.

For a given BN feature vector, a new GMMD feature vector is obtained by calculating log-likelihoods across all the states of the auxiliary GMM model on the given vector. Suppose $\mathbf{o}_t$ is the BN feature vector at time t, then the new GMMD feature vector $\mathbf{f}_t$ is calculated as in Formulas (6.3) and (7.1).

[1]Pitch-features are calculated using the Kaldi toolkit [Povey et al., 2011b] and consist of the following values [Ghahremani et al., 2014]: (1) *probability of voicing* (POV-feature), (2) pitch-feature and (3) delta-pitch-feature. For details see http://kaldi-asr.org/doc/process-kaldi-pitch-feats_8cc.html.

The obtained GMMD feature vector $\mathbf{f}_t$ is concatenated with the original vector $\mathbf{o}_t$. After that, the features are spliced in time taking a context size of 13 frames: [-10,-5..5,10]. These features are used as the input for training a SAT DNN.

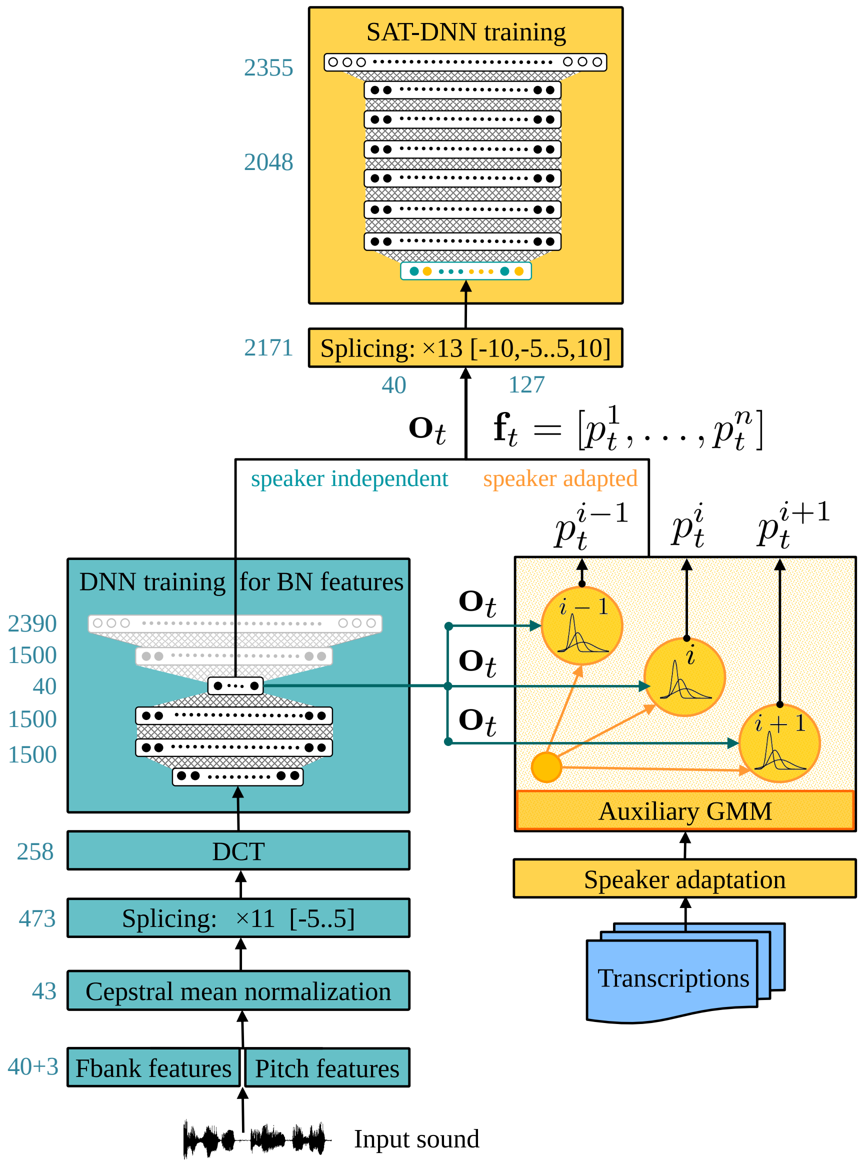


Figure 8.2 Using speaker-adapted BN-based GMMD features for SAT of a DNN-HMM. Numbers correspond to feature or layer dimensions.

8.3 Training TDNN AMs with GMMD features

In addition to the system described in Section 8.2, we aim to explore the effectiveness of using GMMD features to train a *time delay neural network* (TDNN) [Waibel et al., 1989].

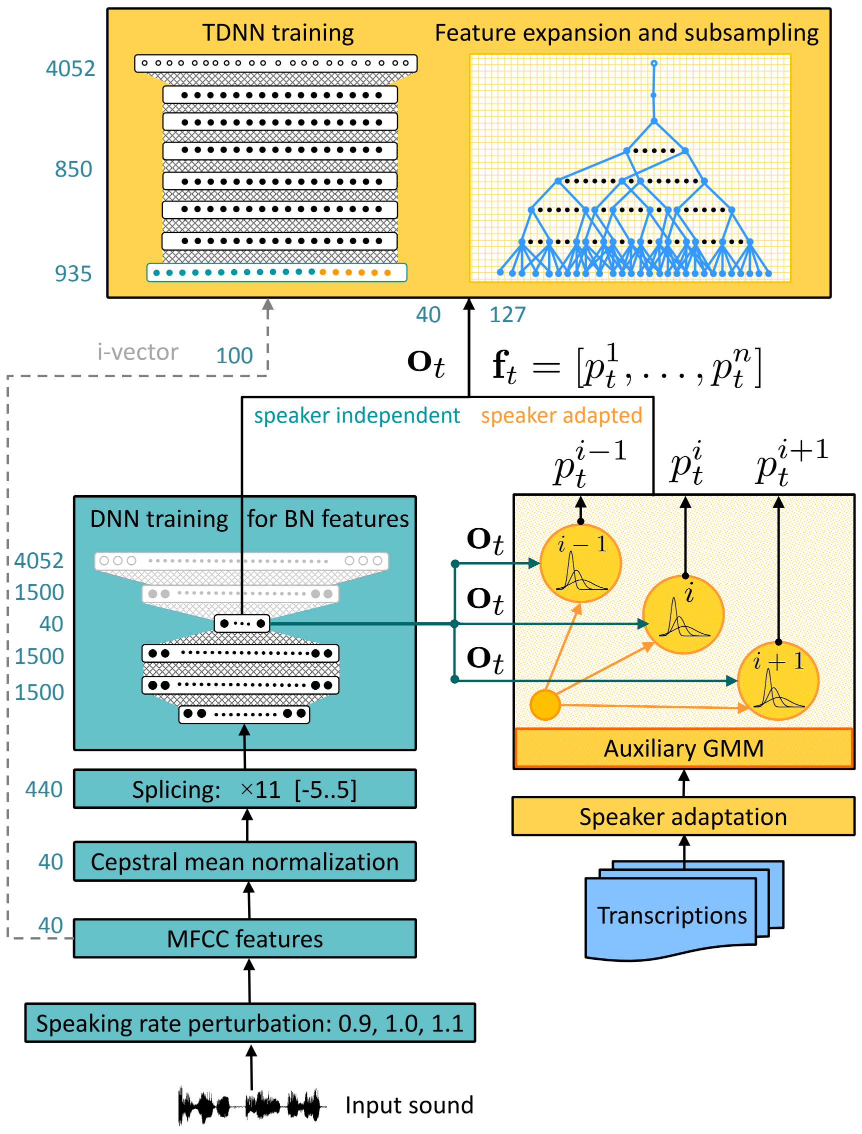


Figure 8.3 Using speaker-adapted BN-based GMMD features for SAT of a TDNN-HMM. Numbers correspond to feature or layer dimensions.

A TDNN model architecture allows to capture the long term dependencies in speech signal. The recently proposed approaches to train TDNN acoustic models [Peddinti et al., 2015] are reported to show higher performance on different LVCSR tasks compared with the standard (best) DNN systems. We aim to incorporate GMMD features into the existing state-of-the art recipe for TDNN model [Peddinti et al., 2015]. For comparison purposes, we take a Kaldi TED-LIUM recipe with a TDNN acoustic model as a basis. An example of using GMMD features for training a TDNN is shown in Figure 8.3. Here, as before, we use

BN features to train the GMM auxiliary model for GMMD feature extraction. Then GMMD features are obtained in the same way as described in Section 8.2.

There are several options to obtain the final features which are fed to the TDNN model. First GMMD features can be combined with the original MFCC features. Another variant consists in combination of GMMD features with BN features, that are used for training the auxiliary GMM model, as shown in Figure 8.3. In both cases we can also use speaker i-vectors as auxiliary features (Section 4.3.6.1). We will experimentally explore all these possibilities in Section 8.6.3.

8.4 Baseline systems and addressed questions

The experiments were conducted on the TED-LIUM corpus (Section 5.2). We used the open-source Kaldi toolkit [Povey et al., 2011b] and mostly followed the standard TED-LIUM Kaldi recipes to train two sets of baseline systems, corresponding to two different types of acoustic models (DNN and TDNN)[2] and two LMs (*LM-cantab* and *LM-lium*, see Section 5.2):

1. *Baseline systems with DNN AMs*. AMs in this set are DNNs trained on BN features, and for the baseline with speaker adaptation we used fMLLR adaptation. *LM-cantab* was used for decoding.

2. *Baseline systems with TDNN AMs*. This set corresponds to a TDNN AMs, with i-vectors and fMLLR for speaker adaptation and *LM-lium* for decoding.

Hence, for each set of baseline systems we trained several acoustic models – SI and SAT AMs. The motivation for creating multiple baseline systems is to have a possibility to compare the proposed adaptation approach not only to some SI baselines, but also to the strong SAT AMs, which use conventional speaker adaptation techniques. Also, one of the question addressed in this study, is to explore several types of basic features to build an auxiliary GMM for GMMD feature extraction. For this reason, we (in addition to the standard original baselines, proposed in Kaldi recipes) built corresponding baseline AMs using the same basic features, as we used for GMMD feature extraction in order to have a more complete picture of experimental results and possibility to compare adaptation techniques under equal conditions.

As we noticed already, besides the standard feed-forward DNNs which were the main focus of all the previous experiments, in this chapter, we also explore TDNN AMs, because recently they have been shown to outperform standard DNNs for many LVCSR tasks [Peddinti et al., 2015].

[2]using *"nnet1"* and *"nnet3"* Kaldi setups: http://kaldi-asr.org/doc/dnn.html

Thus, two series of experiments were conducted:

1. With DNN AMs:
   - Baseline AMs: Section 8.4.2;
   - LM: *LM-cantab*;
   - Proposed systems with GMMD features: Section 8.5.1;
   - Results: Preliminary results to choose basic features and optimal topology of an auxiliary GMM for GMMD feature extraction, as well as the adaptation parameter τ are given Section 8.6.1. Final results are provided in Section 8.6.2.
2. With TDNN AMs:
   - Baseline AMs: Section 8.4.3;
   - LM: *LM-lium*;
   - Proposed systems with GMMD features: Section 8.5.2;
   - Results: Section 8.6.3.

8.4.1 Questions addressed in this study

Several questions addressed in this study include:

1. One of the main questions explored in this chapter is how to effectively integrate the proposed GMMD features into state-of-the art AMs (DNN and TDNN). Some examples of integration were proposed in Sections 8.2 (for DNN) and 8.3 (for TDNN). But we also explore other ways, as will be described below in this chapter.
2. Explore factors which influence GMMD feature extraction:
   - Basic features for training an auxiliary GMM model which is used for GMMD feature extraction:
     - For DNNs we explore 39-dimensional MFCC+$\Delta+\Delta\Delta$ and 40-dimensional BN features (preliminary experiments, Section 8.6.1);
     - For TDNNs we explore 40-dimensional high-resolution MFCC and 40-dimensional BN features (Section 8.6.3);
   - Topology of the auxiliary GMM model (Section 8.6.1)

- Parameter τ in MAP adaptation (see Formula (4.2)), that controls the balance between the maximum likelihood estimate of the mean and its prior value [Gauvain and Lee, 1994] (Section 8.6.3).

3. Study different types of fusion (features-, posterior-, lattice-level) of GMMD features with other features (both SI and adapted) for DNNs (Section 8.6.2) and TDNNs (Section 8.6.3);

4. Compare the proposed adaptation technique with the two most popular adaptation approaches for neural network AMs:
   - fMLLR (for DNNs and TDNNs, Sections 8.6.2 and 8.6.3);
   - i-vectors (for TDNNs, Section 8.6.3).

5. Explore the complementarity of the proposed adaptation technique to other adaptation approaches:
   - fMLLR (for DNNs and TDNNs)
   - i-vectors (for TDNNs)

6. Investigate the impact of the training criteria on the adaptation performance. We compare CE and sMBR criteria for DNNs (Section 8.6.2).

8.4.2 Baseline DNN AMs

We trained four baseline DNN acoustic models:

- **$\mathbf{DNN_{BN}}$-CE** was trained on BN features with CE criterion.
- **$\mathbf{DNN_{BN}}$-sMBR** was obtained from the previous one by performing four epochs of sMBR sequence-discriminative training.
- **$\mathbf{DNN_{BN\text{-}fMLLR}}$-CE** was trained on fMLLR-adapted BN features.
- **$\mathbf{DNN_{BN\text{-}fMLLR}}$-sMBR** was obtained from the previous one by four epochs of sMBR training.

For training DNN models, the initial GMM model was trained first using 39-dimensional MFCC features including delta and acceleration coefficients. Linear discriminant analysis

(LDA), followed by maximum likelihood linear transform (MLLT) and fMLLR transformation, was then applied over these MFCC features to build a GMM-HMM system. Discriminative training with the BMMI objective function (Formula (3.22)) was finally performed on top of this model.

Then a DNN was trained for BN feature extraction. The DNN system was trained using the frame-level cross entropy criterion and the senone alignment generated by the GMM-HMM system. To train this DNN, 40-dimensional log-scale filterbank features concatenated with 3-dimensional pitch-features were spliced across 11 neighboring frames, resulting in 473-dimensional (43×11) feature vectors. After that a DCT transform was applied and the dimension was reduced to 258. A DNN model for extraction 40-dimensional BN features was trained with the following topology: one 258-dimensional input layer; four hidden layers (HL), where the third HL was a BN layer with 40 neurons and other three HLs were 1500-dimensional; the output layer was 2390-dimensional. Based on the obtained BN features we trained the GMM model, which was used to produce the forced alignment, and then SAT-GMM model was trained on fMLLR-adapted BN features. Then fMLLR-adapted BN features were spliced in time with the context of 13 frames: [-10,-5..5,10] to train the final DNN model. The final DNN had a 520-dimensional input layer; six 2048-dimensional HLs with logistic sigmoid activation function, and a 4184-dimensional softmax output layer, with units corresponding to the context-dependent states.

The DNN parameters were initialized with stacked restricted Boltzmann machines (RBMs) by using layer-by-layer generative pre-training. It was trained with an initial learning rate of 0.008 using the cross-entropy objective function to obtain the SAT-DNN-CE model $\mathbf{DNN_{BN\text{-}fMLLR}\text{-}CE}$.

After that four epochs of sequence-discriminative training with per-utterance updates, optimizing state sMBR criteria, were performed to obtain the SAT-DNN-sMBR model $\mathbf{DNN_{BN\text{-}fMLLR}\text{-}sMBR}$.

Baseline SI DNN models ($\mathbf{DNN_{BN}\text{-}CE}$ and $\mathbf{DNN_{BN}\text{-}sMBR}$) were trained in a similar way as the SAT DNNs described above, but without fMLLR adaptation.

8.4.3 Baseline TDNN AMs

We trained five baseline TDNN acoustic models, which differ only in the type of the input features:

- $\mathbf{TDNN_{MFCC}}$ was trained on high-resolution (40-dimensional) MFCC features.

- **TDNN$_{\mathbf{MFCC \oplus i\text{-}vectors}}$** was trained on high-resolution MFCC features, appended with 100-dimensional i-vectors.
- **TDNN$_{\mathbf{BN}}$** was trained on BN features.
- **TDNN$_{\mathbf{BN \oplus i\text{-}vectors}}$** was trained on BN features, appended with 100-dimensional i-vectors.
- **TDNN$_{\mathbf{BN\text{-}fMLLR}}$** was trained on fMLLR-adapted BN features.

The baseline SAT-TDNN model **TDNN$_{\mathbf{MFCC \oplus i\text{-}vectors}}$** is similar to those described in [Peddinti et al., 2015], except for the number of hidden layers and slightly different subsequences of splicing and sub-sampling indexes. The two types of data augmentation strategies were applied for the speech training data: speed perturbation (with factors 0.9, 1.0, 1.1) and volume perturbation. The SAT-TDNN model was trained on high-resolution MFCC features (without dimensionality reduction, keeping all 40 cepstra) concatenated with i-vectors. The 100-dimensional on-line i-vectors were calculated as in [Peddinti et al., 2015], and the statistic for i-vectors was updated every two utterances during the training.

The temporal context was $[t-16, t+12]$ and the splicing indexes used here were[3] $[-2,2]$, $\{-1,2\}$, $\{-3,3\}$, $\{-7,2\}$, $\{0\}$, $\{0\}$. This model has 850-dimensional hidden layers with rectified linear units (ReLU) [Dahl et al., 2013] activation functions, a 4052-dimensional output layer and approximately 10.9 million parameters.

The baseline SI-TDNN model **TDNN$_{\mathbf{MFCC}}$** was trained in a similar way as the SAT-TDNN described above, but without using i-vectors.

In addition to these baseline models, for comparison purpose, we trained two other baseline TDNNs (with and without i-vectors) using BN features instead of high-resolution MFCC features. The same BN features we used later for training an auxiliary monophone GMM model for GMMD feature extraction. These BN features were extracted using a DNN trained in a similar way, as described in Section 8.4.2 for the DNN AM, but on the high-resolution MFCC features (instead of "filter bank $\oplus$ pitch" features) and on the augmented (by means of speed and volume perturbation) data base.

Finally, a TDNN with fMLLR adaptation on BN features was trained (**TDNN$_{\mathbf{BN\text{-}fMLLR}}$**).

[3]For notations see Section 2.4.2 and Figure 2.3

8.5 Proposed systems with GMMD features

For experiments with speaker adaptation we trained two types of acoustic models — DNNs and TDNNs.

8.5.1 DNN AMs on GMMD features

In this set of experiments we trained four DNNs, using the approach proposed in Section 8.2:

- **DNN$_{\textbf{GMMD}\oplus\textbf{BN}}$-CE** is a DNN trained without performing speaker adaptive training, which in our case means that the auxiliary GMM monophone model was not adapted. The model was trained using CE criterion.
- **DNN$_{\textbf{GMMD}\oplus\textbf{BN}}$-sMBR** was obtained from the previous one by performing four epochs of sequence-discriminative training with per-utterance updates, optimizing sMBR criterion.
- **DNN$_{\textbf{GMMD-MAP}\oplus\textbf{BN}}$-CE** was a proposed SAT DNN model trained on speaker adapted GMMD-MAP features, with CE criterion.
- **DNN$_{\textbf{GMMD-MAP}\oplus\textbf{BN}}$-sMBR** was obtained from the previous one by performing four epochs of sMBR sequence training.

Models **DNN$_{\textbf{GMMD-MAP}\oplus\textbf{BN}}$-CE** and **DNN$_{\textbf{GMMD-MAP}\oplus\textbf{BN}}$-sMBR** were trained as described in Section 8.2. The GMMD features were extracted using a monophone auxiliary GMM model, trained on BN features. This GMM model was adapted for each speaker by MAP adaptation algorithm (Section 4.2.1). We took the state tying from the baseline SAT-DNN to train all other models. The purpose of using the same state tying is to allow posterior level fusion for these models.

Both DNN models were trained on the proposed features in the same manner and had the same topology except for the input features, as the final baseline SAT DNN model trained on BN features (Section 8.4.2).

The other two SI models (**DNN$_{\textbf{GMMD}\oplus\textbf{BN}}$-CE** and **DNN$_{\textbf{GMMD}\oplus\textbf{BN}}$-sMBR**) were trained in the same manner, but without speaker adaptation.

8.5.2 TDNN AMs on GMMD features

In this set of experiments we trained five TDNNs, using the approach proposed in Section 8.3.

