Implementation of DNN-HMM Acoustic Models for Phoneme Recognition

by

Sihem Romdhani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Sihem Romdhani

Abstract

Gaussian Mixture Model-Hidden Markov Models (GMM-HMMs) are the state-of-the-art for acoustic modeling in speech recognition. HMMs are used to model the sequential structure and the temporal variability in speech signals. However, GMMs are used to model the local spectral variability in the sound wave at each HMM state. Attempts to use Artificial Neural Networks (ANNs) to substitute GMMs in HMM-based acoustic models led to dismal results for many years. In fact, ANNs could not significantly outperform GMMs due to their shallow architectures. In addition, it was difficult to train networks with many hidden layers on large amount of data using the back-propagation learning algorithm.

In recent years, with the establishment of deep learning technique, ANNs with many hidden layers have been reintroduced as an alternative to GMMs in acoustic modeling, and have shown successful results. The deep learning technique consists of a two-phase procedure. First, the ANN is generatively pre-trained using an unsupervised learning algorithm. Then, it is discriminatively fine-tuned using the back-propagation learning algorithm. The generative pre-training intends to initialize the weights of the network for better generalization performance during the discriminative phase. Combining Deep Neural Networks (DNNs) and HMMs within a single hybrid architecture for acoustic modeling have shown promising results in many speech recognition tasks.

This thesis aims to empirically confirm the capability of DNNs to outperform GMMs in acoustic modeling. It also provides a systematic procedure to implement DNN-HMM acoustic models for phoneme recognition, including the implementation of a GMM-HMM baseline system. This thesis starts by providing a thorough overview of the fundamentals and background of speech recognition. The thesis then discusses DNN architecture and learning technique. In addition, the problems of GMMs and the advantages of DNNs in acoustic modeling are discussed. Finally, DNN-HMM hybrid acoustic modes for phoneme recognition are implemented. The deployed DNN is generatively pre-trained and fine-tuned to produce a posterior distribution over the states of mono-phone HMMs. The developed DNN-HMM phoneme recognition system outperform the GMM-HMM baseline on the TIMIT core test set. An in-depth investigation into the major factors behind the success of DNNs is carried out.

Acknowledgements

First and foremost, I would like to acknowledge my supervisor Dr. Fakhri Karray for his invaluable guidance, support, patience and encouragement all the way through my master study. I am very grateful for his expertise, time, and continued assistance.

I would like to thank all members of Center for Pattern Analysis and Machine Intelligence for their collaboration and help.

I would also like to express my gratitude to all my professors in University of Waterloo and in Tunisia National School of Computer Science for the quality of teaching they provided and for every bit of insight they caused.

I would also like to thank the Tunisian Government for funding my studies and for giving me the opportunity to join such notable and prestigious university.

Special thanks go to Dr. Farouk Mkadem for his support, advice, and assistance over the period of my studies at University of Waterloo.

Last but not least, I do not find suitable words to express my deepest gratitude and love to my husband for his patience, inspiration, and love. His continuous support and concern have made the achievement of this work possible.

Dedication

This work is nicely dedicated to my late father whom his memory has been a source of strength and inspiration, and to my mother for her immeasurable love and for all the sacrifices she has made to get me where I am today.

I also offer this work to my sister and brother for their love and support.

This work is also dedicated to my dear husband for always supporting me and above all for believing in me.

Table of Contents

Table of Contents viii

List of Tables ix

List of Figures x

Nomenclature xii

1 Introduction 1

2 Automatic Speech Recognition: Fundamentals and Recent Advances 5

2.1 Background of ASR Systems 6

2.1.1 Definition 6

2.1.2 The Source-Channel Model of Speech Recognition 6

2.2 Architectural Components of ASR 7

2.2.1 Speech Signal Processing 7

2.2.2 Acoustic Modeling 10

2.2.3 Language Modeling 11

2.2.4 Decoding 13

2.3 Challenges in ASR 14

2.3.1 Variability in Speech Signals 15

2.3.2 Environment Conditions 15

2.3.3 Noise-Robust Speech Recognition 16
2.4 State of the Art ASR 17
2.4.1 An Overview of the Historical Developments in ASR 17
2.4.2 The Recent History of Acoustic Modeling with DNNs for Speech Recognition 19
2.5 Conclusion 22