- **TDNN$_{\mathbf{MFCC \oplus GMMD}}$** was trained on high-resolution MFCC features appended with speaker adapted GMMD features.
- **TDNN$_{\mathbf{MFCC \oplus GMMD \oplus i\text{-}vectors}}$** is a version of the **TDNN$_{\mathbf{MFCC \oplus GMMD}}$** model with 100-dimensional i-vectors appended to input features.
- **TDNN$_{\mathbf{BN \oplus GMMD}}$** was trained on BN features appended with speaker adapted GMMD features.
- **TDNN$_{\mathbf{BN \oplus i\text{-}vectors \oplus GMMD}}$** is a version of the **TDNN$_{\mathbf{BN \oplus GMMD}}$** with 100-dimensional i-vectors appended to input features.
- **TDNN$_{\mathbf{BN\text{-}fMLLR \oplus GMMD}}$** was trained on fMLLR-adapted BN features appended with speaker adapted GMMD features.

All the five TDNN models were trained in the same manner, as the baseline TDNN model (Section 8.4.3), and differ only in the type of the input features.

8.6 Adaptation and fusion results

In this section, the adaptation experiments were conducted in an unsupervised mode on the test data using transcripts from the first decoding pass obtained by the baseline SAT-DNN model, unless explicitly stated otherwise.

8.6.1 Preliminary results: impact of the auxiliary GMM and parameter τ in MAP adaptation on GMMD features

We aim to explore two factors related to the auxiliary GMM, used for GMMD feature extraction: (1) the topology of the model and (2) the type of input features for training this model, and choose the configuration, which is more effective for GMMD feature extraction. We experimented with the following parameters of GMM model: the total number of Gaussians and their distributions between states. Also GMM models were trained on two different types of input features: 39-dimensional MFCC and BN features, extracted as described in Section 8.4.2. In addition, we extracted features with different values of adaptation parameter τ (in Formula (4.2) τ is the parameter that controls the balance between the maximum likelihood estimate of the mean and its prior value in MAP adaptation).

In order to speed up these preliminary experiments, we performed them on a smaller (85 hours) subset of the training dataset. The performance results in terms of WER for DNN models, used for BN feature extraction, are presented in Table 8.1. Parameter *Power* in the table controls the distribution of number of Gaussians between states in the GMM-HMM model according to the distribution of data in the training corpus between these states. It is the exponent for the number of Gaussians according to occurrence counts. When $Power = 0$, all states in the GMM-HMM are modeled with the same number of Gaussians, otherwise the number of Gaussians, used to model a given state, depends on the amount of data in the training corpus belonging to this state.

We can see that for GMMD feature extraction it is better to train an auxiliary GMM model on BN features than on MFCC, and that equal distribution of number of Gaussians between states ($Power = 0$) performs worse than distribution which is dependent on occurrence counts. We set parameter $\tau = 5$ for all the following experiments.

Table 8.1 Impact of the parameters of the auxiliary model topology and τ (adaptation parameter in MAP, see Formula (4.2)) on GMMD feature extraction (on the development set of the TED-LIUM corpus)

Features	Number of Gaussians	τ	Power	WER,%
MFCC	2500	5	0.5	13.89
	2500	5	0.0	14.05
	3800	5	0.5	13.75
	3800	5	0.0	13.65
BN	2500	1	0.5	13.69
	2500	3	0.5	13.51
	2500	5	0.5	13.34
	2500	7	0.5	13.33
	2500	10	0.5	13.40
	2500	5	0.0	13.34
	3800	5	0.5	13.33
	3800	5	0.0	13.48
	10200	5	0.5	13.92

8.6.2 Results for DNN models

We empirically studied different types of fusion described in Section 8.1 and applied them to DNN models trained using GMMD-features extracted as proposed in Section 8.5.1. The performance results in terms of WER for SI and SAT DNN-HMM models are presented in

Table 8.2. The first four lines of the table correspond to the baseline SI (#1, #2) and SAT (#3, #4) DNNs, which were trained as described in Section 8.4.2. Lines #4–#8 correspond to feature level fusion of conventional and GMMD features. For comparison purpose with lattice-level fusion (which we consider further in this chapter) we report WER of the *consensus hypothesis*[4] in parentheses for experiments #4 and #8 – these models will be used in the fusion. Parameter τ in MAP adaptation for both acoustic model training and decoding was set equal to 5.

Table 8.2 Summary of unsupervised adaptation results for DNN models on the TED-LIUM corpus. The results in parentheses correspond to WER of the consensus hypothesis. WERs for baseline systems are given with confidence intervals, corresponding to 5% level of significance.

#	Model	Features	DNN training criterion	WER,%		
				Development	Test$_1$	Test$_2$
1	SI	BN	CE	13.16 ±0.35	11.94 ±0.34	15.43 ±0.31
2			sMBR	12.14 ±0.33	10.77 ±0.32	13.75 ±0.30
3	SAT	BN-fMLLR	CE	11.72	10.88	14.21
4			sMBR	10.64 (10.57)	9.52 (9.46)	12.78 (12.67)
5	SI	GMMD⊕BN	CE	12.92	11.62	15.19
6			sMBR	11.80	10.47	13.52
7	SAT	GMMD-MAP⊕BN	CE	10.46	9.74	13.03
8			sMBR	**10.26** (10.23)	**9.40** (9.31)	**12.52** (12.46)

As we can see from Table 8.2, using speaker-independent (SI) GMMD features in concatenation with SI BN features (#5 and #6) improves the SI baselines (#1 and #2 correspondingly) for all data sets, for both training criteria (CE and sMBR).

By comparing results for AMs trained with CE criterion in lines #5 and #7, we can conclude that the proposed adaptation technique with MAP provides for different datasets approximately 14–18% of relative WER reduction with respect to the SI AM trained on GMMD⊕BN features. For sMBR criterion (lines #6 and #8) the relative WER reduction is approximately 7–13%.

If we compare AMs adapted with fMLLR (#3, #4) with AMs adapted with MAP (#7, #8 correspondingly), we observe that for AMs trained with CE criterion (#3 and #7) the

[4]Consensus hypothesis is obtained by finding the path through the confusion graph with the highest combined link weight [Mangu et al., 2000]

MAP-adapted AM outperforms the fMLLR-adapted AM, and relative WER reduction for different datasets is in the interval 8–11%. However, for AMs trained with sMBR criterion this gain reduces to 1–3% of relative WER reduction.

Table 8.2 shows the effectiveness of feature-level fusion of GMMD and conventional BN features. We are also interested in other types of fusion (posterior- and lattice-level). For these experiments we chose the best AM, trained on MAP-adapted GMMD features (#8) and the AM, trained on fMLLR-adapted BN features (baseline SAT AM: #4). After that, we made posterior fusion for these AMs. The result is given in Table 8.3, line #9. Value α in Formula (8.1) (Section 8.1.2) is a weight of the baseline SAT-DNN model. Parameter α was optimized on the development set.

Table 8.3 Summary of the fusion results for DNN models on the TED-LIUM corpus. The results in parentheses correspond to WER of the consensus hypothesis. Relative WER reduction (Δ_{rel}WER,%) is calculated for consensus hypothesis with respect to AM trained on BN-fMLLR (#4 in Table 8.2). The bold figures in the table indicate the best performance improvement.

#	Fusion: #4 and #8	Development		Test$_1$		Test$_2$	
		WER	Δ_{rel}WER	WER	Δ_{rel}WER	WER	Δ_{rel}WER
9	Posterior fusion, $\alpha = 0.45$	9.98 (9.91)	**6.2**	9.15 (9.06)	**4.3**	12.11 (12.04)	**5.0**
10	Lattice fusion, $\alpha = 0.46$	10.06	4.8	9.09	4.0	12.12	4.4

Finally, we applied lattice fusion (Section 8.1.3) for the same pair of models (line #10). In this type of fusion, before merging lattices, for each edge, scores were replaced by their a posteriori probabilities. Posteriors were computed for each lattice independently. The optimal normalizing factors for each model were found independently on the development set. Then the two lattices were merged into a single lattice and posteriors were weighted using parameter α. As before, value α in Formula (8.1) corresponds to the weight of the baseline SAT-DNN model. The resulting lattice was converted into the confusion network [Mangu et al., 2000] and the final result was obtained from this confusion network.

We can see that both posterior and lattice types of fusion provide similar improvement for all three models: approximately 4%–6% of relative WER reduction in comparison with the adapted baseline model (SAT DNN on fMLLR features, #4), and 12%–18% of relative WER reduction in comparison with the SI baseline model (#2). For models #7–8 only MAP adaptation was applied.

Experiments #9–10 present combination of two different adaptation types: MAP and fMLLR. Is is interesting to note that in all experiments optimal value of α is close to 0.5, so all types of models are equally important for fusion. We can see that MAP adaptation on GMMD features can be complementary to fMLLR adaptation on conventional BN features.

8.6.3 Results for TDNN models

Summary of the adaptation results for the TDNN models is given in Table 8.4. The first five lines of the table correspond to the baseline AMs: SI (#1, #3) and SAT (#2, #4, #5). The rest of the table (lines #6–#10) shows the results for different AMs, trained using GMMD features. Since the performance of MAP adaptation depends on the quality of transcripts, used for adaptation, for these models, for comparison purposes, we present results, obtained with the use of transcripts from three different AMs, used in the first decoding pass (#3, #4, #5). These results have the corresponding letters in numeration (a, b, c).

For SI AM trained on BN features, using GMMD adapted features provides 3.2–10.1% of relative WER reduction (#3 vs #8.a). If we compare results in lines #4 and #9.b, we can see that the use of GMMD features gives an additional improvement over i-vector based adaptation (0.8–6.9% of relative WER reduction for different sets). The best result over all AMs was obtained by $\mathbf{TDNN_{GMMD \oplus BN\text{-}fMLLR}}$ (line #10.c).

To further investigate the complementary of the different adaptation techniques, we performed CNC of recognition results for different TDNN models (Table 8.5). The best results (#24 or #25) are obtained by combinations of TDNN models (#2, #7) or (#2, #10) and provide approximately 8–15% of relative WER reduction in comparison with the conventional state-of-the-art SAT baseline model $\text{TDNN}_{\text{MFCC}\oplus\text{i-vectors}}$, and 5–7% of relative WER reduction in comparison with the strongest baseline AM we could achieve without GMMD features (#5).

Table 8.4 Summary of unsupervised adaptation results for TDNN models on TED-LIUM corpus. The results in parentheses correspond to WER of the consensus hypothesis (they are given only for those results, that will be used further in fusion). Indices, following GMMD features, denotes the AM, used to obtain transcriptions for adaptation. The bold figures in the table indicate the best performance. WERs for baseline systems are given with confidence intervals, corresponding to 5% level of significance.

#	Model	Features	WER,%		
			Development	Test$_1$	Test$_2$
1	SI	MFCC	13.69 ±0.35	11.34 ±0.33	14.38 ±0.30
2	SAT	MFCC⊕i-vectors	11.63 (11.56)	9.62 (9.51)	13.28 (13.19)
3	SI	BN	12.32 ±0.34	10.48 ±0.32	14.00 ±0.30
4	SAT	BN⊕i-vectors	11.62 (11.45)	9.75 (9.69)	13.30 (13.23)
5	SAT	BN-fMLLR	10.70 (10.60)	9.28 (9.25)	12.84 (12.83)
6.a		GMMD$_{\#3}$⊕MFCC	11.30	9.75	13.74
6.b	SAT	GMMD$_{\#4}$⊕MFCC	10.91	9.47	13.45
6.c		GMMD$_{\#5}$⊕MFCC	10.39	9.31	13.27
7.a		GMMD$_{\#3}$⊕MFCC⊕i-vectors	11.23	9.91	13.69
7.b	SAT	GMMD$_{\#4}$⊕MFCC⊕i-vectors	10.90	9.45	13.49
7.c		GMMD$_{\#5}$⊕MFCC⊕i-vectors	10.31 (10.29)	9.32 (9.34)	13.26 (13.20)
8.a		GMMD$_{\#3}$⊕BN	11.07	9.75	13.55
8.b	SAT	GMMD$_{\#4}$⊕BN	10.83	9.36	13.13
8.c		GMMD$_{\#5}$⊕BN	10.29	9.20	13.04
9.a		GMMD$_{\#3}$⊕BN⊕i-vectors	11.01	9.73	13.59
9.b	SAT	GMMD$_{\#4}$⊕BN⊕i-vectors	10.82	9.32	13.20
9.c		GMMD$_{\#5}$⊕BN⊕i-vectors	10.30 (10.22)	9.11 (9.09)	13.09 (13.02)
10.a		GMMD$_{\#3}$⊕BN-fMLLR	10.92	9.54	13.27
10.b	SAT	GMMD$_{\#4}$⊕BN-fMLLR	10.70	9.17	13.01
10.c		GMMD$_{\#5}$⊕BN-fMLLR	**10.15 (10.10)**	**9.06 (9.03)**	**12.84 (12.82)**

Table 8.5 Summary of the fusion results (CNC) for TDNN models on TED-LIUM corpus. Relative WER reduction (Δ_{rel}WER,%) is calculated for consensus hypothesis with respect to the baseline **$\textbf{TDNN}_{\textbf{MFCC}\oplus\textbf{i-vectors}}$**, α is a weight of TDNN$_1$ in the fusion. Numeration of AMs corresponds to numeration in Table 8.4. The bold figures in the table indicate the best performance.

#	TDNN$_1$	TDNN$_2$	α	Development		Test$_1$		Test$_2$	
				WER	Δ_{rel}WER	WER	Δ_{rel}WER	WER	Δ_{rel}WER
11	2	4	0.50	10.83	6.3	9.06	4.7	12.51	5.1
12	4	5	0.29	10.35	10.5	8.92	6.2	12.55	4.8
13	2	5	0.45	10.25	11.3	8.63	9.3	12.34	6.5
14	7.c	9.c	0.50	10.13	12.3	9.05	4.9	12.94	1.9
15	10.c	9.c	0.43	10.01	13.4	8.93	6.1	12.69	3.8
16	10.c	7.c	0.47	9.97	13.7	8.95	5.9	12.69	3.8
17	5	9.c	0.38	9.96	13.8	8.86	6.8	12.40	6.0
18	4	9.c	0.43	9.96	13.8	8.75	8.0	12.35	6.4
19	5	7.c	0.45	9.94	14.0	8.87	6.7	12.40	6.0
20	4	10.c	0.50	9.93	14.1	8.67	8.9	12.28	6.9
21	4	7.c	0.43	9.91	14.3	8.85	7.0	12.39	6.1
22	2	9.c	0.43	9.91	14.3	8.47	11.0	12.22	7.4
23	5	10.c	0.52	9.87	14.6	8.82	7.2	12.35	6.4
24	2	10.c	0.49	9.85	14.8	**8.45**	**11.1**	12.21	7.5
25	2	7.c	0.49	**9.83**	**15.0**	8.66	8.9	**12.20**	**7.5**

8.7 Results for Arabic MGB 2016 Challenge

This section presents the results of using the described approach for speaker adaptation as a part of the LIUM ASR system [Tomashenko et al., 2016f] in 2016 Multi-Genre Broadcast (MGB-2) Challenge in the Arabic language [Ali et al., 2016].

One objective of this section is to study the effectiveness of the proposed adaptation technique when it is directly applied to novel conditions: language, size of the training database, amount of adaptation data for training and test speakers, etc. The language (MSA Arabic, switching to various Arabic dialects), the quality of audio recordings and transcriptions in the training database, as well as other factors make this task more difficult for ASR, and resulted in much higher WER of the baseline system in comparison with the results, reported for WSJ0 and TED-LIUM datasets. The amount of the training data used in the Arabic MGB 2016 dataset is approximately 3.8 times more than in TED-LIUM dataset. Also, the amount of data in development and test datasets is not balanced with respect to speakers. Another aspect addressed in this experiments is to show the performance of AMs trained on GMMD features in combination (particularly, lattice-level fusion) with more different AMs than before (in the previous experiments we reported lattice-level fusion results only for two AMs).

8.7.1 Acoustic models

The Kaldi speech recognition toolkit [Povey et al., 2011b] was used for AM training. The corpus description is given in Section 5.4. The LIUM developed five different AMs, while the same LM set was used for every system. Below we will provide a brief description of these AMs; further details can be found in [Tomashenko et al., 2016f]. All the AMs except the second one (**TDNN$_2$**) were built using the grapheme-based lexicon.

TDNN chain model with 1-to-1 word-to-grapheme lexicon (TDNN$_1$)

The first acoustic model is a recently proposed type of model in the Kaldi toolkit, the so-called *chain* TDNN model [Povey et al., 2016]. Training this model was done by using high-resolution PLP features (without dimensionality reduction, keeping the 40 cepstra) concatenated with 100-dimensional i-vectors for speaker adaptation. We also, as in a standard Kaldi recipe, applied data augmentation techniques before performing the actual training, namely time-warping of the raw audio by factors of 0.9, 1.0 and 1.1, as well as frame-shifting by 0, 1 and 2 frames. On top of this network, we performed a sequence-discriminative training, using the sMBR criterion (Formula (3.3.3.1)).

TDNN chain model with vowels and phonetization (TDNN$_2$)

This acoustic model is a TDNN chain model built in the same way as the one described above, to the exception that no sequence-discriminative training was performed. The model was trained on a realigned vowelized training set, where words were replaced by the two best diacritization candidates provided by MADAMIRA toolkit [Pasha et al., 2014]. Thereby, the phoneme-based lexicon was used for training, while pronunciations were mapped to the grapheme words in the decoding process.

DNN on fMLLR-adapted BN features ($\mathbf{DNN_{BN}}$)

This DNN model was trained on fMLLR-adapted 40-dimensional BN features, spliced in time with the context of 13 frames: [-10,-5...5,10]. It had a 520-dimensional input layer; six 2048-dimensional hidden layers with logistic sigmoid activation function, and a 8467-dimensional softmax output layer, with units corresponding to the context-dependent states. The DNN parameters were initialized with stacked RBMs by using layer-by-layer generative pretraining and trained with CE objective function. After that four epochs of sequence-discriminative training with per-utterance updates, optimizing sMBR criterion, were performed.

DNN on MAP-adapted GMMD features ($\mathbf{DNN_{GMMD}}$)

For training this DNN acoustic model we used speaker-adapted GMMD features. The MAP-adapted GMMD 130-dimensional feature vectors were concatenated with unadapted 40-dimensional BN feature vectors and spliced across 13 frames as before [-10,-5...5,10], resulting in 2210-dimensional feature vectors, for training a new DNN model. The topology of $\mathbf{DNN_{GMMD}}$ is similar to $\mathbf{DNN_{BN}}$, except for the dimension of the input layer. Further $\mathbf{DNN_{GMMD}}$ was trained in the same way as $\mathbf{DNN_{BN}}$ model. In the experimental results, for adaptation on the *Dev* set, we used the best transcripts, obtained from other AMs.

TDNN on MAP-adapted GMMD features ($\mathbf{TDNN_{GMMD}}$)

For training this TDNN acoustic model we used the same features, as for the model $\mathbf{DNN_{GMMD}}$: speaker-adapted GMMD features concatenated with speaker-independent BN features. Except for the difference in the input features and the absence of data augmentation, the $\mathbf{TDNN_{GMMD}}$ model was trained in the same way as $\mathbf{TDNN_1}$. We did not not perform sequence training for this model (particularly due to competition constrains), hence in the sense of training criteria this model is weaker than the $\mathbf{TDNN_1}$ model, and we can not compare them directly. Also no data augmentation was applied for this AM. However, the experiments reported in this section are interesting because they can provide additional understanding of how systems built on GMMD features perform in combination with the other systems built on conventional features.

8.7.2 Results

As earlier in this chapter (Section 8.6.2), we applied the fusion of the recognition results from different AMs, on the word lattice level (Section 8.1.3), in the form of CNC [Evermann and Woodland, 2000].

Table 8.6 summaries the recognition results in terms of WER on the *Dev* dataset. Results are reported after rescoring word-lattices with the 4-gram LM described in Section 5.4. These word-lattices were obtained by using the 2-gram LM to decode the audio signal. Numbers in the columns 3–7 are weights of the recognition results (lattices) from corresponding AMs in CNC. Lines 1–5 represent results for single AMs, both for the 1-best result from lattices, and for the 1-best result from consensus hypotheses. Lines 6–15 show results for pairwise combination, lines 16–25 — fusion results for three models, and lines 26–30 — fusion results for four models. Finally, line 31 demonstrates the WER=19.48%, obtained by fusion of the results from all the five AMs.

If we compare lines #5 and #24, we can see, that by adding in the combination the AMs, trained with the use of GMMD features, we can achieve 7.6% of relative WER reduction in comparison with best single AM ($\mathbf{TDNN_1}$). Even when we combine three AMs (#20, WER=19.86%), we can further slightly improve the result using GMMD features (#31, WER=19.48%) by 2% of relative WER reduction.

We do not report results on the *Test* dataset, because they were computed by the MGB organizers and did not cover all our experiments on the *Dev* dataset. But those results that were available showed that results on *Test* and *Dev* were consistent. For more details see [Tomashenko et al., 2016f].

8.7.3 Discussion

In this section, we described the application of the proposed adaptation approach to the LIUM ASR system, that has been ranked in second position in the 2016 Arabic MGB Challenge. In comparison with the previous experiments, the AMs, described in this section, were trained on a larger speech corpus (648 hours) and were used without any a priori knowledge about test speaker statistics. Also, combining more than two models by CNC, we can observe, that GMMD features continues to provide an improvement in the overall ASR accuracy.

Table 8.6 Recognition results on the *Dev* dataset for different AMs (systems: #1–#5) and their fusion (systems: #6–#31) for MGB-2016. The second column (#) represents the system identification numbers. Numbers in the columns 3—7 are weights of the recognition results (lattices) from corresponding AMs in CN combination. The last column represents 1-best result from consensus hypotheses.

Number of AMs in fusion	#	Fusion weights					WER, %	
		TDNN_1	TDNN_2	DNN_{BN}	DNN_{GMMD}	$\text{TDNN}_{\text{GMMD}}$	Lattice	Cons. hyp.
1	1	-	1	-	-	-	23.66	23.25
	2	-	-	1	-	-	22.67	22.56
	3	-	-	-	-	1	22.31	22.11
	4	-	-	-	1	-	22.05	21.89
	5	1	-	-	-	-	21.69	21.37
2	6	-	-	0.525	0.475	-		21.02
	7	-	0.529	0.471	-	-		20.65
	8	0.620	-	-	0.380	-		20.52
	9	-	-	0.461	-	0.539		20.50
	10	-	0.550	-	0.450	-		20.47
	11	-	-	-	0.468	0.532		20.46
	12	0.598	-	0.402	-	-		20.42
	13	-	0.501	-	-	0.499		20.38
	14	0.562	0.438	-	-	-		20.36
	15	0.567	-	-	-	0.433		20.01
3	16	0.432	-	0.298	0.271	-		20.11
	17	-	-	0.341	0.293	0.365		20.08
	18	-	0.423	0.264	0.313	-		19.99
	19	0.303	0.435	-	0.262	-		19.87
	20	0.369	0.332	0.298	-	-		19.86
	21	-	0.357	0.340	-	0.304		19.86
	22	-	0.383	-	0.249	0.368		19.81
	23	0.475	-	0.187	-	0.338		19.76
	24	0.379	-	-	0.218	0.403		19.75
	25	0.350	0.358	-	-	0.292		19.71
4	26	-	0.314	0.200	0.252	0.235		19.70
	27	0.307	0.326	0.183	0.183	-		19.68
	28	0.252	0.270	0.207	-	0.271		19.64
	29	0.342	-	0.226	0.188	0.244		19.64
	30	0.302	0.259	-	0.193	0.246		19.49
5	31	0.297	0.179	0.130	0.183	0.211		**19.48**

8.8 Conclusions

In this chapter we have investigated the GMM framework for adaptation of DNN-HMM acoustic models and combination of MAP-adapted GMMD with conventional features at different levels of DNN and TDNN architectures [Tomashenko et al., 2016a,b].

Experimental results on the TED-LIUM corpus demonstrated that, in an unsupervised adaptation mode, the proposed adaptation and fusion techniques can provide approximately, a 12–18% relative WER reduction on different adaptation sets, compared to the SI DNN system built on conventional (MFCC) features, and a 4–6% relative WER reduction compared to the strong adapted baseline — SAT-DNN trained on fMLLR adapted features. For TDNN models using the adapted GMMD features and fusion techniques leads to improvement of 15–28% of relative WER reduction in comparison with SI model trained on conventional features and 8–15% of relative WER reduction in comparison with SAT model trained with i-vectors. Hence, for both considered adaptation techniques, fMLLR and i-vectors, the proposed adaptation approach has appeared to be complementary and provide an additional improvement in recognition accuracy.

In addition, we reported the results of applying the proposed adaptation approach to the LIUM ASR system in the 2016 Arabic MGB Challenge, and demonstrated that this complicated ASR system, trained with many hundred hours of speech data and resulting from a combination of several AMs, can also benefit from the use of the adapted GMMD features [Tomashenko et al., 2016f].

Chapter 9

Towards speaker adaptation for end-to-end AMs

In this chapter we explore effectiveness of the proposed speaker adaptation technique for end-to-end AMs.

9.1 Introduction

Recently, various neural end-to-end approaches to ASR have been proposed in the literature [Audhkhasi et al., 2017; Bahdanau et al., 2016; Chan et al., 2016; Collobert et al., 2016; Fritz and Burshtein, 2017; Hannun et al., 2014]. End-to-end acoustic models (AMs) [Chorowski et al., 2014; Graves and Jaitly, 2014; Miao et al., 2015a; Zhang et al., 2017] attempt to map an acoustic signal to a phoneme or grapheme sequence directly by means of neural network models (see Section 2.4). They have been developed as an alternative to the traditional hybrid HMM-DNN approach.

However, the major part of the published works, devoted to end-to-end technology, does not use any speaker adaptation techniques. This lack may be justified by the strong focus of these papers on the neural core of the technology they introduce.

A few papers have offered some preliminary and promising information about the benefits provided by some speaker adaptation techniques to end-to-end AMs. In [Miao et al., 2016], VTLN has been applied to filterbank features, for a neural end-to-end AM training through CTC criterion, providing 3% of relative WER reduction. Speaker i-vectors, appended to the acoustic features, are used in [Audhkhasi et al., 2017] for training phone and word CTC models. Also fMLLR-adapted features are used to train attention-based RNNs [Chorowski et al., 2014]. However in these works [Audhkhasi et al., 2017; Chorowski et al., 2014], no comparison results with the unadapted models are given. Work [Yi et al., 2016] proposes a CTC regularized model adaptation method for the accent adaptation task. Speaker adaptation with speaker codes of RNN-BLSTM AMs is studied in [Huang et al., 2016] for the phone recognition task. In [Huang et al., 2016] AMs were trained with CE criterion, and the adaptation provides about 10% of relative reduction in phone error rate.

The aim of this chapter is to explore the efficiency of speaker adaptation for end-to-end ASR systems with the focus on the proposed approach, on the example of CTC-BLSTM AMs (or shortly, CTC AMs). For this purpose we implemented three different speaker adaptation algorithms to this type of AMs and performed an experimental analysis of these methods. Furthermore, a comparative study of the adaptation techniques was conducted for CTC AMs and TDNN AMs trained with traditional frame-wise CE criterion.

9.2 Speaker adaptation for BLSTM-CTC models

In this chapter we focus on the feature space adaptation techniques for end-to-end acoustic models. In this section we describe the adaptation approach, which is based on using speaker-adapted GMMD features for training BLSTM-CTC models.

The incorporation of the adapted GMMD features into the recipe for training sequence-to-sequence AMs is straightforward. The scheme for SAT of BLST-CTC AMs models with GMM-based adaptation framework is shown in Figure 9.1.

An auxiliary monophone GMM-HMM model is used to transform acoustic feature vectors into log-likelihoods vectors. At this step, speaker adaptation of the auxiliary SI GMM-HMM model is performed for each speaker in the training corpus using correct transcriptions and a new speaker-adapted GMM-HMM model is created in order to obtain speaker-adapted GMMD features.

For a given acoustic feature vector, a new GMMD feature vector is obtained by calculating log-likelihoods across all the states of the auxiliary GMM model on the given vector. Suppose $\mathbf{o}_t$ is the acoustic feature vector at time t, then the new GMMD feature vector $\mathbf{f}_t$ is calculated as in Formulas (6.3) and (7.1).

The adapted GMMD feature vector $\mathbf{f}_t$ is concatenated with the original vector $\mathbf{o}_t$ to obtain vector $\mathbf{x}_t$. These features are used as the input to train a SAT BLSTM-CTC AM.

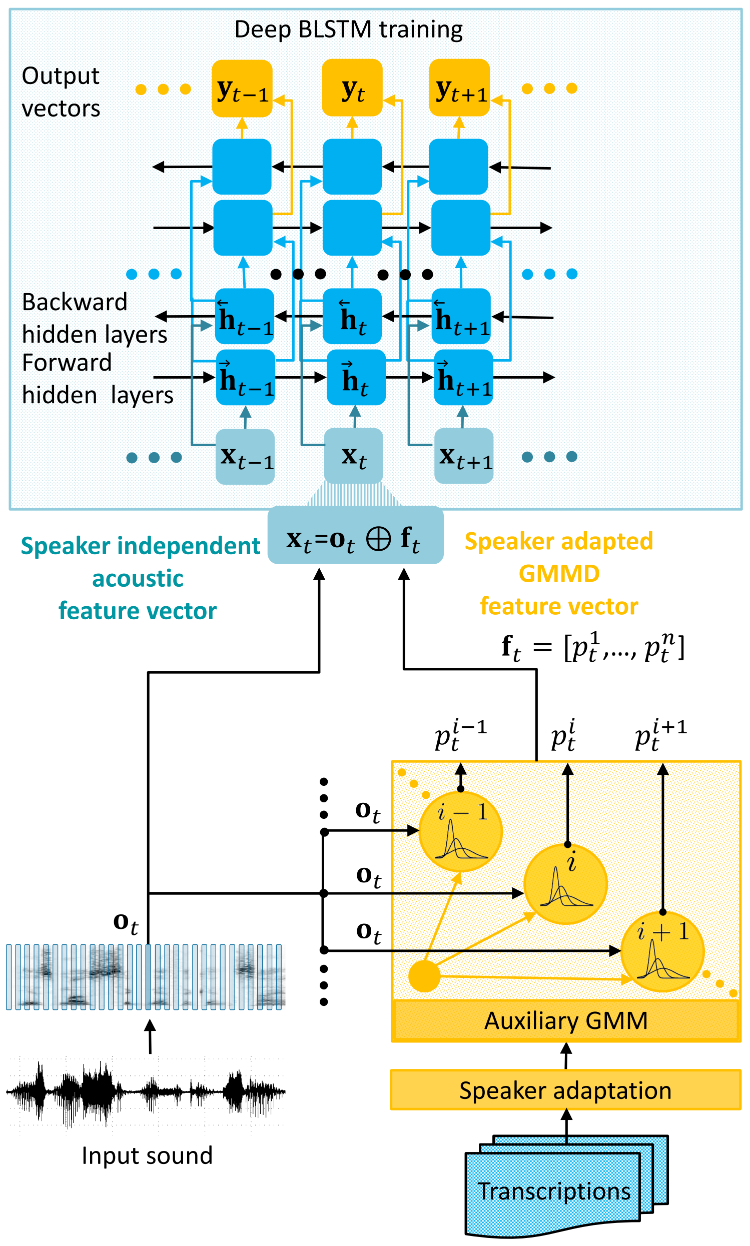

Deep BLSTM training
Output vectors
$\mathbf{y}_{t-1}$
$\mathbf{y}_t$
$\mathbf{y}_{t+1}$
Backward hidden layers
Forward hidden layers
$\overleftarrow{\mathbf{h}}_{t-1}$
$\overleftarrow{\mathbf{h}}_t$
$\overleftarrow{\mathbf{h}}_{t+1}$
$\overrightarrow{\mathbf{h}}_{t-1}$
$\overrightarrow{\mathbf{h}}_t$
$\overrightarrow{\mathbf{h}}_{t+1}$
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
$\mathbf{x}_{t+1}$
$\mathbf{x}_t = \mathbf{o}_t \oplus \mathbf{f}_t$
Speaker independent acoustic feature vector
Speaker adapted GMMD feature vector
$\mathbf{f}_t = [p_t^1, \ldots, p_t^n]$
p_t^{i-1}
p_t^i
p_t^{i+1}
$\mathbf{o}_t$
$i-1$
i
$i+1$
Auxiliary GMM
Speaker adaptation
Transcriptions
Input sound

Figure 9.1 SAT for the BLSTM-CTC AM using GMMD features

9.3 Experiments

The three different types of AM adaptation were explored in this chapter: (1) fMLLR, (2) adaptation using i-vectors, and (3) MAP adaptation using GMMD. The experiments were conducted on the TED-LIUM corpus (Section 5.2). The 4-gram language model *LM-lium* (Section 5.2) was used for evaluation.

9.3.1 Baseline systems

We used the open-source Kaldi toolkit [Povey et al., 2011b] and the Eesen system [Miao et al., 2015a] for the experiments presented in this chapter. Three baseline SI AMs were trained using the Eesen system in a similar manner, and differ only in the front-end processing. The following three type of features were used:

1. *fbanks* $\oplus \Delta \oplus \Delta\Delta$ *(dimension* $= 120$*)*: 40-dimensional filterbank features appended with their first and second-order temporal derivatives;

2. *high-resolution MFCC features (dimension* $= 40$*)*: features extracted without dimensionality reduction, keeping all 40 cepstra;

3. *bottleneck (BN) features (dimension* $= 40$*)*.

The first type of features is the same as proposed in the original Eesen recipe for the TED-LIUM corpus. For the AMs with the two other types of features, also the two types of data augmentation strategies were applied for the speech training data: speed perturbation (with factors 0.9, 1.0, 1.1) and volume perturbation, as in [Peddinti et al., 2015].

The first baseline AM was trained as described in [Miao et al., 2015a] with the CTC criterion and the deep BLSTM architecture. The BLSTM network contains five bidirectional LSTM layers with 320 memory cells in each forward and backward sub-layer. The input features were normalized with per-speaker mean subtraction and variance normalization. The output layer is a 41-dimensional softmax layer with the units corresponding to 39 context-independent phones, 1 noise model and 1 blank symbol.

The third SI AM was trained on BN features [Grézl et al., 2007]. A DNN model for extraction 40-dimensional BN features was trained with the following topology: one 440-dimensional input layer; four hidden layers (HLs), where the third HL was a BN layer with 40 neurons and other three HLs were 1500-dimensional; the output layer was 4052-dimensional. The input features for training this BN extractor were 440-dimensional (40×11): 40-dimensional high-resolution MFCCs spliced across 11 neighboring frames (± 5).

9.3.2 Adapted models

The three types of AM adaptation were empirically explored in this section: fMLLR, adaptation using i-vectors, and MAP adaptation using GMMD features. For all the adapted AMs the same data augmentation strategies were applied during the training, as for the SI ones. All the SAT models were trained with the same neural network topology (except for the input layer) and training criterion, as described in Section 9.3.1 for SI AMs. The six SAT AMs were trained on the following features:

4. *MFCC* $\oplus$ *i-vectors (dimension* $= 140$*)*;
5. *BN* $\oplus$ *i-vectors (dimension* $= 140$*)*;
6. *BN with fMLLR (dimension* $= 40$*)*;
7. *MFCC* $\oplus$ *GMMD (dimension* $= 167$*)*;
8. *BN* $\oplus$ *GMMD (dimension* $= 167$*)*;
9. *BN with fMLLR* $\oplus$ *GMMD (dimension* $= 167$*)*.

For the AMs trained on features #5 and #6, the 100-dimensional on-line i-vectors were calculated as in [Peddinti et al., 2015], and the statistic for i-vectors was updated every two utterances during the training.

For AMs #7–#9 we used BN features to train the auxiliary GMM model for GMMD feature extraction. The speaker-adapted GMMD features were obtained in the same way as described in Section 9.2. Parameter τ in MAP adaptation (see Formula (7.2)) was set equal to 5 for both acoustic model training and decoding.

9.3.3 Adaptation results for CTC AMs

The adaptation experiments were conducted in an unsupervised mode on the test data using transcripts from the first decoding pass.

The performance results in terms of word error rate (WER) for SI and SAT AMs models are presented in Table 9.1. The first three lines of the table (#1–#3) correspond to the baseline SI AMs, which were trained as described in Section 9.3.1, where the very first line represents the Eesen baseline [Miao et al., 2015a]. The next six lines (#4–#9) show the results for the adapted models. The numeration in Table 9.1 coincides with the numeration in Sections 9.3.1 and 9.3.2.

The two last lines of the table (#10 and #11) are obtained with the same AMs, as the lines #8 and #9 correspondingly, but for the extraction of GMMD-adapted features in #10 and #11 we used the transcriptions from the adapted model #6). Notice, that for all other tests (#7–#9) we used transcriptions from the SI model #2.

The best result among all the systems #1–#9 is obtained by system #9, which corresponds to the use of MAP-adapted GMMD features appended with fMLLR-adapted BN features.

Table 9.1 Summary of unsupervised adaptation results for CTC AMs on TED-LIUM corpus. Indices for GMMD features denote the AM, which was used to obtain transcripts for adaptation.

#	Model	Features	WER,%		
			Dev.	Test_1	Test_2
1	SI	fbanks $\oplus\,\Delta\oplus\Delta\Delta$	14.57	11.71	15.29
2	SI	MFCC	13.21	11.16	14.15
3		BN	13.63	11.84	15.06
4	SAT	MFCC $\oplus$ i-vectors	12.92	10.45	14.09
5		BN $\oplus$ i-vectors	13.47	11.37	14.31
6		BN-fMLLR	12.45	10.96	13.79
7		MFCC $\oplus$ $\text{GMMD}_{\#2}$	11.95	10.20	14.04
8		BN $\oplus$ $\text{GMMD}_{\#2}$	11.66	10.14	13.88
9		BN-fMLLR $\oplus$ $\text{GMMD}_{\#2}$	**11.63**	**9.91**	**13.58**
10	SAT	BN $\oplus$ $\text{GMMD}_{\#6}$	11.67	10.11	13.70
11		BN-fMLLR $\oplus$ $\text{GMMD}_{\#6}$	**11.41**	9.93	**13.47**

It can be only slightly improved (#11) for two sets by using the adapted model on the first decoding pass. Among all the adaptation methods, applied separately (#4–#8), the MAP adaptation of GMMD features shows the best performance for both BN and MFCC features.

9.3.4 Comparison of adaptation behavior for BLSTM-CTC and TDNN AMs

In this series of experiments we aim to compare the adaptation behavior of SAT CTC models with the different type of neural network AMs. For this purpose we chose a TDNN model topology, because such models are shown to achieve the best performance result in many state-of-the ASR systems [Peddinti et al., 2015]. These AMs were trained with the CE criterion.

For comparison we used the same set of SI and SAT AMs, as before for CTC-AMs (see Sections 9.3.1 and 9.3.2), except for #1. All SI ans SAT TDNN models were trained in a similar way and have the same model topology. They differ only in the type of the input features. These are the same AMs, as were described in Sections 8.4.3 and 8.5.2. Note that not all the TDNN AMs explored in the mentioned sections are used in the current chapter.

Table 9.2 Summary of unsupervised adaptation results for TDNN AMs on TED-LIUM corpus

#	Model	Features	WER,%		
			Dev.	Test_1	Test_2
2	SI	MFCC	13.69	11.34	14.38
3		BN	12.32	10.48	14.00
4	SAT	MFCC $\oplus$ i-vectors	11.63	9.62	13.28
5		BN $\oplus$ i-vectors	11.62	9.75	13.30
6		BN-fMLLR	**10.70**	**9.28**	**12.84**
7		MFCC $\oplus$ $\text{GMMD}_{\#2}$	11.30	9.75	13.74
8		BN $\oplus$ $\text{GMMD}_{\#2}$	11.07	9.75	13.55
9		BN-fMLLR $\oplus$ $\text{GMMD}_{\#2}$	10.92	9.54	13.27
10	SAT	BN $\oplus$ $\text{GMMD}_{\#6}$	10.29	9.20	13.04
11		BN-fMLLR $\oplus$ $\text{GMMD}_{\#6}$	**10.15**	**9.06**	**12.84**

The results for TDNN AMs are reported in Table 9.2. They include some results from Table 8.4, but we show them here again for clarity and ease of comparison with CTC AMs. Also Figure 9.2 presents the comparison of different adaptation algorithms in terms of relative WER reduction for the speakers from test and development datasets for BLSTM-CTC (Figure 9.2a) and TDNN (Figure 9.2b) AMs. The relative WER reduction is calculated with respect to the SI AMs trained on BN features.

For TDNN AMs we also added in Figure 9.2b the results obtained with the use of SAT AMs for the first decoding pass, because they provide a consistent additional improvement in performance in comparison with the use of SI AMs.

Table 9.3 shows the relative WER reduction for BLSTM-CTC and TDNN AMs in comparison with the best corresponding SI AMs (#2 for CTC and #3 for TDNN). We can see that the optimal choice of features depends on the AM architecture. For SI AMs, BNs have appeared to perform better than MFCCs for TDNN AMs, but for CTC AMs the situation is reversed. Also for SAT CTC and SAT TDNN AMs, the ranking of the systems by the WER is different.