3 Deep Neural Networks **23**
3.1 Motivations for Training DNNs 23
3.2 Breakthrough of DNNs 26
3.3 Generative Pre-Training of DNNs 27
3.4 Deep Belief Networks(DBNs) for Generative Pre-Training 29
3.4.1 Restricted Boltzmann Machines and Contrastive Divergence Learning 30
3.4.2 Learning DBNs by Stacking RBMs 34
3.5 Discriminative Training of DNNs 35
3.6 Conclusion 36

4 DNN-HMM for Acoustic Modeling **38**
4.1 HMMs for Acoustic Modeling in Speech Recognition 39
4.1.1 The Forward Algorithm to Evaluate an HMM 42
4.1.2 The Viterbi Algorithm for Decoding an HMM 43
4.1.3 The Baum-Welch Algorithm for Estimating the HMM Parameters . 44
4.2 Limitations of GMMs in GMM-HMM Based Acoustic Models 47
4.3 DNN-HMM Hybrid Architecture for Acoustic Modeling 49
4.3.1 Interfacing DNNs with HMMs 50
4.3.2 Advantages of DNNs for Acoustic Modeling 52
4.4 Conclusion 53

5 Implementation of DNN-HMM Acoustic Models for Phoneme Recognition **55**

5.1 DNN-HMM Acoustic Models for Phoneme Recognition 56

5.1.1 Implementation Procedure of DNN-HMM 56

5.1.2 Challenges 61

5.2 Experimental Setup 61

5.2.1 Dataset Description 61

5.2.2 Computational Setup 62

5.3 Experiments 64

5.3.1 Building the GMM-HMM Baseline System 64

5.3.2 Training the DNN 66

5.3.3 DNN-HMM Results and Analysis 67

5.4 Conclusion 74

6 Conclusion and Future Works **75**

References **78**

List of Tables

5.1 Partial list of HMM state ID -DNN class label ID mapping 60

5.2 Mapping the TIMIT phonemes into the standard phonemes 63

5.3 Recognition Performance on TIMIT core test set of the GMM-HMM baseline as a function of the number of Gaussian mixture components 64

5.4 GMM-HMM performance with different values for Viterbi parameters . . 66

5.5 Summery of DNN training time with CPU and GPU 67

5.6 The performance of 3-hidden layer DNN vs. the number of hidden neurons 69

5.7 Performance of the DNN with different number of hidden layers and different number of pre-training epochs 70

5.8 Effect of varying the values of LMSF and WIP on recognition accuracy . . 73

5.9 Reported results on TIMIT core test set 73

List of Figures

2.1 Source-channel model of speech recognition [41] 7
2.2 Architectural components of ASR system 8
2.3 Speech signal processing steps 9
2.4 Illustration of a five-state left-to-right HMM 11
2.5 Illustration of the speech recognition process [82] 14

3.1 Feed-forward ANN with multiple inputs, two outputs and one hidden layer 24
3.2 Sigmoid Belief Network composed of binary units 28
3.3 Sigmoid Belief Network with many hidden layers 29
3.4 Illustration of an RBM 30
3.5 Illustration of the Contrastive Divergence learning procedure for RBMs [33]. 33
3.6 A pseudo code for training an RBM with K iterations CD [74] 33
3.7 Creating a DBN by stacking three RBMs 35
3.8 Creating a DNN from a DBN by adding a Softmax output layer on the top 36

4.1 Frame-level DNN for phoneme recognition [55] 51
4.2 Interfacing DNN and HMM for continuous speech recognition [84] 52

5.1 Parameters for coding the speech data 56
5.2 Mono phone 5-states HMM with single Gaussian at each state 57
5.3 GMM-HMM phone accuracy rate and average training time vs. the number of Gaussian mixtures at each HMM state 65

5.4 Validation error of DNN with different number of hidden layers vs. the number of fine-tuning epochs 71

5.5 Validation error of DNN with different fine-tuning rates vs. the number of fine-tuning epochs 72

Nomenclature

AI	Artificial Intelligence
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CD	Contrastive Divergence
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
EM	Expectation Maximization
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GPGPU	General Purpose Graphical Processing Unit
GPU	Graphics Processing Unit
GRBM	Gaussian-Bernoulli Restricted Boltzmann Machine
HMM	Hidden Markov Model
HPC	High Performance Computing
HTK	Hidden Markov Model Tool Kit

IVR	Interactive Voice Response
LMSF	Language Model Scaling Factor
LPC	Linear Prediction Coefficient
LW	Language Weight
MFCC	Mel Frequency Cepstral Coefficient
MLE	Maximum Likelihood Estimation
MRF	Markov Random Field
PAR	Phone Accuracy Rate
PER	Phone Error Rate
PLP	Perceptual Linear Prediction
RBM	Restricted Boltzmann Machine
SBN	Sigmoid Belief Network
WIP	Word Insertion Penalty

Chapter 1

Introduction

Speech is the most direct and appropriate mode of communication between people. Building Artificial Intelligence (AI) systems capable of naturally communicating with humans has attracted a great deal of attention for centuries. Automatic recognition and understanding of fluently spoken language are considered the first and most important steps toward effective natural human-machine communication [43]. Over the past decades, Automatic Speech Recognition (ASR) systems have shown enormous development and significant improvement in their accuracy, which make it possible now for people to efficiently control and interact with their devices using only their voices. Today, we find ASR engines deployed and integrated in computers, cars, mobile phones, aircraft systems, robots, games, and many other devices. Their usage has pervaded widespread applications in several tasks such as voice dialling, call routing, Interactive Voice Response (IVR), data entry, dictation and speech-to-text translation, machine command and control, computer aided-language learning, and speech-to-speech translation [81, 38]. Thus, ASR systems have made life easier for everyone, especially those with disabilities. However, these systems are still facing many challenges such as in achieving robustness, reliability, and building engines that are not heavy in memory.

The process of building ASR systems is composed of many sub-processes, mainly signal processing, feature extraction, acoustic modeling, and language modeling. In fact, acoustic modeling in speech recognition refers to the process of creating statistical representations for the feature vector sequences computed from speech waveforms. These acoustic models should be robust and capable of modeling not only the variability intrinsic in human speech related to accent, dialect, pronunciation differences, etc., but also the noise that charac-

terizes the background environment and is considered the main reason behind distortion of captured speech.

Hidden Markov Models (HMMs) are typically the most useful acoustic models in speech recognition. The main reasons behind the success of HMM in acoustic modeling are the richness of the immanent mathematical statistical framework [44, 64], the availability of training and decoding algorithms, and the flexibility of its structure. This flexibility is due to the easiness when changing the size, type, or architecture of the HMM in order to fit different recognition units (phonemes, words, or sentences) [44]. In addition, HMMs are easy to implement for speech recognition tasks; indeed, the model parameters can be automatically learned and estimated from speech training data [82].

HMMs are used to model the sequential structure of speech signals [55, 58] with Gaussian densities at each state in order to model the local spectral variability in the sound wave[55]. In fact, Gaussian Mixture Models (GMMs) are the dominant technique for modeling the emission distribution of the HMMs' states. Each emitting distribution depicts a particular sound event, and thus "the distribution must be specific enough to allow discrimination between different sounds as well as robust enough to account for the variability in natural speech" [81]. Indeed, with enough components, GMMs can accurately model the probability distribution over the acoustic input feature vectors associated with each state of an HMM [32]. Nonetheless, GMMs in HMM-based acoustic models suffer from some major limitations (discussed in detail in Chapter 4 Section 2) that have motivated researchers to find other models.