9.4 Conclusions

This chapter explored how the end-to-end AMs can benefit from speaker adaptation and demonstrated that speaker adaptation has remained an essential mechanism for improving

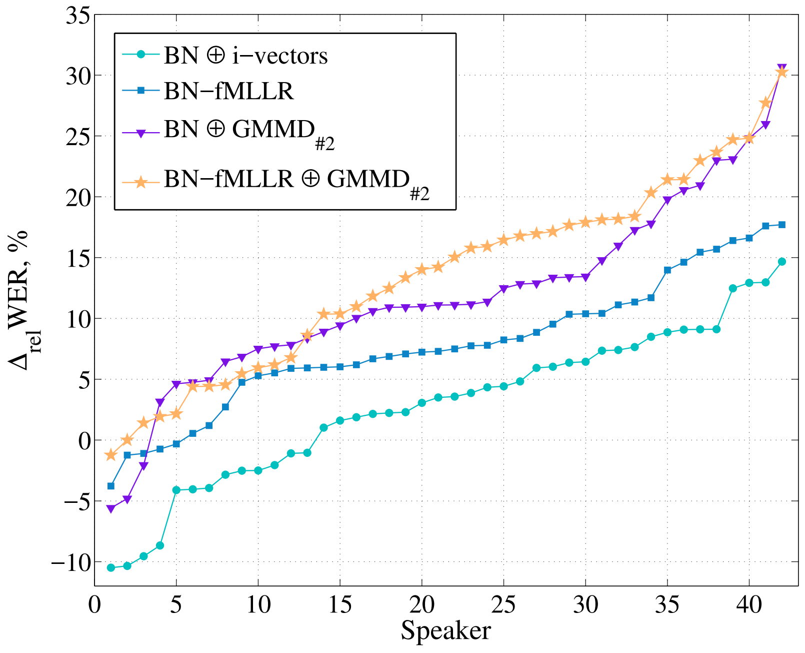
Table 9.3 Relative WER reduction (Δ_{rel}WER) for adapted BLSTM-CTC and TDNN AMs in comparison with the best SI AMs for each AM type (#2 for CTC and #3 for TDNN). Δ_{rel}WER values are calculated based on the results from Tables 9.1 and 9.2.

#	Features	**CTC**: Δ_{rel}WER,%			**TDNN**: Δ_{rel}WER,%		
		Dev.	Test$_1$	Test$_2$	Dev.	Test$_1$	Test$_2$
4	MFCC $\oplus$ i-vectors	2.20	6.36	0.42	5.60	8.21	5.14
5	BN $\oplus$ i-vectors	-1.97	-1.88	-1.13	5.68	6.97	5.00
6	BN-fMLLR	5.75	1.79	2.54	13.15	11.45	8.29
7	MFCC $\oplus$ GMMD$_{\#2}$	9.54	8.60	0.78	8.28	6.97	1.86
8	BN $\oplus$ GMMD$_{\#2}$	11.73	9.14	1.91	10.15	6.97	3.21
9	BN-fMLLR $\oplus$ GMMD$_{\#2}$	11.96	**11.20**	4.03	11.36	8.97	5.21
10	BN $\oplus$ GMMD$_{\#6}$	11.66	9.41	3.18	16.48	12.21	6.86
11	BN-fMLLR $\oplus$ GMMD$_{\#6}$	**13.63**	11.02	**4.81**	**17.61**	**13.55**	**8.29**

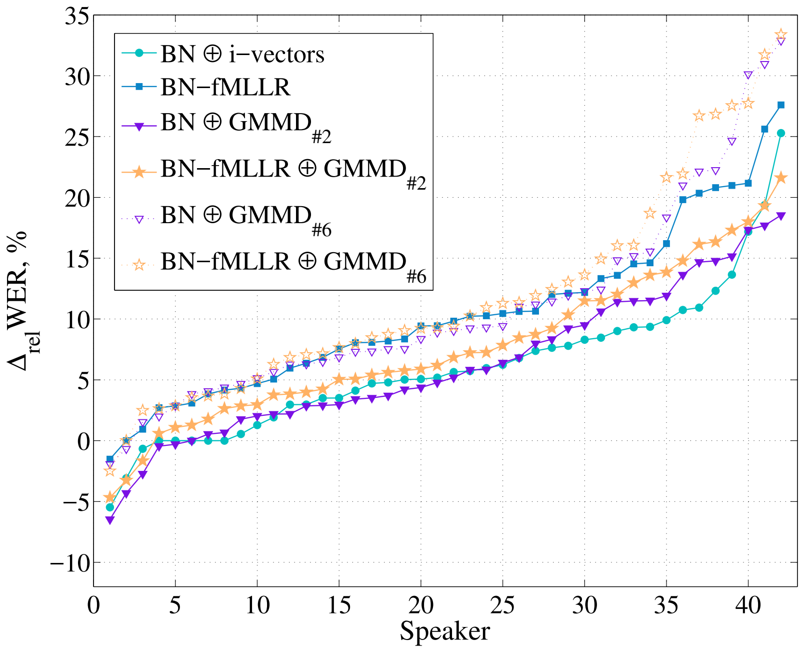
the performance of an ASR system in the new end-to-end speech recognition paradigm. Experimental results on the TED-LIUM corpus showed that in an unsupervised adaptation mode, the adaptation and data augmentation techniques can provide approximately a 10–20% relative WER reduction on different adaptation sets, compared to the SI BLSTM-CTC system built on filter-bank features. In average, the best results for BLSTM-CTC AMs were obtained by using GMMD features and MAP adaptation, which can be further slightly improved by combination with fMLLR adaptation technique.

We found out, that the type of the neural network AM architecture can differently influence the adaptation performance. The comparison with the TDNN-CE AMs showed that for these models, in contradiction to BLSTM-CTC AMs, MAP adaptation using GMMD features outperforms fMLLR only when it uses SAT model in the first decoding pass to obtain transcriptions for adaptation.

Also the obtained results allow us to compare TDNN-CE and BLSTM-CTC AMs in the realistic conditions, when the speaker adaptation is applied, which is important because usually end-to-end and hybrid AMs are compared on incomplete unadapted systems. The best SI TDNN-CE AM outperforms the best SI BLSTM-CTC AM by 1–7% of relative WER reduction for different test sets. For the best SAT AMs this gap in WER for TDNN-CE and BLSTM-CTC AMs increases and reaches 5–13% of relative WER reduction.



(a) CTC



(b) TDNN

Figure 9.2 Per-speaker relative WER reduction (Δ_{rel}WER) for speakers from test and development datasets of the TED-LIUM corpus for different adaptation algorithms with respect to the SI AMs, trained on BN features (#3). Adaptation is performed in an unsupervised mode. Results are ordered in ascending WER reduction order for each AM.

Chapter 10

GMMD feature analysis

The objective of this chapter is to analyze the proposed GMMD features and the adaptation algorithm for better understanding their nature and properties at different levels.

10.1 Phoneme posterior based features

Phoneme posterior based (PPB) features are obtained from time dependent phoneme posterior scores [Gollan and Bacchiani, 2008; Uebel and Woodland, 2001] by computing arc posteriors from the output lattices of the decoder. These features contain more information about the decoding process, than the posterior probabilities from neural networks. We use this type of features to analyze the adaptation performance for TDNN acoustic models.

Let $\{ph_1, \ldots, ph_N\}$ be a set of phonemes and the silence model. For each time frame t we calculate p_t^n — the confidence score of phoneme ph_n $(1 \leq n \leq N)$ at time t in the decoding lattice by calculating arc posterior probabilities. The forward-backward algorithm is used to calculate these arc posterior probabilities from the lattice as follows:[1]

$$P(l|O) = \frac{\sum_{q \in Q_l} P_{ac}(O|q)^{\frac{1}{\lambda}} P_{lm}(w)}{P(O)}, \tag{10.1}$$

where λ is the scale factor (the optimal value of λ is found empirically by minimizing WER of the consensus hypothesis [Mangu et al., 2000]); q is a path through the lattice corresponding to the word sequence w; Q_l is the set of paths passing through arc l; $P_{ac}(O|q)$

[1]We already used the lattice scores obtained from arc posterior probabilities to improve MAP adaptation in Section 7 (Formula (7.2)), but here we again provide the description as a reminder.

is the acoustic likelihood; $P_{lm}(w)$ is the language model probability; and $P(O)$ is the overall likelihood of all paths through the lattice.

For the given frame $\mathbf{o}_t$ at time t we calculate its probability $P(\mathbf{o}_t \in ph_n)$ of belonging to phoneme ph_n, using lattices obtained from the first decoding pass:

$$p_t^n = P(\mathbf{o}_t \in ph_n) = \sum_{l \in S_n(\mathbf{o}_t)} P(l|O), \tag{10.2}$$

where $S_n(\mathbf{o}_t)$ is the set of all arcs corresponding to the phoneme ph_n in the lattice at time t; $P(l|O)$ is the posterior probability of arc l in the lattice.

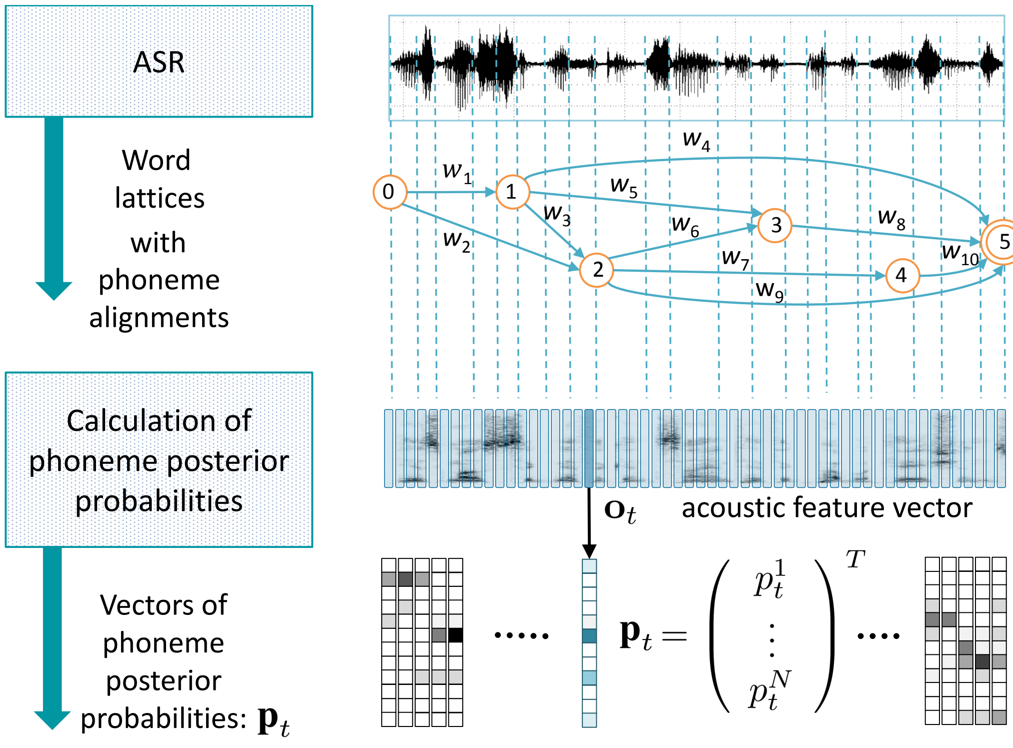

ASR
Word lattices with phoneme alignments
Calculation of phoneme posterior probabilities
Vectors of phoneme posterior probabilities: $\mathbf{p}_t$
w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10}
0 1 2 3 4 5
$\mathbf{o}_t$ acoustic feature vector
$\mathbf{p}_t = \begin{pmatrix} p_t^1 \\ \vdots \\ p_t^N \end{pmatrix}^T$

Figure 10.1 Scheme of phoneme posterior based (PPB) feature extraction

The obtained probability $P(\mathbf{o}_t \in ph_n)$ of frame $\mathbf{o}_t$ belonging to phoneme ph_n is the coordinate value p_t^n on the new feature vector $\mathbf{p}_t$. Thus for a given acoustic feature vector $\mathbf{o}_t$ at time t we obtain a new feature vector $\mathbf{p}_t$:

$$\mathbf{p}_t = \left(p_t^1, \ldots, p_t^N\right), \tag{10.3}$$

where N is the number of phones in the phoneme set used in the ASR system.

Hence for each frame $\mathbf{o}_t$ we have a N-dimensional vector $\mathbf{p}_t$, each coordinate of which represents the probability of this frame to belong to a certain phoneme. When some phonemes are not present in the lattice for a certain frame, we set probabilities equal to some very small value ε for them in the vector (where ε is a minimum value from lattices: $\varepsilon \approx 10^{-9}$). In addition to this, we use state index information (position of the state in phoneme HMM: 0, 1 or 2) from the Viterbi alignment from the original transcripts.

10.2 Visual analysis using t-SNE

The PPB features were visualized using *t-distributed stochastic neighbor embedding* (t-SNE) analysis [Maaten and Hinton, 2008]. This technique allows us to visualize high-dimensional data into two or three dimensional space, in such a way that the vectors, which are close in the original space, are also close in the low dimensional t-SNE representation.

We are interested in how well the different acoustic models can cluster different phoneme states. For better visualization we used data only from inter-word phones and only from the middle state of HMM (State 1). We choose only those phonemes for which we have sufficient amount of data for analysis and perform t-SNE analysis independently on three different groups of phonemes[2]:

- *Vowels (UH, OW, AO, EY, ER, AA, AY, IY, EH, AE, IH, AH);*
- *Consonants-1: Liquids (L, R), Nasals (M, N, NG), Semivowels (W);*
- *Consonants-2: Stops (P, T, D, K), Affricates (CH), Fricatives (F, V, TH, S, Z, SH).*

Each phoneme in these groups represents a separate cluster in our study.

10.3 Davies-Bouldin index

In order to support the visual analysis, we also use the Davies-Bouldin (DB) index [Davies and Bouldin, 1979] to evaluate the quality of the phoneme state clusters obtained from the PPB features:

$$DB = \frac{1}{K}\sum_{k=1}^{K}\max_{j\neq k}\left(\frac{\sigma_k+\sigma_j}{\rho_{k,j}}\right), \tag{10.4}$$

where

[2]The notations are given according to the ARPAbet phoneme set: https://en.wikipedia.org/wiki/Arpabet

K is the number of clusters;

σ_k is the scatter within the cluster k, which is our case the standard deviation of the distance of all vectors corresponding to cluster k, to the cluster center (other possible metric variants are described in [Davies and Bouldin, 1979]);

$\rho_{k,j}$ is a between-cluster separation measure, which in our case is the Euclidean distance between the centroids of clusters k and j.

Smaller values of DB index correspond to better clusters.

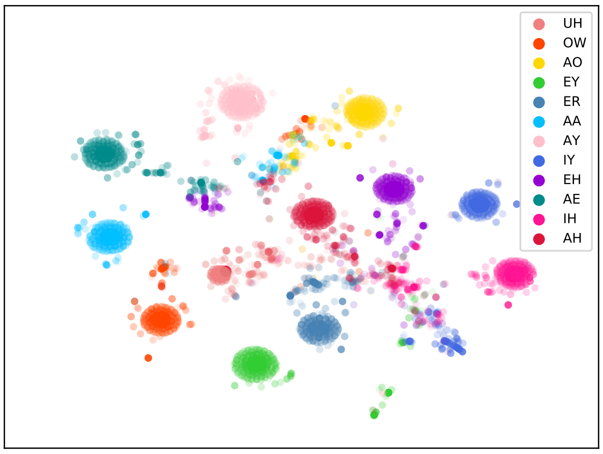
10.4 Analysis for TDNN models

In this set of experiments we compare the following three TDNN acoustic models: $\textbf{TDNN}_{\textbf{MFCC}}$, $\textbf{TDNN}_{\textbf{BN}\oplus\textbf{i-vectors}}$ and $\textbf{TDNN}_{\textbf{BN}\oplus\textbf{i-vectors}\oplus\textbf{GMMD}}$ (from Chapter 8). All the experiments described in this section (except for Figure 10.6) are performed using PPB features (Section 10.1) on the *Development* dataset.

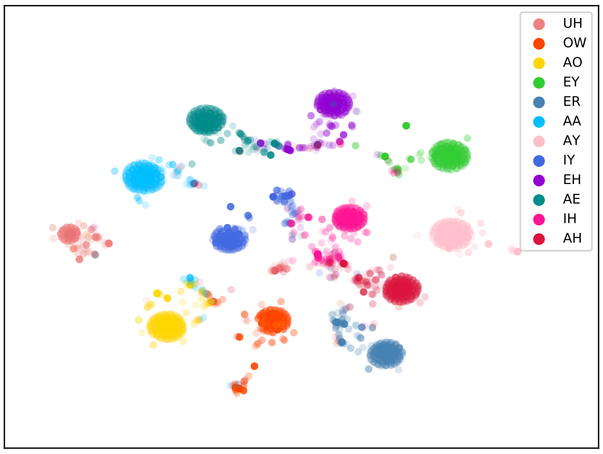
First we analyzed the adaptation algorithm using t-SNE (Section 10.2). The results of the visual t-SNE analysis are given in Figure 10.2 for the group of vowels and in Figures 10.3, 10.4 – for the two group of consonants. We can observe for all groups of phonemes that the adapted features (Figures 10.2b, 10.3b, 10.4b) form more distinct and clear phone clusters than the unadapted features (Figures 10.2a, 10.3a, 10.4a). Also we can note that the use of GMMD features helps to further slightly improve cluster separability (Figures 10.2c, 10.3c, 10.4c).

To support this visual analysis of cluster separation, we calculated DB index (Section 10.3) for all phonemes, separately for each state type, depending on its position in phoneme HMM (State 0, 1, 2). As we can see in Table 10.1, DB index decreases for all HMM states when we move from unadapted (MFCC) to adapted (BN$\oplus$i-vectors) features. That confirms the fact the clusters are better for adapted features. The acoustic model with the adapted GMMD features (BN$\oplus$i-vectors$\oplus$GMMD) shows best result (the smallest value of DB index).

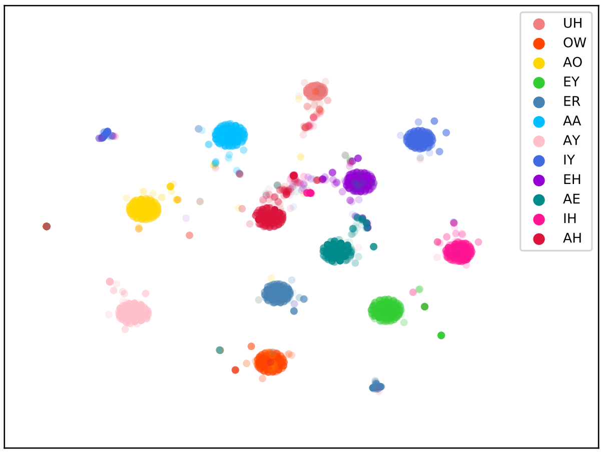
In order to more deeply investigate the adaptation behavior, we calculated additional statistics for PPB features (Table 10.2). Frame error rate (FER) is calculated on the phoneme level using only speech frames (excluding silence). Oracle FER was calculated also only on speech frames as follows: if the correct phoneme is not present in the list of all candidates in the lattices for a given frame, then it was considered as an error.



(a) MFCC

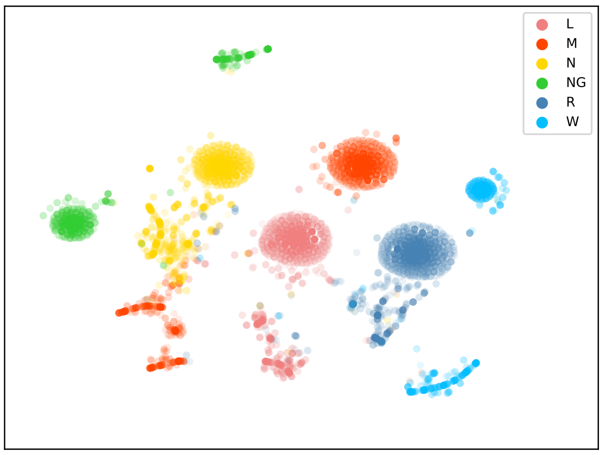


(b) BN $\oplus$ i-vectors

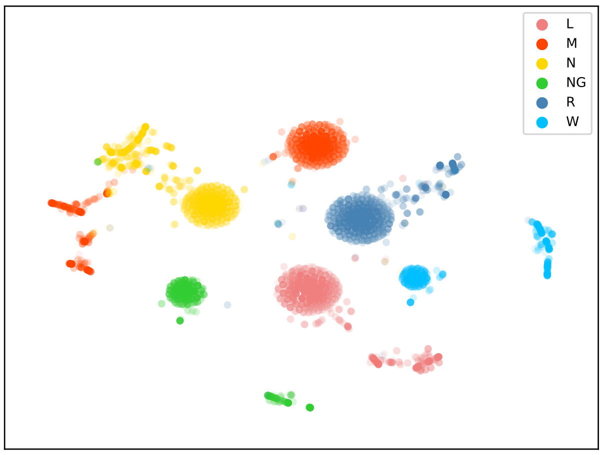


(c) BN $\oplus$ i-vect $\oplus$ GMMD

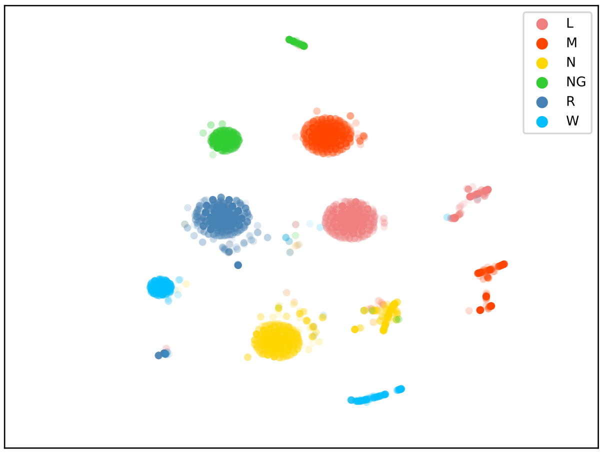
Figure 10.2 Analysis of PPB features for *vowels* using t-SNE for TDNN models, trained on different basic features. Results are shown for the *Development* dataset of the TED-LIUM corpus.



(a) MFCC

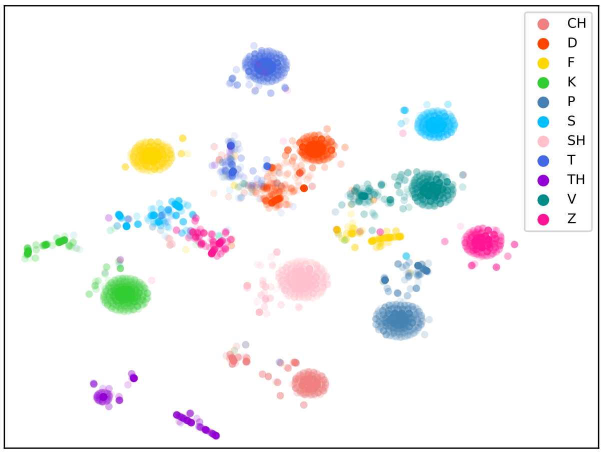


(b) BN ⊕ i-vectors

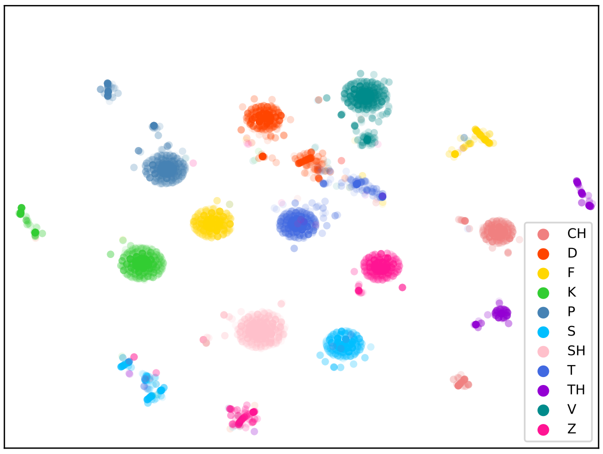


(c) BN ⊕ i-vect ⊕ GMMD

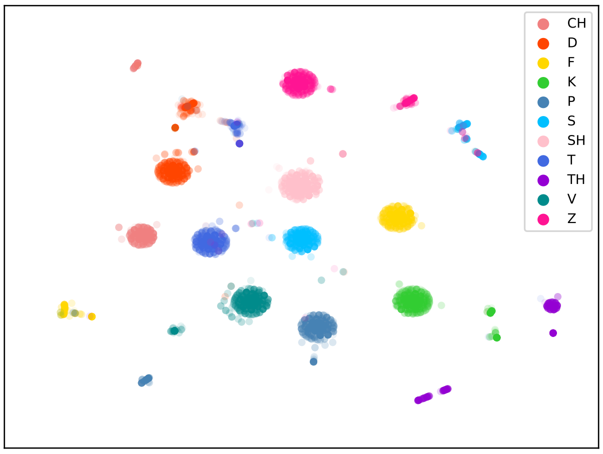
Figure 10.3 Analysis of PPB features for *consonants-1* using t-SNE for TDNN models, trained on different basic features. Results are shown for the *Development* dataset of the TED-LIUM corpus.



(a) MFCC



(b) BN ⊕ i-vectors



(c) BN ⊕ i-vect ⊕ GMMD

Figure 10.4 Analysis of PPB features for *consonants-2* using t-SNE for TDNN models, trained on different basic features. Results are shown for the *Development* dataset of the TED-LIUM corpus.

Table 10.1 Davies-Bouldin (DB) index for different types of features used in TDNN training. The DB index is calculated on PPB features produced by the corresponding model. Results are provided for the development data set of the TED-LIUM corpus.

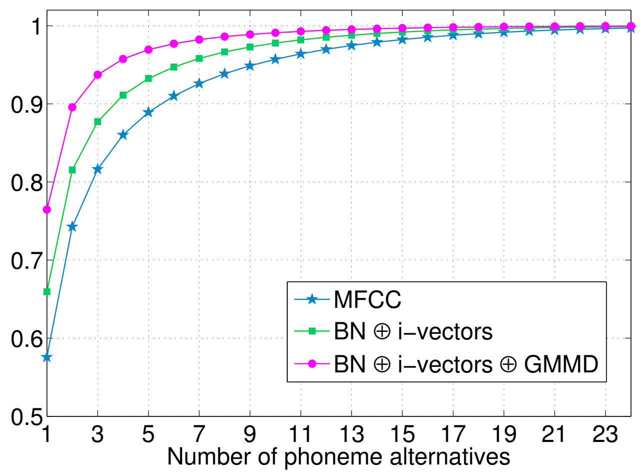
Features	State 0	State 1	State 2
MFCC	1.67	1.52	1.71
BN $\oplus$ i-vectors	1.53	1.36	1.41
BN $\oplus$ i-vectors $\oplus$ GMMD	**1.39**	**1.26**	**1.27**

Table 10.2 Statistics for PPB features, obtained using the corresponding TDNN models. All statistics in the table are calculated only for speech frames (excluding silence). The average log-probability of the correct phoneme is given with the standard deviation. Results are provided for the development data set of the TED-LIUM corpus.

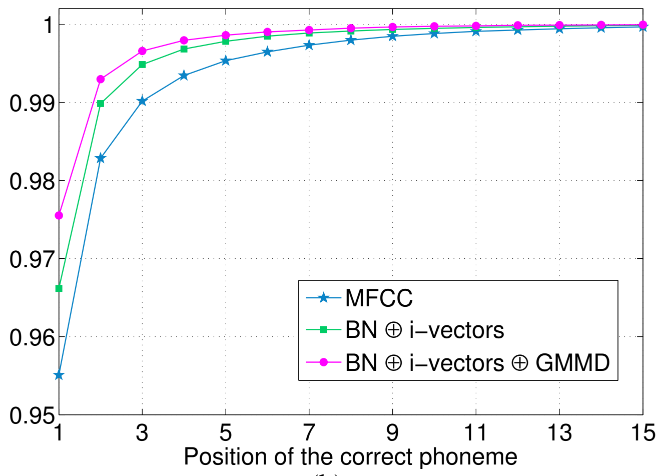
Features	FER	Oracle FER	Aver. correct log-prob.
MFCC	5.18	**0.72**	-0.17 ± 0.83
BN $\oplus$ i-vectors	4.11	0.75	-0.11 ± 0.64
BN $\oplus$ i-vectors $\oplus$ GMMD	**3.64**	1.23	-0.08 ± 0.52

We can see that FER decreases when moving from the unadapted features to the adapted ones, and then to the use of the adapted GMMD features, that correlates with the WER behavior (Table 8.4). It is interesting to note, that Oracle FER, on the contrary, increases with the adaptation. One of the possible explanation for this unusual situation can be phonetic transcriptions errors which occur in the lexicon. The adapted models, which can be more sensitive to the phonetic transcription errors, can more strictly supplant, during the decoding process, hypotheses that do not match the sound.

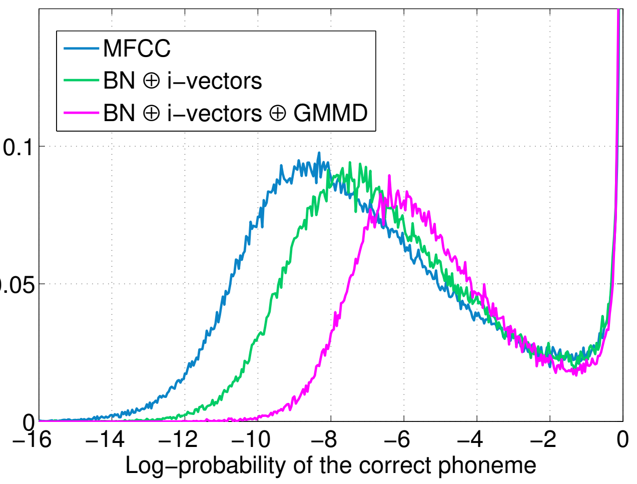
Decoding parameters, such as *decoding beam* and *lattice beam*, were the same for all models, but the adapted models in average have less alternative phoneme candidates for a certain frame, than the unadapted one. This can be seen in Figure 10.5a, which shows the *cumulative distribution functions* (CDFs) of the number of phoneme alternatives presented in the lattices for a certain frame, estimated only for speech frames. Figure 10.5b demonstrates CDFs of position of the correct phoneme (if it exists) in lattices for a certain speech frame in the list of all phoneme candidates ordered by their posterior probabilities. We can conclude from this figure, that for adapted models (especially for the AM with GMMD features), the correct candidate has less incorrect alternatives with higher probabilities than its own.



(a)

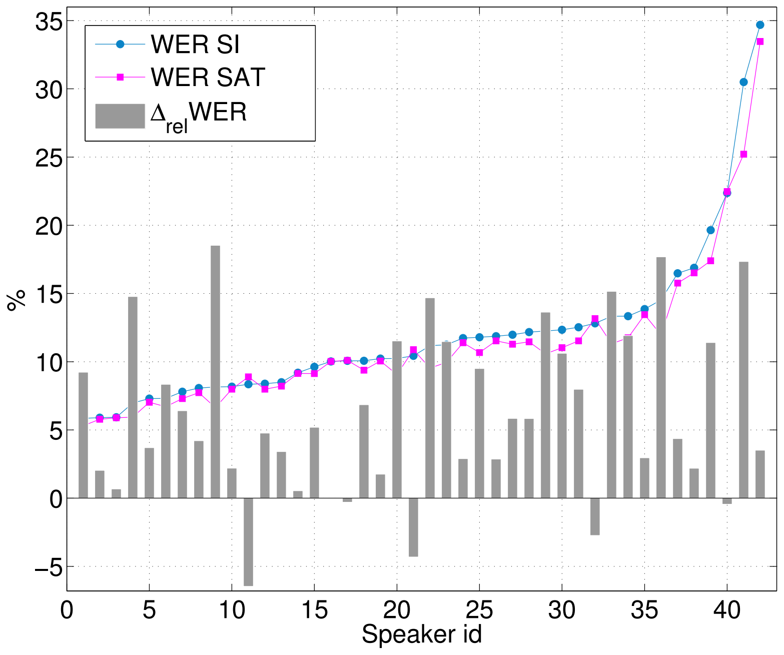


(b)

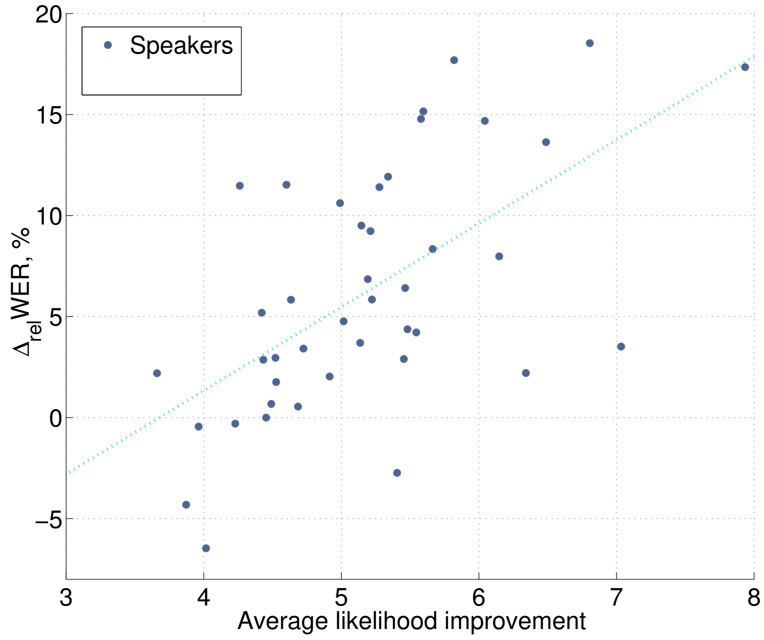


(c)

Figure 10.5 Adaptation analysis based on PPB feature statistics for the three TDNN models on the development data set of the TED-LIUM corpus: (a) CDF of the number of phoneme alternatives in the lattices for a certain frame; (b) CDF of the position of the correct phoneme in the list of all phoneme candidates (ordered by the posterior probability) presented in lattices for a certain frame; (c) Log-probability histogram of the correct phoneme (if it exists) in the lattice for a certain frame.



(a)



(b)

Figure 10.6 Summary statistics for all speakers from the development and the two test datasets of the TED-LIUM corpus: (a) WERs(%) for two (SI and SAT) TDNN models: SI – **TDNN$_{\mathbf{BN}}$**, SAT – **TDNN$_{\mathbf{BN \oplus GMMD}}$**. Relative WER reduction (%) is computed for the given WERs. Results are ordered by increasing WER values for the SI model; (b) Dependence of relative WER reduction (the same as in (a)) on average likelihood improvement, obtained by MAP adaptation of the auxiliary monophone model. The line corresponds to the linear regression model.

Also, the average *correct log-probability* (it is a value from a PPB features vector, which corresponds to the correct phoneme for a given frame) has a maximum value for $\mathbf{TDNN_{BN \oplus i\text{-}vectors \oplus GMMD}}$ model (see the last column of Table 8.4 and a histogram on Figure 10.5c).

Hence, if we compare the statistics presented in Table 10.2 and in Figure 10.5, we can conclude that the adapted models tend to be more "selective" and "discriminative" in comparison with unadapted models in the sense that: (1) they reduce the number of alternatives in the hypothesis more aggressively; (2) they give higher probability values for correct candidates; and (3) the correct phoneme candidate, if it exists in the lattice for a given frame, has in average, less incorrect competitor alternatives with higher probabilities than its own. The AM trained on the adapted GMMD features most strongly shows the same properties.

This analysis demonstrates that the acoustic models trained with the proposed adapted GMMD features perform better than the baseline adapted model not only by comparing the WER (which is the main metric), but also on the other levels.

Also, what is important, this gives us an understanding of the possible way of the adaptation performance improvement through more careful handling of the transcripts, for example, by automatically estimating their quality and reliability.

Finally, Figure 10.6 shows the statistics obtained for 42 speakers from *Development*, $Test_1$ and $Test_2$ data sets for $\mathbf{TDNN_{BN}}$ and $\mathbf{TDNN_{BN \oplus GMMD}}$ models. We can observe that the proposed adaptation approach improves recognition accuracy for 83% of speakers.

10.5 Conclusions

We have looked from the various points of view at the proposed adaptation approach exploring the phoneme posterior based features, generated from the decoding lattices and have demonstrated, that the advantage of using MAP-adapted GMMD features manifests itself at different levels of the decoding process. This analysis also shows a possible potential and direction for improvement of the proposed adaptation approach through more careful handling of quality of the phonetics transcripts used in adaptation. This will be a focus of our future work.

Chapter 11

Conclusions and future work

11.1 Overview of contributions

We have developed and investigated a novel technique for adaptation of DNN AMs.

Chapter 6 introduced a feature-space transformation method for the adaptation of DNN-HMM models [Tomashenko and Khokhlov, 2014b], where input features for DNN training are generated from the likelihoods of the GMM-HMM model. The main advantage of these GMM-derived (GMMD) features is the possibility of performing adaptation of a DNN-HMM model through the adaptation of the auxiliary GMM-HMM. Experiments demonstrate that in supervised adaptation mode MAP adaptation is very effective for this scheme and gives, on average, approximately 35-36% of relative WER reduction for a 5 minute adaptation sample and 5% relative WER reduction for a 5 second adaptation sample, as compared to a SI-DNN model.

Then, Chapter 7 extended the proposed adaptation approach by applying the concept SAT to DNNs built on GMM-derived features. Traditional adaptation algorithms, such as MAP and fMLLR were performed for the auxiliary GMM model used in a SAT procedure for a DNN [Tomashenko and Khokhlov, 2015]. Experimental results on the WSJ0 corpus demonstrate that, in an unsupervised adaptation mode, the proposed adaptation technique can provide, approximately, a 17–20% relative WER reduction for MAP and 18–28% relative WER reduction for fMLLR adaptation on different adaptation sets, compared to the SI DNN-HMM system built on conventional 11×39MFCC features. It has been shown, that fMLLR adaptation for GMMD features is efficient when using a small amount of adaptation data, while MAP adaptation works better when more adaptation data are available. Another contribution of this chapter concerns a method of improving the adaptation algorithm by using confidence scores in MAP adaptation [Tomashenko et al., 2016d]. In addition, data augmentation and data selection strategies, were introduced for improving the regularization in MAP adaptation for DNN. These approaches can be especially useful when the training corpus is small, or when the amount of adaptation data is not known in advance and can vary.

Various ways of integration of adapted GMMD features into different state-of-the art neural network architectures DNN [Tomashenko et al., 2016a,b] and TDNN have been investigated in Chapter 8. Experimental results on the TED-LIUM corpus demonstrate that, in an unsupervised adaptation mode, the proposed adaptation and fusion techniques can provide approximately, a 12–18% relative WER reduction on different adaptation sets, compared to the SI DNN system built on conventional features, and a 4–6% relative WER reduction compared to the strong adapted baseline — SAT-DNN trained on fMLLR adapted features. For TDNN models, using the adapted GMMD features and fusion techniques leads to improvement of 15–28% relative WER reduction in comparison with SI model trained on conventional features and 8–15% relative WER reduction in comparison with SAT model trained with i-vectors. Hence, for both considered adaptation techniques, fMLLR and i-vectors, the proposed adaptation approach has appeared to be complementary and provide an additional improvement in recognition accuracy. In addition, we report the results of applying the proposed adaptation approach to the LIUM ASR system in the 2016 Arabic MGB Challenge, and demonstrate, that the complicated ASR system, trained with many hundred hours of speech data and comprised a combination of several AMs, can also benefit from the use of the adapted GMMD features [Tomashenko et al., 2016f].

Chapter 9 has explored how the end-to-end ASR technology can benefit from speaker adaptation and has demonstrated that speaker adaptation has remained an essential mechanism for improving the performance of an ASR system in the new end-to-end speech recognition paradigm. Experimental results on the TED-LIUM corpus showed that in an unsupervised adaptation mode, the adaptation and data augmentation techniques can provide approximately a 10–20% relative WER reduction on different adaptation sets, compared to the SI BLSTM-CTC system built on filter-bank features. The best results, for BLSTM-CTC AMs, in average, were obtained using GMMD features and MAP adaptation, which can be further slightly improved by combination with fMLLR adaptation technique. We found out, that the type of the neural network AM architecture can differently influence the adaptation performance. The comparison with the TDNN-CE AMs showed that for these models, in contradiction to BLSTM-CTC AMs, MAP adaptation using GMMD features outperforms fMLLR only when it uses SAT model in the first decoding pass to obtain transcriptions for adaptation. Also the obtained results allow us to compare TDNN-CE and BLSTM-CTC AMs in the realistic conditions, when the speaker adaptation is applied, which is important because usually end-to-end and hybrid AMs are compared on incomplete unadapted systems. The best SI TDNN-CE AM outperforms the best SI BLSTM-CTC AM by 1–7% of relative WER reduction for different test sets. For the best SAT AMs this gap in WER for TDNN-CE and BLSTM-CTC AMs increases and reaches 5–13% of relative WER reduction.

Finally, in Chapter 10, we have looked from the various points of view at the proposed adaptation approach exploring the *phoneme posterior based* (PPB) features, generated from the decoding lattices and have demonstrated, that the advantage of using MAP-adapted GMMD features manifests itself at different levels of the decoding process. Visual *t-distributed stochastic neighbor embedding* (t-SNE) analysis applied to different phoneme groups, *Davies-Bouldin* (DB) index and analysis of other statistics from decoding lattices have shown a possible potential and direction for improvement of the proposed adaptation approach through more careful handling the quality of the phonetic transcripts, used in MAP adaptation.

The proposed adaptation approach was experimentally studied on four different speech corpora, corresponding to three languages (English (WSJ0 and TED-LIUM), Russian (STC) and Arabic (MGB-2016). The developed algorithms are integrated into the Kaldi environment. For TED-LIUM and WSJ0 corpora, we plan to prepare and release training recipes based on the proposed approach.

The most important advantage of the proposed approach is that it provides a general framework for adapting DNN acoustic models. In this thesis we used MAP and fMLLR adaptation algorithms as examples to study and demonstrate the effectiveness of the proposed framework, but any other adaptation algorithms can be used instead of them.

11.2 Perspectives

Different possible directions of future developments can be suggested for the work, presented in this thesis.

Application to other adaptation domains

The proposed adaptation framework opens new opportunities not only for the speaker adaptation task, which was the main focus of this thesis, but also to other adaptation domains, such as channel and noise adaptation, or other tasks. For example, the application of the idea, presented in works [Tomashenko and Khokhlov, 2014b, 2015] for speaker adaptation, has later been successfully applied in [Kundu et al., 2016] to noise adaptation of a DNN AM with the use of vector Taylor series (VTS) for an auxiliary GMM model.

Combination with model-based adaptation techniques

In this thesis it was demonstrated, that the proposed adaptation algorithm is complementary to the most commonly used feature-space adaptation techniques for DNNs, such as fMLLR and i-vectors. Also it is interesting to investigate the complementarity of the proposed algorithm to model-based adaptation techniques for DNNs, for example *learning hidden unit contributions* (LHUC), or others.

Speaker adaptive training for GMMD with transcripts from ASR

In the proposed SAT approach, at the training stage, adaptation is performed using original (correct) transcripts from the training corpus. However, at the test time, inaccurate transcripts from the ASR system are used for adaptation. Hence, speaker adaptive training and decoding are performed in different conditions. Probably, using transcriptions from the ASR system for adaptation in the SAT procedure can help to make adaptation more robust to overfitting during the training or to transcription errors during adaptation at the test time. In the proposed SAT scheme we assume that the targets and alignment for training are obtained using the correct transcripts.

Improving the quality of adaptation transcripts

The GMMD feature analysis in Chapter 10 shows a possible direction for improvement of the proposed adaptation approach through more careful handling the quality of the phonetic transcripts used in adaptation. Various approaches can be proposed for this purpose.

We assume that the fixed lexicon for the ASR system can not cover all pronunciations of words that can occur in speech. But we can not extend the lexicon enough with all additional pronunciation variants because too much pronunciations in lexicon degrades the performance of the ASR system. Hence one approach is to add possible pronunciations dynamically during decoding in a two-pass scheme.

GMMD features: high dimension and correlation problem

The dimension n of the GMMD features equals to the number of states in the auxiliary GMM. In our study we usually use a monophone GMM model with 3 states in each monophone.

Depending on the number of phonemes in the language, n can be roughly estimated as $100 < n < 200$. Hence, the dimension of GMMD features is typically larger than the dimension of conventional features (fbanks, MFCC+Δ+$\Delta\Delta$, BN, etc.) which often does not exceed 40. When we aim to train a DNN, a context extension is usually applied which may increase the dimension of the input features up to 30 times. This can lead to too high dimension of the input layer (up to 6000 units).

Using triphones in the auxiliary GMM model instead of monophones can be attractive because adaptation of a triphone GMM can provide more accurate adaptation when we have more adaptation data. But in this case the problem of high dimension becomes critical, because with triphone states the dimension of the input layer can increase in 10–30 times more in comparison with monophones.

Another aspect of this problem is correlation between GMMD feature components, which increases when the number of states in the auxiliary GMM increases.

Proposed solution

To solve this problem, standard feature dimensionality reduction and decorrelation techniques (such as PCA, LDA, HLDA, and others) can be applied.

An alternative promising solution is to use *convolutional neural networks* (CNNs, see Section 2.4.1) for this purpose and train CNNs directly on high-dimensional GMMD features, obtained after splicing with a wide context. Through convolutional and pooling operations, CNNs can solve the problem of a high dimension and correlation in a natural and easy way. Also, to train a CNN more effectively, it is possible to place GMMD features into 2-dimensional plane (to make a “picture" of triphone states) in such a way that acoustically close senones are locally close on this “picture”.

In order to adapt GMMD features extracted using a triphone auxiliary GMM with many states, different approaches, such as *vector field smoothing* (MAP-VFS, Section 4.2.1.2), *structural MAP* (SMAP, Section 4.2.1.3), MLLR with regression classes (Section 4.2.2.1), *maximum a posteriori linear regression* (MAPLR) or other algorithms can be applied.

Publications

International conferences and journals

This thesis mainly contains the work published in the following papers:

1. Tomashenko, N., Khokhlov, Y., and Esteve, Y. (2016). On the use of Gaussian mixture model framework to improve speaker adaptation of deep neural network acoustic models. In INTERSPEECH, pp. 3788-3792. [Tomashenko et al., 2016b]

2. Tomashenko, N., Khokhlov, Y., and Esteve, Y. (2016). A New Perspective on Combining GMM and DNN Frameworks for Speaker Adaptation. In International Conference on Statistical Language and Speech Processing, SLSP-2016, pp. 120-132. Springer International Publishing. [Tomashenko et al., 2016a]

3. Tomashenko, N., Khokhlov, Y., Larcher, A., and Estève, Y. (2016). Exploring GMM-derived features for unsupervised adaptation of deep neural network acoustic models. In International Conference on Speech and Computer, pp. 304-311. Springer International Publishing. [Tomashenko et al., 2016d]

4. Tomashenko N., Vythelingum K., Rousseau A., and Esteve Y. (2016). LIUM ASR systems for the 2016 Multi-Genre Broadcast Arabic Challenge // IEEE Workshop on Spoken Language Technology, SLT-2016, pp. 285–291. [Tomashenko et al., 2016f]

5. Tomashenko, N. A., and Khokhlov, Y. Y. (2015). GMM-derived features for effective unsupervised adaptation of deep neural network acoustic models. In INTERSPEECH, pp. 2882-2886. [Tomashenko and Khokhlov, 2015]

6. Tomashenko, N. A., and Khokhlov, Y. Y. (2014). Speaker adaptation of context dependent deep neural networks based on MAP-adaptation and GMM-derived feature processing. In INTERSPEECH, pp. 2997-3001. [Tomashenko and Khokhlov, 2014b]

Some aspects of the work, which remained beyond the scope of this thesis, but were indirectly connected with some of its points, are presented in the following papers:

7. Khomitsevich O.G., Mendelev V.S., Tomashenko N.A., Rybin S.V., Medennikov I.P., and Kudubayeva S.A. (2015). A Bilingual Kazakh-Russian System for Automatic Speech Recognition and Synthesis // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 9319, pp. 25-33. [Khomitsevich et al., 2015]

8. Bulgakova E., Sholohov A., Tomashenko N., and Matveev Y. (2015). Speaker Verification Using Spectral and Durational Segmental Characteristics // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 9319, pp. 397-404. [Bulgakova et al., 2015a]

9. Tomashenko N., and Khokhlov Y. (2014). Speaking Rate Estimation Based on Deep Neural Networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. [Tomashenko and Khokhlov, 2014c]

10. Chernykh G., Korenevsky M., Levin K., Ponomareva I., and Tomashenko N. (2014). State Level Control for Acoustic Model Training // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8773, No. LNAI, pp. 435–442. [Chernykh et al., 2014b]

11. Levin, K., Ponomareva, I., Bulusheva, A., Chernykh, G., Medennikov, I., Merkin, N., Prudnikov, A., and Tomashenko, N. (2014). Automated closed captioning for Russian live broadcasting. In INTERSPEECH (pp. 1438-1442). [Levin et al., 2014]

12. Tomashenko, N. A., and Khokhlov, Y. Y. (2013). Fast Algorithm for Automatic Alignment of Speech and Imperfect Text Data. In Speech and Computer: 15th International Conference, SPECOM 2013, September 1-5, 2013, Pilsen, Czech Republic, Proceedings. Vol. 8113, pp. 146–153. Springer. [Tomashenko and Khokhlov, 2013]

13. Khokhlov, Y., and Tomashenko, N. (2011). Speech recognition performance evaluation for LVCSR system. In Speech and Computer: 14th International Conference, SPECOM 2011. Kazan pp. 129-135. [Khokhlov and Tomashenko, 2011]

In addition to the above works, during my PhD study, I had an opportunity to contribute towards the two speech recognition evaluation companies: MGB Challenge 2016[1] *and*

[1]The Multi-Genre Broadcast (MGB) Challenge: http://www.mgb-challenge.org/

OpenKWS 2016[2]*, which resulted in the following publications (besides [Tomashenko et al., 2016f]):*

14. Khokhlov Y., Tomashenko N., Medennikov I., and Romanenko A. (2017). Fast and Accurate OOV Decoder on High-Level Features. In INTERSPEECH. pp 2884-2888. [Khokhlov et al., 2017b]

15. Khokhlov Y., Medennikov I., Romanenko A, Mendelev V., Korenevsky M., Prudnikov A., Tomashenko N., and Zatvornitsky A. (2017). The STC Keyword Search System For OpenKWS 2016 Evaluation. In INTERSPEECH. pp 3602-3606. [Khokhlov et al., 2017a]

16. Medennikov, I., Romanenko, A., Prudnikov, A., Mendelev, V., Khokhlov, Y., Korenevsky, M., Tomashenko N., and Zatvornitskiy, A. (2017). Acoustic Modeling in the STC Keyword Search System for OpenKWS 2016 Evaluation. In International Conference on Speech and Computer. pp. 76-86. Springer International Publishing. [Medennikov et al., 2017]

National conferences and journals (France, Russia)

17. Tomashenko, N., Khokhlov, Y., Larcher, A., and Estève, Y. (2016). Exploration de paramètres acoustiques dérivés de GMM pour l'adaptation non supervisée de modèles acoustiques à base de réseaux de neurones profonds. Proceedings of the 31éme Journées d'Études sur la Parole (JEP). (In French). [Tomashenko et al., 2016c]

18. Tomashenko N.A., Khokhlov Yu.Yu., Larcher A., Estève Ya., and Matveev Yu. N. (2016). Gaussian mixture models for adaptation of deep neural network acoustic models in automatic speech recognition systems. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 16, no. 6, pp. 1063–1072. (In Russian). [Tomashenko et al., 2016e]

19. Bulgakova E.V., Sholokhov A.V., and Tomashenko N.A. (2015). Speakers' identification method based on comparison of phoneme lengths statistics. Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 15, no. 1, pp. 70–77. (In Russian). [Bulgakova et al., 2015b]

[2]Open Keyword Search (OpenKWS) Evaluation: https://www.nist.gov/itl/iad/mig/openkws16-evaluation

20. Tomashenko N.A., and Khokhlov Y. Y. (2014). Analysis of data balancing problem in acoustic modeling of automatic speech recognition system. Scientific and Technical Journal "Priborostroenie", 2(57), pp. 17-23. (In Russian). [Tomashenko and Khokhlov, 2014a]

21. Chernykh G. A., Korenevsky M. L., Levin K. E., Ponomareva I. A., and Tomashenko N. A. (2014). Cross-validation state control in acoustic model training of automatic speech recognition system. Scientific and Technical Journal "Priborostroenie", 2(57), pp. 23-28. (In Russian). [Chernykh et al., 2014a]

22. Solomennik, A., Chistikov, P., Rybin, S., Talanov, A. and Tomashenko, N. (2013). Automation of New Voice Creation Procedure For a Russian TTS System. Vestnik MGTU. Priborostroenie, Biometric Technologies, 2, pp. 29-32. (In Russian). [Solomennik et al., 2013]

Bibliography

Abdel-Hamid, O., Deng, L., and Yu, D. (2013). Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Interspeech*, pages 3366–3370.

Abdel-Hamid, O. and Jiang, H. (2013). Fast speaker adaptation of hybrid nn/hmm model for speech recognition based on discriminative learning of speaker code. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7942–7946.

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., and Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545.

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., and Penn, G. (2012). Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4277–4280. IEEE.

Abrash, V., Franco, H., Sankar, A., and Cohen, M. (1995). Connectionist speaker normalization and adaptation. In *in Eurospeech*. Citeseer.

Albesano, D., Gemello, R., Laface, P., Mana, F., and Scanzio, S. (2006). Adaptation of artificial neural networks avoiding catastrophic forgetting. In *Proc. IJCNN'06*, pages 1554–1561. IEEE.

Ali, A., Bell, P., Glass, J., Messaoui, Y., Mubarak, H., Renals, S., and Zhang, Y. (2016). The MGB-2 challenge: Arabic multi-dialect broadcast media recognition. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 279–284. IEEE.

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276.

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., et al. (2015). Deep speech 2: End-to-end speech recognition in English and Mandarin. *arXiv preprint arXiv:1512.02595*.

Anastasakos, T. and Balakrishnan, S. V. (1998). The use of confidence measures in unsupervised adaptation of speech recognizers.

Anastasakos, T., McDonough, J., Schwartz, R., and Makhoul, J. (1996). A compact model for speaker-adaptive training. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 1137–1140. IEEE.

Aubert, X. L. (2002). An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech & Language*, 16(1):89–114.

Audhkhasi, K., Ramabhadran, B., Saon, G., Picheny, M., and Nahamoo, D. (2017). Direct acoustics-to-word models for english conversational speech recognition. *arXiv preprint arXiv:1703.07754*.

Bacchiani, M., Senior, A. W., and Heigold, G. (2014). Asynchronous, online, GMM-free training of a context dependent acoustic model for speech recognition. In *Interspeech*, pages 1900–1904.

Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4945–4949. IEEE.

Bahl, L., Brown, P., De Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, volume 11, pages 49–52. IEEE.

Ban, S. M. and Kim, H. S. (2012). Speaking rate dependent multiple acoustic models using continuous frame rate normalization. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–4. IEEE.

Baum, L. E., Eagon, J. A., et al. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363.

Bell, P. (2010). *Full covariance modelling for speech recognition*. PhD thesis, University of Edinburgh.

Bell, P., Driesen, J., and Renals, S. (2014). Cross-lingual adaptation with multi-task adaptive networks. In *Interspeech*, pages 21–25.

Bell, P., Swietojanski, P., and Renals, S. (2017). Multitask learning of context-dependent targets in deep neural network acoustic models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(2):238–247.

Benesty, J., Sondhi, M. M., and Huang, Y. (2007). *Springer handbook of speech processing*. Springer Science & Business Media.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Bengio, Y., LeCun, Y., Nohl, C., and Burges, C. (1995). LeRec: A NN/HMM hybrid for on-line handwriting recognition. *Neural Computation*, 7(6):1289–1303.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Benzeghiba, M., De Mori, R., Deroo, O., Dupont, S., Erbes, T., Jouvet, D., Fissore, L., Laface, P., Mertins, A., Ris, C., et al. (2007). Automatic speech recognition and speech variability: A review. *Speech communication*, 49(10):763–786.

Bilmes, J. A. and Bartels, C. (2005). Graphical model architectures for speech recognition. *IEEE signal processing magazine*, 22(5):89–100.

Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-García, J., Petrovska-Delacrétaz, D., and Reynolds, D. A. (2004). A tutorial on text-independent speaker verification. *EURASIP journal on applied signal processing*, 2004:430–451.

Bisani, M. and Ney, H. (2004). Bootstrap estimates for confidence intervals in asr performance evaluation. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–409. IEEE.

Bisani, M. and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bottou, L. (1998). Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., et al. (1994). Comparison of classifier methods: a case study in handwritten digit recognition. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 77–82. IEEE.

Bottou, L., Soulie, F. F., Blanchet, P., and Liénard, J.-S. (1990). Speaker-independent isolated digit recognition: Multilayer perceptrons vs. dynamic time warping. *Neural Networks*, 3(4):453–465.

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118.

Bourlard, H., Morgan, N., Wooters, C., and Renals, S. (1992). CDNN: A context dependent neural network for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 2, pages 349–352. IEEE.

Bourlard, H. and Wellekens, C. J. (1990). Links between markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pages 227–236. Springer.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Bulgakova, E., Sholohov, A., Tomashenko, N., and Matveev, Y. (2015a). Speaker verification using spectral and durational segmental characteristics. In *International Conference on Speech and Computer*, pages 397–404. Springer.

Bulgakova, E., Sholokhov, A., and Tomashenko, N. (2015b). Speakers' identification method based on comparison of phoneme lengths statistics. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 15(1):70–77.

Burget, L., Schwarz, P., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Povey, D., et al. (2010). Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4334–4337. IEEE.

Caruana, R. (1998). Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE.

Chelba, C. and Jelinek, F. (2000). Structured language modeling. *Computer Speech & Language*, 14(4):283–332.

Chen, C., Bunescu, R., Xu, L., and Liu, C. (2016). Tone classification in mandarin chinese using convolutional neural networks. *Interspeech 2016*, pages 2150–2154.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics.

Chernykh, G., A., Korenevsky, M., L., Levin, K., E., Ponomareva, I., A., and Tomashenko, N., A. (2014a). Cross-validation state control in acoustic model training of automatic speech recognition system. *Scientific and Technical Journal «Priborostroenie»*, 57(2):23–28.

Chernykh, G., Korenevsky, M., Levin, K., Ponomareva, I., and Tomashenko, N. (2014b). State level control for acoustic model training. In *International Conference on Speech and Computer*, pages 435–442. Springer.

Chesta, C., Siohan, O., and Lee, C.-H. (1999). Maximum a posteriori linear regression for hidden markov model adaptation. In *Eurospeech*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602*.

Chou, W. (1999). Maximum a posterior linear regression with elliptically symmetric matrix variate priors. In *Eurospeech*.

Chou, W., Lee, C.-H., and Juang, B.-H. (1993). Minimum error rate training based on N-best string models. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 652–655. IEEE.

Collobert, R., Puhrsch, C., and Synnaeve, G. (2016). Wav2letter: an end-to-end ConvNet-based speech recognition system. *arXiv preprint arXiv:1609.03193*.

Cui, X., Goel, V., and Kingsbury, B. (2015). Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(9):1469–1477.

Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613.

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2011). Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4688–4691. IEEE.

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Damerau, F. J. (1971). *Markov models and linguistic theory: an experimental study of a model for English*. Number 95. Mouton De Gruyter.

Darken, C. and Moody, J. E. (1991). Note on learning rate schedules for stochastic optimization. In *Advances in neural information processing systems*, pages 832–838.

Dave, N. (2013). Feature extraction methods LPC, PLP and MFCC in speech recognition. *International journal for advance research in engineering and technology*, 1(6):1–4.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227.

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

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Delcroix, M., Kinoshita, K., Hori, T., and Nakatani, T. (2015). Context adaptive deep neural networks for fast acoustic model adaptation. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4535–4539. IEEE.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Deng, L., Abdel-Hamid, O., and Yu, D. (2013a). A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6669–6673. IEEE.

Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., et al. (2013b). Recent advances in deep learning for speech research at Microsoft. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8604–8608. IEEE.

Digalakis, V. V. and Neumeyer, L. G. (1996). Speaker adaptation using combined transformation and Bayesian methods. *IEEE transactions on speech and audio processing*, 4(4):294–300.

Digalakis, V. V., Rtischev, D., and Neumeyer, L. G. (1995). Speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Transactions on speech and Audio Processing*, 3(5):357–366.

Dimitriadis, D., Thomas, S., and Ganapathy, S. (2016). An investigation on the use of i-vectors for robust ASR. In *Interspeech 2016*, pages 3828–3832.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Dupont, S. and Cheboub, L. (2000). Fast speaker adaptation of artificial neural networks for automatic speech recognition. In *Proc. ICASSP*, volume 3, pages 1795–1798. IEEE.

Ellis, D. P. and Reyes-Gomez, M. (2001). Investigations into tandem acoustic modeling for the aurora task. In *Eurospeech 2001: Scandinavia: 7th European Conference on Speech Communication and Technology: September 3-7, 2001, Aalborg Congress and Culture Centre, Aalborg-Denmark: proceedings*, pages 189–192. ISCA-Secretariat.

Ellis, D. P., Singh, R., and Sivadas, S. (2001). Tandem acoustic modeling in large-vocabulary recognition. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Emami, A., Xu, P., and Jelinek, F. (2003). Using a connectionist model in a syntactical based language model. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–I. IEEE.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Evermann, G. and Woodland, P. (2000). Posterior probability decoding, confidence estimation and system combination. In *Proc. Speech Transcription Workshop*, volume 27. Baltimore.

Fan, Y., Qian, Y., Xie, F.-L., and Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Interspeech*, pages 1964–1968.

Fiscus, J. e. a. (2009). Rich transcription 2009 evaluation.

Fiscus, J. G. (1997). A post-processing system to yield reduced word error rates: recognizer output voting error reduction (ROVER). In *Proc. ASRU*, pages 347–354. IEEE.

Fiscus, J. G., Ajot, J., Radde, N., and Laprun, C. (2006). Multiple dimension levenshtein edit distance calculations for evaluating automatic speech recognition systems during simultaneous speech. In *The International Conference on language Resources and Evaluation (LERC)*. Citeseer.

Fiscus, J. G. and et al (1998). The NIST speech recognition scoring toolkit (SCTK). https://www.nist.gov/itl/iad/mig/tools.

Fisher, W. M. and Fiscus, J. G. (1993). Better alignment procedures for speech recognition evaluation. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 59–62. IEEE.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Frankel, J. and King, S. (2007). Speech recognition using linear dynamic models. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):246–256.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Fritz, L. and Burshtein, D. (2017). End-to-end MAP training of a hybrid HMM-DNN model. *arXiv preprint arXiv:1703.10356*.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130.

Gales, M. J. (1998). Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech and language*, 12(2):75–98.

Gales, M. J. (2000). Cluster adaptive training of hidden Markov models. *IEEE transactions on speech and audio processing*, 8(4):417–428.

Gales, M. J. and Woodland, P. C. (1996). Mean and variance adaptation within the MLLR framework. *Computer Speech & Language*, 10(4):249–264.

Garimella, S., Mandal, A., Strom, N., Hoffmeister, B., Matsoukas, S., and Parthasarathi, S. H. K. (2015). Robust i-vector based adaptation of DNN acoustic model for speech recognition. In *Interspeech*, pages 2877–2881.

Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Trans. Speech and Audio Proc.*, 2:291–298.

Gemello, R., Mana, F., Scanzio, S., Laface, P., and De Mori, R. (2006). Adaptation of hybrid ANN/HMM models using linear hidden transformations and conservative training. In *Proc. ICASSP*, pages 1189–1192.

Gers, F. A. and Schmidhuber, J. (2000). Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 189–194. IEEE.

Ghahremani, P., BabaAli, B., Povey, D., Riedhammer, K., Trmal, J., and Khudanpur, S. (2014). A pitch extraction algorithm tuned for automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 2494–2498. IEEE.

Ghahremani, P., Manohar, V., Povey, D., and Khudanpur, S. (2016). Acoustic modelling from the signal domain using CNNs. *Interspeech 2016*, pages 3434–3438.

Gillick, D., Gillick, L., and Wegmann, S. (2011). Don't multiply lightly: Quantifying problems with the acoustic model assumptions in speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 71–76. IEEE.

Gillick, L. and Cox, S. J. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 532–535. IEEE.

Goldwater, S., Jurafsky, D., and Manning, C. D. (2010). Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates. *Speech Communication*, 52(3):181–200.

Gollan, C. and Bacchiani, M. (2008). Confidence scores for acoustic model adaptation. In *Proc. ICASSP*, pages 4289–4292.

Goo, J., Kim, Y., Lim, H., and Kim, H. (2016). Speaker normalization through feature shifting of linearly transformed i-vector. In *Interspeech 2016*, pages 3489–3493.

Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, pages 237–264.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A. C., and Bengio, Y. (2013). Maxout networks. *ICML (3)*, 28:1319–1327.

Graves, A. (2012). Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*.

Grézl, F., Karafiát, M., Kontár, S., and Cernocky, J. (2007). Probabilistic and bottle-neck features for LVCSR of meetings. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–757. IEEE.

Grézl, F., Karafiát, M., and Vesely, K. (2014). Adaptation of multilingual stacked bottle-neck neural network structure for new language. In *Proc. ICASSP*, pages 7654–7658. IEEE.

Gulcehre, C., Cho, K., Pascanu, R., and Bengio, Y. (2014). Learned-norm pooling for deep feedforward and recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 530–546. Springer.

Gupta, V., Kenny, P., Ouellet, P., and Stafylakis, T. (2014). I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription. In *Proc. ICASSP*, pages 6334–6338. IEEE.

Guyon, I., Albrecht, P., Le Cun, Y., Denker, J., and Hubbard, W. (1991). Design of a neural network character recognizer for a touch terminal. *Pattern recognition*, 24(2):105–119.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Hazen, T. J. and Glass, J. R. (1997). A comparison of novel techniques for instantaneous speaker adaptation. In *Eurospeech*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

He, L., Wu, J., Fang, D., and Wu, W. (2000). Speaker adaptation based on combination of MAP estimation and weighted neighbor regression. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II981–II984. IEEE.

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

Hermansky, H., Ellis, D. P., and Sharma, S. (2000). Tandem connectionist feature extraction for conventional hmm systems. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1635–1638. IEEE.

Hermansky, H. and Sharma, S. (1999). Temporal patterns (TRAPS) in ASR of noisy speech. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 1, pages 289–292. IEEE.

Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. *Momentum*, 9(1):926.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, et al. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hirschman, L. and Thompson, H. S. (1997). Overview of evaluation in speech and natural language processing.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hori, T., Hori, C., Minami, Y., and Nakamura, A. (2007). Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition. *IEEE Transactions on audio, speech, and language processing*, 15(4):1352–1365.

Huang, C., Chen, T., Li, S. Z., Chang, E., and Zhou, J.-L. (2001). Analysis of speaker variability. In *Interspeech*, pages 1377–1380.

Huang, Y. and Gong, Y. (2015). Regularized sequence-level deep neural network model adaptation. In *Interspeech*, pages 1081–1085.

Huang, Z., Li, J., Siniscalchi, S. M., Chen, I.-F., Weng, C., and Lee, C.-H. (2014). Feature space maximum a posteriori linear regression for adaptation of deep neural networks. In *Proc. Interspeech*, pages 2992–2996.

Huang, Z., Li, J., Siniscalchi, S. M., Chen, I.-F., Wu, J., and Lee, C.-H. (2015a). Rapid adaptation for deep neural networks through multi-task learning. In *Proc. Interspeech*, pages 2329–9290.

Huang, Z., Siniscalchi, S. M., Chen, I.-F., Li, J., Wu, J., and Lee, C.-H. (2015b). Maximum a posteriori adaptation of network parameters in deep models. In *Proc. Interspeech*, pages 1076–1080.

Huang, Z., Tang, J., Xue, S., and Dai, L. (2016). Speaker adaptation of RNN-BLSTM for speech recognition based on speaker code. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5305–5309. IEEE.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154.

Hwang, M.-Y., Huang, X., and Alleva, F. A. (1996). Predicting unseen triphones with senones. *IEEE Transactions on speech and audio processing*, 4(6):412–419.

Illina, I., Fohr, D., and Jouvet, D. (2011). Grapheme-to-phoneme conversion using conditional random fields. In *12th Annual Conference of the International Speech Communication Association-Interspeech 2011*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Jaitly, N. and Hinton, G. (2011). Learning a better representation of speech soundwaves using restricted Boltzmann machines. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5884–5887. IEEE.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Jia, Y., Huang, C., and Darrell, T. (2012). Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3370–3377. IEEE.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM.

Jiang, H. (2005). Confidence measures for speech recognition: A survey. *Speech communication*, 45(4):455–470.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proc. 8th Annu. Conf. of the Cognitive Science Society, 1986*.

Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495.

Jou, S.-C. S., Schultz, T., and Waibel, A. (2004). Adaptation for soft whisper recognition using a throat microphone. In *Interspeech*.

Jouvet, D., Fohr, D., and Illina, I. (2012). Evaluating grapheme-to-phoneme converters in automatic speech recognition context. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4821–4824. IEEE.

Juang, B.-H. and Katagiri, S. (1992). Discriminative learning for minimum error classification (pattern recognition). *IEEE Transactions on signal processing*, 40(12):3043–3054.

Juang, B.-H., Levinson, S., and Sondhi, M. (1986). Maximum likelihood estimation for multivariate mixture observations of markov chains (corresp.). *IEEE Transactions on Information Theory*, 32(2):307–309.

Kaiser, J., Horvat, B., and Kacic, Z. (2000). A novel loss function for the overall risk criterion based discriminative training of hmm models. In *Sixth International Conference on Spoken Language Processing*.

Kanagawa, H., Tachioka, Y., Watanabe, S., and Ishii, J. (2015). Feature-space structural maplr with regression tree-based multiple transformation matrices for DNN. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 86–92. IEEE.

Kaplan, R. M. and Kay, M. (1994). Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378.

Karanasou, P., Wang, Y., Gales, M. J., and Woodland, P. C. (2014). Adaptation of deep neural network acoustic models using factorised i-vectors. In *Proc. Interspeech*, pages 2180–2184.

Karlsson, I., Bänziger, T., Dankovicova, J., Johnstone, T., Lindberg, J., Melin, H., Nolan, F., and Scherer, K. R. (1998). Within-speaker variability due to speaking manners. In *ICSLP*.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Kenny, P., Boulianne, G., Ouellet, P., and Dumouchel, P. (2007). Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1435–1447.

Khokhlov, Y., Medennikov, I., Romanenko, A., Mendelev, V., Korenevsky, M., Prudnikov, A., Tomashenko, N., and Zatvornitsky, A. (2017a). The STC keyword search system for OpenKWS 2016 evaluation. *Proc. Interspeech 2017*, pages 3602–3606.

Khokhlov, Y. and Tomashenko, N. (2011). Speech recognition performance evaluation for LVCSR system. In *Proc. of the 14th Intern. Conf. on Speech and Computer (SPECOM 2011), Kazan*, pages 129–135.

Khokhlov, Y., Tomashenko, N., Medennikov, I., and Romanenko, A. (2017b). Fast and accurate OOV decoder on high-level features. *Proc. Interspeech 2017*, pages 2884–2888.

Khomitsevich, O., Mendelev, V., Tomashenko, N., Rybin, S., Medennikov, I., and Kudubayeva, S. (2015). A bilingual Kazakh-Russian system for automatic speech recognition and synthesis. In *International Conference on Speech and Computer*, pages 25–33. Springer.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingsbury, B., Sainath, T. N., and Soltau, H. (2012). Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization. In *Interspeech*, pages 10–13.

Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226–239.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Kosaka, T. and Sagayama, S. (1994). Tree-structured speaker clustering for fast speaker adaptation. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume 1, pages I–245. IEEE.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuhn, R., Junqua, J.-C., Nguyen, P., and Niedzielski, N. (2000). Rapid speaker adaptation in eigenvoice space. *IEEE Transactions on Speech and Audio Processing*, 8(6):695–707.

Kumar, N. and Andreou, A. G. (1998). Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition. *Speech communication*, 26(4):283–297.

Kundu, S., Sim, K. C., and Gales, M. J. (2016). Incorporating a generative front-end layer to deep neural network for noise robust automatic speech recognition. In *Interspeech*, pages 2359–2363.

Lang, K. J. and Hinton, G. E. (1988). *A time-delay neural network architecture for speech recognition*. Carnegie Mellon University, Computer Science Department.

Laurent, A., Deléglise, P., Meignier, S., and Spécinov-Trélazé, F. (2009). Grapheme to phoneme conversion using an SMT system.

Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

Le, H.-S., Oparin, I., Allauzen, A., Gauvain, J.-L., and Yvon, F. (2011). Structured output layer neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5524–5527. IEEE.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

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

LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, pages 143–155.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Lee, C.-H. and Huo, Q. (2000). On adaptive decision rules and decision parameter adaptation for automatic speech recognition. *Proceedings of the IEEE*, 88(8):1241–1269.

Lee, C.-H., Lin, C.-H., and Juang, B.-H. (1991). A study on speaker adaptation of the parameters of continuous density hidden markov models. *IEEE Transactions on Signal Processing*, 39(4):806–814.

Lee, L. and Rose, R. C. (1996). Speaker normalization using efficient frequency warping procedures. In *Proc. ICASSP*, volume 1, pages 353–356. IEEE.

Lee, W., Han, K. J., and Lane, I. (2016). Semi-supervised speaker adaptation for in-vehicle speech recognition with deep neural networks. In *Interspeech 2016*, pages 3843–3847.

Leggetter, C. J. and Woodland, P. C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech & Language*, 9(2):171–185.

Lei, X., Lin, H., and Heigold, G. (2013). Deep neural networks with auxiliary Gaussian mixture models for real-time speech recognition. In *Proc. ICASSP*, pages 7634–7638. IEEE.

Levin, K., Ponomareva, I., Bulusheva, A., Chernykh, G., Medennikov, I., Merkin, N., Prudnikov, A., and Tomashenko, N. (2014). Automated closed captioning for Russian live broadcasting. In *Fifteenth Annual Conference of the International Speech Communication Association, Interspeech*.

Li, B. and Sim, K. C. (2010). Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems. In *Proc. Interspeech*, pages 526–529.

Li, J., Deng, L., Haeb-Umbach, R., and Gong, Y. (2015a). *Robust Automatic Speech Recognition: A Bridge to Practical Applications*. Academic Press.

Li, J., Huang, J.-T., and Gong, Y. (2014a). Factorized adaptation for deep neural network. In *Proc. ICASSP*, pages 5537–5541. IEEE.

Li, J., Zheng, R., Xu, B., et al. (2014b). Investigation of cross-lingual bottleneck features in hybrid ASR systems. In *Interspeech*, pages 1395–1399.

Li, L., Wu, Z., Xu, M., Meng, H., and Cai, L. (2016). Combining CNN and BLSTM to extract textual and acoustic features for recognizing stances in mandarin ideological debate competition. *Interspeech 2016*, pages 1392–1396.

Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014c). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670. ACM.

Li, S., Lu, X., Akita, Y., and Kawahara, T. (2015b). Ensemble speaker modeling using speaker adaptive training deep neural network for speaker adaptation. In *Proc. Interspeech*, pages 2892–2896.

Li, X. and Bilmes, J. (2006). Regularized adaptation of discriminative classifiers. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE.

Li, X. and Wu, X. (2015a). I-vector dependent feature space transformations for adaptive speech recognition. In *Interspeech*, pages 3635–3639.

Li, X. and Wu, X. (2015b). Modeling speaker variability using long short-term memory networks for speech recognition. In *Interspeech*, pages 1086–1090.

Liao, H. (2013). Speaker adaptation of context dependent deep neural networks. In *Proc. ICASSP*, pages 7947–7951. IEEE.

Liu, H., Zhu, Z., Li, X., and Satheesh, S. (2017). Gram-CTC: Automatic unit selection and target decomposition for sequence labelling. *arXiv preprint arXiv:1703.00096*.

Liu, S. and Sim, K. C. (2014). On combining DNN and GMM with unsupervised speaker adaptation for robust automatic speech recognition. In *Proc. ICASSP*, pages 195–199. IEEE.

Liu, Y., Shriberg, E., Stolcke, A., Hillard, D., Ostendorf, M., and Harper, M. (2006). Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on audio, speech, and language processing*, 14(5):1526–1540.

Liu, Y., Zhang, P., and Hain, T. (2014). Using neural network front-ends on far field multiple microphones based speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5542–5546.

Lu, L., Ghoshal, A., and Renals, S. (2011). Regularized subspace Gaussian mixture models for cross-lingual speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 365–370. IEEE.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Maier, V. (2002). Evaluating RIL as basis of automatic speech recognition devices and the consequences of using probabilistic string edit distance as input. *Univ. of Sheffield, third year project*.

Manenti, C., Pellegrini, T., and Pinquier, J. (2016). CNN-based phone segmentation experiments in a less-represented language. *Interspeech 2016*, pages 3549–3553.

Mangu, L., Brill, E., and Stolcke, A. (2000). Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

Martin, S., Liermann, J., and Ney, H. (1998). Algorithms for bigram and trigram word clustering. *Speech communication*, 24(1):19–37.

Medennikov, I., Romanenko, A., Prudnikov, A., Mendelev, V., Khokhlov, Y., Korenevsky, M., Tomashenko, N., and Zatvornitskiy, A. (2017). Acoustic modeling in the STC keyword search system for OpenKWS 2016 evaluation. In *International Conference on Speech and Computer*, pages 76–86. Springer.

Metze, F., Sheikh, Z. A., Waibel, A., Gehring, J., Kilgour, K., Nguyen, Q. B., and Nguyen, V. H. (2013). Models of tone for tonal and non-tonal languages. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 261–266. IEEE.

Miao, Y., Gowayyed, M., and Metze, F. (2015a). EESEN: End-to-end speech recognition using deep rnn models and WFST-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 167–174. IEEE.

Miao, Y., Gowayyed, M., Na, X., Ko, T., Metze, F., and Waibel, A. (2016). An empirical exploration of CTC acoustic models. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2623–2627. IEEE.

Miao, Y., Zhang, H., and Metze, F. (2014). Towards speaker adaptive training of deep neural network acoustic models. pages 2189–2193.

Miao, Y., Zhang, H., and Metze, F. (2015b). Speaker adaptive training of deep neural network acoustic models using i-vectors. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(11):1938–1949.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Milner, B. (2002). A comparison of front-end configurations for robust speech recognition. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–797. IEEE.

Mirghafori, N., Fosler, E., and Morgan, N. (1996). Towards robustness to fast speech in ASR. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 335–338. IEEE.

Mitra, V. and Franco, H. (2015). Time-frequency convolutional networks for robust speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 317–323. IEEE.

Mohamed, A.-r., Dahl, G. E., and Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22.

Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Mohri, M., Pereira, F., and Riley, M. (2005). Weighted automata in text and speech processing. *arXiv preprint cs/0503077*.

Moore, B. C. (1997). *An introduction to the psychology of hearing*. Academic Press.

Moreno, P. J. (1996). *Speech recognition in noisy environments*. PhD thesis, Carnegie Mellon University Pittsburgh.

Morgan, N. (2012). Deep and wide: Multiple layers in automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):7–13.

Morgan, N. and Bourlard, H. (1990). Continuous speech recognition using multilayer perceptrons with hidden markov models. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 413–416. IEEE.

Morris, A. (2002). An information theoretic measure of sequence recognition performance. Technical report, IDIAP.

Morris, A. C., Maier, V., and Green, P. D. (2004). From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In *Interspeech*.

Müller, M., Stüker, S., Sheikh, Z., Metze, F., and Waibel, A. (2014). Multilingual deep bottle neck features: a study on language selection and training techniques. In *International Workshop on Spoken Language Translation*, pages 257–264.

Murali Karthick, B., Kolhar, P., and Umesh, S. (2015). Speaker adaptation of convolutional neural network using speaker specific subspace vectors of SGMM. In *Proc. Interspeech*, pages 1096–1100.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nanjo, H., Kato, K., and Kawahara, T. (2001). Speaking rate dependent acoustic modeling for spontaneous lecture speech recognition. In *Interspeech*, pages 2531–2534.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate o(1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376.

Neto, J., Almeida, L., Hochberg, M., Martins, C., Nunes, L., Renals, S., and Robinson, T. (1995). Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system. pages 2171–2174.

Neumeyer, L., Sankar, A., and Digalakis, V. (1995). A comparative study of speaker adaptation techniques. *system*, 1:2.

Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, 8(1):1–38.

Ney, H., Haeb-Umbach, R., Tran, B.-H., and Oerder, M. (1992). Improvements in beam search for 10000-word continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 9–12. IEEE.

Ochiai, T., Matsuda, S., Lu, X., Hori, C., and Katagiri, S. (2014). Speaker adaptive training using deep neural networks. In *Proc. ICASSP*, pages 6349–6353. IEEE.

Ochiai, T., Watanabe, S., Hori, T., and Hershey, J. R. (2017). Multichannel end-to-end speech recognition. *arXiv preprint arXiv:1703.04783*.

Ohkura, K., Sugiyama, M., and Sagayama, S. (1992). Speaker adaptation based on transfer vector field smoothing with continuous mixture density HMMs. In *Second International Conference on Spoken Language Processing*.

Padmanabhan, M., Bahl, L. R., Nahamoo, D., and Picheny, M. A. (1998). Speaker clustering and transformation for speaker adaptation in speech recognition systems. *IEEE Transactions on Speech and Audio Processing*, 6(1):71–77.

Pallet, D. S., Fisher, W. M., and Fiscus, J. G. (1990). Tools for the analysis of benchmark speech recognition tests. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 97–100. IEEE.

Parthasarathi, S. H. K., Hoffmeister, B., Matsoukas, S., Mandal, A., Strom, N., and Garimella, S. (2015). fMLLR based feature-space speaker adaptation of DNN acoustic models. In *Proc. Interspeech*, pages 3630–3634.

Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R. (2014). MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *LREC*, volume 14, pages 1094–1101.

Paul, D. B. (1992). An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 25–28. IEEE.

Paul, D. B. and Baker, J. M. (1992). The design for the wall street journal-based CSR corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics.

Peddinti, V., Povey, D., and Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In *Interspeech*, pages 3214–3218.

Pham, V., Bluche, T., Kermorvant, C., and Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE.

Picheny, M. N. M. (1999). Speed improvement of the time-asynchronous acoustic fast match.

Pinto, J. P. and Hermansky, H. (2008). Combining evidence from a generative and a discriminative model in phoneme recognition. Technical report, IDIAP.

Pironkov, G., Dupont, S., and Dutoit, T. (2016a). I-vector estimation as auxiliary task for multi-task learning based acoustic modeling for automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 1–7. IEEE.

Pironkov, G., Dupont, S., and Dutoit, T. (2016b). Speaker-aware long short-term memory multi-task learning for speech recognition. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 1911–1915. IEEE.

Pitz, M., Wessel, F., and Ney, H. (2000). Improved MLLR speaker adaptation using confidence measures for conversational speech recognition. In *Interspeech*, pages 548–551.

Povey, D. (2004). *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Glembek, O., Goel, N. K., Karafiát, M., Rastrow, A., et al. (2010). Subspace Gaussian mixture models for speech recognition. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4330–4333. IEEE.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., et al. (2011a). The subspace Gaussian mixture model — a structured model for speech recognition. *Computer Speech & Language*, 25(2):404–439.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011b). The Kaldi speech recognition toolkit. In *Proc. ASRU*.

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., and Visweswariah, K. (2008). Boosted MMI for model and feature-space discriminative training. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 4057–4060. IEEE.

Povey, D., Kingsbury, B., Mangu, L., Saon, G., Soltau, H., and Zweig, G. (2005). fMPE: Discriminatively trained features for speech recognition. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 1, pages I–961. IEEE.

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755.

Povey, D. and Woodland, P. (2001). Improved discriminative training techniques for large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 45–48. IEEE.

Povey, D. and Woodland, P. C. (2002). Minimum phone error and I-smoothing for improved discriminative training. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–105. IEEE.

Price, R., i. Iso, K., and Shinoda, K. (2014). Speaker adaptation of deep neural networks using a hierarchy of output layers. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 153–158.

Psutka, J., Müller, L., and Psutka, J. V. (2001). Comparison of mfcc and plp parameterizations in the speaker independent continuous speech recognition task. In *Interspeech*, pages 1813–1816.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Qian, Y., Tan, T., and Yu, D. (2016). Neural network based multi-factor aware joint training for robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2231–2240.

Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rao, K., Peng, F., Sak, H., and Beaufays, F. (2015). Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE.

Rath, S. P., Povey, D., Vesely̌, K., and Cernocky̌, J. (2013). Improved feature processing for deep neural networks. In *Proc. Interspeech*, pages 109–113.

Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41.

Richardson, F., Ostendorf, M., and Rohlicek, J. R. (1995). Lattice-based search strategies for large vocabulary speech recognition. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 576–579. IEEE.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025.

Rosenblatt, F. (1961). Principles of neurodynamics. Perceptrons and the theory of brain mechanisms. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Rousseau, A., Deléglise, P., and Esteve, Y. (2012). TED-LIUM: an automatic speech recognition dedicated corpus. In *Proc. LREC*, pages 125–129.

Rousseau, A., Deléglise, P., and Estève, Y. (2014). Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks. In *Proc. LREC*, pages 3935–3939.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–538.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ. San Diego, Inst. for Cognitive Science.

Rybach, D., Ney, H., and Schlüter, R. (2013). Lexical prefix tree and WFST: A comparison of two dynamic search concepts for LVCSR. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(6):1295–1307.

Rybach, D., Schlüter, R., and Ney, H. (2011). A comparative analysis of dynamic network decoding. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5184–5187. IEEE.

Sainath, T. N., Kingsbury, B., Mohamed, A.-r., Dahl, G. E., Saon, G., Soltau, H., Beran, T., Aravkin, A. Y., and Ramabhadran, B. (2013a). Improvements to deep convolutional neural networks for LVCSR. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 315–320. IEEE.

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. (2013b). Deep convolutional neural networks for LVCSR. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*, pages 8614–8618. IEEE.

Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Sak, H., Senior, A., Rao, K., Irsoy, O., Graves, A., Beaufays, F., and Schalkwyk, J. (2015). Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4280–4284. IEEE.

Samarakoon, L. and Sim, K. C. (2016a). Factorized hidden layer adaptation for deep neural network based acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2241–2250.

Samarakoon, L. and Sim, K. C. (2016b). Multi-attribute factorized hidden layer adaptation for DNN acoustic models. In *Interspeech 2016*, pages 3484–3488.

Samarakoon, L. and Sim, K. C. (2016c). Subspace LHUC for fast adaptation of deep neural network acoustic models. In *Interspeech 2016*, pages 1593–1597.

Saon, G., Povey, D., and Zweig, G. (2005). Anatomy of an extremely fast LVCSR decoder. In *Ninth European Conference on Speech Communication and Technology*.

Saon, G., Soltau, H., Nahamoo, D., and Picheny, M. (2013). Speaker adaptation of neural network acoustic models using i-vectors. In *Proc. ASRU*, pages 55–59. IEEE.

Scherer, K. R. (2003). Vocal communication of emotion: A review of research paradigms. *Speech communication*, 40(1):227–256.

Schluter, R. and Macherey, W. (1998). Comparison of discriminative training criteria. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 1, pages 493–496. IEEE.

Schlüter, R., Macherey, W., Müller, B., and Ney, H. (2001). Comparison of discriminative training criteria and optimization methods for speech recognition. *Speech Communication*, 34(3):287–310.

Schlüter, R., Müller, B., Wessel, F., and Ney, H. (1999). Interdependence of language models and discriminative training. In *Proc. IEEE ASRU Workshop*, pages 119–122.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Schwarz, P. (2009). Phoneme recognition based on long temporal context.

Schwenk, H. (2007). Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Seide, F., Li, G., Chen, X., and Yu, D. (2011a). Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proc. ASRU*, pages 24–29. IEEE.

Seide, F., Li, G., and Yu, D. (2011b). Conversational speech transcription using context-dependent deep neural networks. In *Interspeech*, pages 437–440.

Senior, A., Heigold, G., Bacchiani, M., and Liao, H. (2014). GMM-free DNN acoustic model training. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5602–5606. IEEE.

Senior, A. and Lopez-Moreno, I. (2014). Improving DNN speaker independence with i-vector inputs. In *Proc. ICASSP*, pages 225–229.

Sercu, T., Puhrsch, C., Kingsbury, B., and LeCun, Y. (2016). Very deep multilingual convolutional neural networks for LVCSR. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4955–4959. IEEE.

Serre, T. and Riesenhuber, M. (2004). Realistic modeling of simple and complex cell tuning in the HMAX model, and implications for invariant object recognition in cortex. Technical report, DTIC Document.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., and Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE transactions on pattern analysis and machine intelligence*, 29(3).

Shen, P., Lu, X., and Kawai, H. (2016). Comparison of regularization constraints in deep neural network based speaker adaptation. In *Chinese Spoken Language Processing (ISCSLP), 2016 10th International Symposium on*, pages 1–5. IEEE.

Shinoda, K. (2010). Acoustic model adaptation for speech recognition. *IEICE transactions on information and systems*, 93(9):2348–2362.

Shinoda, K. (2011). Speaker adaptation techniques for automatic speech recognition. *Proc. APSIPA ASC 2011*.

Shinoda, K. and Lee, C.-H. (1997). Structural MAP speaker adaptation using hierarchical priors. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 381–388. IEEE.

Shinoda, K. and Lee, C.-H. (2001). A structural Bayes approach to speaker adaptation. *IEEE Transactions on Speech and Audio Processing*, 9(3):276–287.

Siegler, M. A. (1995). *Measuring and compensating for the effects of speech rate in large vocabulary continuous speech recognition*. PhD thesis, Carnegie Mellon University Pittsburgh.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Siohan, O., Myrvoll, T. A., and Lee, C.-H. (2002). Structural maximum a posteriori linear regression for fast HMM adaptation. *Computer Speech & Language*, 16(1):5–24.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document.

Solomennik, A., Chistikov, P., Rybin, S., Talanov, A., and Tomashenko, N. (2013). Automation of new voice creation procedure for a russian TTS system. *Vestnik MGTU. Priborostroenie,"Biometric Technologies*, 2:29–32.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stadermann, J. and Rigoll, G. (2005). Two-stage speaker adaptation of hybrid tied-posterior acoustic models. In *Proc. ICASSP*, pages 977–980.

Stolcke, A. et al. (2002). SRILM - an extensible language modeling toolkit. In *Interspeech*, volume 2002, page 2002.

Strik, H., Cucchiarini, C., and Kessens, J. M. (2000). Comparing the recognition performance of CSRs: in search of an adequate metric and statistical significance test. In *Interspeech*, pages 740–743. Citeseer.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *Interspeech*, pages 194–197.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Suzuki, M., Tachibana, R., Thomas, S., Ramabhadran, B., and Saon, G. (2016). Domain adaptation of CNN based acoustic models under limited resource settings. *Interspeech 2016*, pages 1588–1592.

Świętojański, P. (2016). *Learning Representations for Speech Recognition using Artificial Neural Networks*. PhD thesis, University of Edinburgh.

Swietojanski, P., Bell, P., and Renals, S. (2015). Structured output layer with auxiliary targets for context-dependent acoustic modelling. In *Proc. Interspeech*, pages 3605–3609.

Swietojanski, P., Ghoshal, A., and Renals, S. (2013). Revisiting hybrid and GMM-HMM system combination techniques. In *Proc. ICASSP*, pages 6744–6748. IEEE.

Swietojanski, P., Ghoshal, A., and Renals, S. (2014). Convolutional neural networks for distant speech recognition. *IEEE Signal Processing Letters*, 21(9):1120–1124.

Swietojanski, P., Li, J., and Renals, S. (2016). Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(8):1450–1463.

Swietojanski, P. and Renals, S. (2014). Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *Proc. SLT*, pages 171–176. IEEE.

Swietojanski, P. and Renals, S. (2016). Differentiable pooling for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(10):1773–1784.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Takahashi, J.-i. and Sagayama, S. (1997). Vector-field-smoothed bayesian learning for fast and incremental speaker/telephone-channel adaptation. *Computer Speech & Language*, 11(2):127–146.

Tan, T., Qian, Y., Yin, M., Zhuang, Y., and Yu, K. (2015). Cluster adaptive training for deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4325–4329. IEEE.

Tang, Z., Li, L., Wang, D., and Vipperla, R. (2017). Collaborative joint training with multitask recurrent model for speech and speaker recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(3):493–504.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

Tomashenko, Natalia, A. and Khokhlov, Yuri, Y. (2014a). Analysis of data balancing problem in acoustic modeling of automatic speech recognition system. *Scientific and Technical Journal «Priborostroenie»*, 57(2):17–22.

Tomashenko, N. and Khokhlov, Y. (2014b). Speaker adaptation of context dependent deep neural networks based on MAP-adaptation and GMM-derived feature processing. In *Proc. Interspeech*, pages 2997–3001.

Tomashenko, N. and Khokhlov, Y. (2014c). Speaking rate estimation based on deep neural networks. In *International Conference on Speech and Computer*, pages 418–424. Springer.

Tomashenko, N. and Khokhlov, Y. (2015). GMM-derived features for effective unsupervised adaptation of deep neural network acoustic models. In *Proc. Interspeech*, pages 2882–2886.

Tomashenko, N., Khokhlov, Y., and Estève, Y. (2016a). A new perspective on combining GMM and DNN frameworks for speaker adaptation. In *Statistical Language and Speech Processing: 4th International Conference, SLSP 2016, Pilsen, Czech Republic, October 11-12, 2016, Proceedings*, pages 120–132. Springer International Publishing.

Tomashenko, N., Khokhlov, Y., and Esteve, Y. (2016b). On the use of Gaussian mixture model framework to improve speaker adaptation of deep neural network acoustic models. In *Proc. Interspeech*, pages 3788–3792.

Tomashenko, N., Khokhlov, Y., Larcher, A., and Estève, Y. (2016c). Exploration de paramètres acoustiques dérivés de GMM pour l'adaptation non supervisée de modèles acoustiques à base de réseaux de neurones profonds. In *Proc. 31éme Journées d'Études sur la Parole (JEP)*, pages 337–345.

Tomashenko, N., Khokhlov, Y., Larcher, A., and Esteve, Y. (2016d). Exploring GMM-derived features for unsupervised adaptation of deep neural network acoustic models. In *Proc. International Conference on Speech and Computer*, pages 304–311. Springer.

Tomashenko, N., Khokhlov, Y., Larcher, A., Estève, Y., and Matveev, Y. N. (2016e). Gaussian mixture models for adaptation of deep neural network acoustic models in automatic speech recognition systems. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 16(6):1063–1072.

Tomashenko, N., Vythelingum, K., Rousseau, A., and Estève, Y. (2016f). LIUM ASR systems for the 2016 multi-genre broadcast Arabic challenge. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 285–291. IEEE.

Tomashenko, N. A. and Khokhlov, Y. Y. (2013). Fast algorithm for automatic alignment of speech and imperfect text data. In *International Conference on Speech and Computer*, pages 146–153. Springer.

Tonomura, M., Kosaka, T., and Matsunaga, S. (1995). Speaker adaptation based on transfer vector field smoothing using maximum a posteriori probability estimation. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 688–691. IEEE.

Tóth, L. and Gosztolya, G. (2016). *Adaptation of DNN Acoustic Models Using KL-divergence Regularization and Multi-task Training*, pages 108–115. Springer International Publishing.

Tran, D. T., Delroix, M., Ogawa, A., and Nakatani, T. (2016). Factorized linear input network for acoustic model adaptation in noisy conditions. *Interspeech 2016*, pages 3813–3817.

Trentin, E. (2001). Networks with trainable amplitude of activation functions. *Neural Networks*, 14(4):471–493.

Trentin, E. and Gori, M. (2001). A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37(1):91–126.

Trmal, J., Zelinka, J., and Müller, L. (2010). Adaptation of a feedforward artificial neural network using a linear transform. In *Text, Speech and Dialogue*, pages 423–430. Springer.

Uebel, L. and Woodland, P. C. (2001). Improvements in linear transform based speaker adaptation. In *Proc. ICASSP*, pages 49–52.

Variani, E., Lei, X., McDermott, E., Moreno, I. L., and Gonzalez-Dominguez, J. (2014). Deep neural networks for small footprint text-dependent speaker verification. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4052–4056. IEEE.

Vertanen, K. (2004). An overview of discriminative training for speech recognition. *University of Cambridge*, pages 1–14.

Vesel`y, K., Ghoshal, A., Burget, L., and Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *Interspeech*, pages 2345–2349.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Vu, N. T., Weiner, J., and Schultz, T. (2014). Investigating the learning effect of multilingual bottle-neck features for ASR. In *Interspeech*, pages 825–829.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

Wang, D. and King, S. (2011). Letter-to-sound pronunciation prediction using conditional random fields. *IEEE Signal Processing Letters*, 18(2):122–125.

Wang, D. and Narayanan, S. S. (2002). A confidence-score based unsupervised MAP adaptation for speech recognition. In *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, volume 1, pages 222–226. IEEE.

Wang, S. I. and Manning, C. D. (2013). Fast dropout training. In *ICML (2)*, pages 118–126.

Wang, W., Stolcke, A., and Harper, M. P. (2004). The use of a linguistically motivated language model in conversational speech recognition. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–261. IEEE.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wester, M. (2003). Pronunciation modeling for ASR–knowledge-based and data-derived methods. *Computer Speech & Language*, 17(1):69–85.

Woodland, P. C. (2001). Speaker adaptation for continuous density HMMs: A review. In *ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*.

Wu, C. and Gales, M. J. (2015). Multi-basis adaptive neural network for rapid adaptation in speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4315–4319. IEEE.

Xu, H., Povey, D., Mangu, L., and Zhu, J. (2011). Minimum bayes risk decoding and system combination based on a recursion for edit distance. *Computer Speech & Language*, 25(4):802–828.

Xu, P. and Mangu, L. (2005). Using random forest language models in the IBM RT-04 CTS system. In *Interspeech*, pages 741–744.

Xu, W. and Rudnicky, A. I. (2000). Can artificial neural networks learn language models?

Xue, J., Li, J., Yu, D., Seltzer, M., and Gong, Y. (2014a). Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *Proc. ICASSP*, pages 6359–6363. IEEE.

Xue, S., Abdel-Hamid, O., Jiang, H., and Dai, L. (2014b). Direct adaptation of hybrid DNN/HMM model for fast speaker adaptation in lvcsr based on speaker code. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6339–6343. IEEE.

Xue, S., Abdel-Hamid, O., Jiang, H., Dai, L., and Liu, Q. (2014c). Fast adaptation of deep neural network based on discriminant codes for speech recognition. *Audio, Speech, and Language Processing, IEEE/ACM Trans. on*, 22(12):1713–1725.

Xue, S., Jiang, H., and Dai, L. (2014d). Speaker adaptation of hybrid NN/HMM model for speech recognition based on singular value decomposition. In *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*, pages 1–5. IEEE.

Yao, K., Cohn, T., Vylomova, K., Duh, K., and Dyer, C. (2015). Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*.

Yao, K., Yu, D., Seide, F., Su, H., Deng, L., and Gong, Y. (2012). Adaptation of context-dependent deep neural networks for automatic speech recognition. In *Proc. SLT*, pages 366–369. IEEE.

Yao, K. and Zweig, G. (2015). Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Yi, J., Ni, H., Wen, Z., Liu, B., and Tao, J. (2016). CTC regularized model adaptation for improving LSTM RNN based multi-accent mandarin speech recognition. In *Chinese Spoken Language Processing (ISCSLP), 2016 10th International Symposium on*, pages 1–5. IEEE.

Yoshioka, T., Karita, S., and Nakatani, T. (2015). Far-field speech recognition using CNN-DNN-HMM with convolution in time. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4360–4364. IEEE.

Yoshizawa, S., Baba, A., Matsunami, K., Mera, Y., Yamada, M., and Shikano, K. (2001). Unsupervised speaker adaptation based on sufficient HMM statistics of selected speakers. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 341–344. IEEE.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2002). The HTK book. *Cambridge university engineering department*, 3:175.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2006). The HTK book (for HTK version 3.4). *Cambridge university engineering department*, 2(2):2–3.

Young, S. J., Odell, J. J., and Woodland, P. C. (1994). Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, pages 307–312. Association for Computational Linguistics.

Yu, D., Chen, X., and Deng, L. (2012). Factorized deep neural networks for adaptive speech recognition. In *in Proc. Int. Workshop Statist. Mach. Learn. Speech Process*. Citeseer.

Yu, D. and Deng, L. (2014). *Automatic speech recognition: A deep learning approach*. Springer.

Yu, D., Deng, L., and Dahl, G. (2010). Roles of pre-training and fine-tuning in context-dependent dbn-hmms for real-world speech recognition. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Yu, D. and Seltzer, M. L. (2011). Improved bottleneck features using pretrained deep neural networks. In *Interspeech*, volume 237, page 240.

Yu, D., Xiong, W., Droppo, J., Stolcke, A., Ye, G., Li, J., and Zweig, G. (2016). Deep convolutional neural networks with layer-wise context expansion and attention. In *Proc. Interspeech*.

Yu, D., Yao, K., Su, H., Li, G., and Seide, F. (2013). KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *Proc. ICASSP*, pages 7893–7897.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, X., Trmal, J., Povey, D., and Khudanpur, S. (2014). Improving deep neural network acoustic models using generalized maxout networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 215–219. IEEE.

Zhang, Y., Chan, W., and Jaitly, N. (2016a). Very deep convolutional networks for end-to-end speech recognition. *arXiv preprint arXiv:1610.03022*.

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., and Courville, A. (2017). Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*.

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Laurent, C., Bengio, Y., and Courville, A. (2016b). Towards end-to-end speech recognition with deep convolutional neural networks. *Interspeech 2016*, pages 410–414.

Zhang, Z.-P., Furui, S., and Ohtsuki, K. (2000). On-line incremental speaker adaptation with automatic speaker change detection. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II961–II964. IEEE.

Zheng, H., Zhang, S., Qiao, L., Li, J., and Liu, W. (2016). Improving large vocabulary accented mandarin speech recognition with attribute-based i-vectors. In *Interspeech 2016*, pages 3454–3458.

Zhou, Y. and Chellappa, R. (1988). Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pages 71–78.

Zhu, L., Kilgour, K., Stüker, S., and Waibel, A. (2015). Gaussian free cluster tree construction using deep neural network. In *Interspeech*, pages 3254–3258.

Zhu, Z., Engel, J. H., and Hannun, A. (2016). Learning multiscale features directly from waveforms. *arXiv preprint arXiv:1603.09509*.