In the late 1980s and early 1990s, Artificial Neural Networks (ANNs) that have shown great capability in modeling highly non-linear data were used to replace GMMs in HMM-based acoustic models. However, they could not significantly outperform GMMs due to their shallow architectures. In addition, it was not possible to train networks with many hidden layers on large amount of data using the back-propagation learning algorithm [32].

The introduction of a new procedure [33] for learning Deep Neural Networks (DNNs) with many hidden layers of nonlinear units, in 2006, represented a paradigm shift in the area of machine learning and artificial intelligence. In fact, this procedure involves two steps: i) a generative layer-by-layer pre-training step that aims at extracting higher new representation of the input data at each layer without any knowledge about the labels; ii) a subsequent discriminative training using the back-propagation learning algorithm. This

technique has shown great success in many applications such as dimensionality reduction [34], 3-D object recognition [62], and information retrieval [35].

Since 2009, DNNs have been successfully reintroduced as an alternative to GMMs for acoustic modeling, and have shown excellent performance (e.g., [56, 32, 14, 13, 55, 84, 60, 70]). The establishment of the new deep learning procedure was the major factor in successfully using neural networks that have many hidden layers as high-quality HMM-based acoustic models. In [18], it has been asserted that there are three other major factors behind the interesting results of these deep models. First, better results can be achieved by increasing the number hidden layers. Second, the new way of initializing the weights and the availability of more powerful hardware enabled the training of DNNs. Finally, the ability to handle a large output layer has a great impact on the performance of the networks as acoustic models. Indeed, deep neural networks have shown capability to outperform GMMs, and to improve the accuracy of speech recognizers.

The development of DNN-HMM hybrid acoustic models is growing rapidly and on a large scale [19, 18]. However, the availability of tools or open source libraries for implementing these hybrid acoustic models is quasi absent, and the implementation procedure remains ambiguous, due to the fact that the approach was initially based on intuitive guesses [18].

Objectives and Outline

The objective of this thesis is to implement DNN-HMM hybrid acoustic models for phoneme recognition, with the aim of empirically confirming the potency and effectiveness of DNNs in acoustic modeling and their impact on the performance of the speech recognizer.

This thesis focuses on three areas:

1. Comprehensive study of the structure and learning technique of DNNs, initially proposed in [33].
2. Systematic procedure to implement DNN-HMM acoustic models for phoneme recognition is described, including the implementation of GMM-HMM baseline and how interfacing HMMs with DNN in order to replace GMMs.

3. Investigation into the major factors behind the power of DNN-HMM in acoustic modeling.

This thesis is organized as follows:

- Chapter 1 serves as introduction to the motivations and objectives of this thesis.
- Chapter 2 starts with an overview of the fundamentals of ASR systems and their components. The challenges that are facing this area are then discussed. Finally, a review of the historical developments and milestones of ASR systems is presented, with an emphasis on the recent advances in acoustic modeling using DNNs.
- Chapter 3 is dedicated to a comprehensive study of DNNs, namely their structures and learning technique.
- Chapter 4 describes the DNN-HMM hybrid architecture for acoustic modeling. It starts with an overview of HMMs, their parameters and their learning and decoding algorithms. The major problems of GMMs in acoustic modeling are then discussed. Finally, the DNN-HMM hybrid acoustic models are explored and the advantages of DNNs are revealed.
- Chapter 5 starts with a description of the step-by-step implementation procedure of DNN-HMM acoustic models for phoneme recognition, including the implementation of the GMM-HMM baseline. Then, several experiments are conducted to study the behaviour of the DNNs, and the impact of varying the number of layers and the number of units per layer on the recognition performance. In addition, the role of the pre-training and fine-tuning steps in improving the recognition accuracy are discussed. The efficiency of DNNs at acoustic estimation is also studied.
- Finally, a summary of the work is presented along with a discussion of potential future research directions.

Chapter 2

Automatic Speech Recognition: Fundamentals and Recent Advances

The field of ASR has become an important area of research and is attractive to industry. In fact, it has known impressive progress and significant improvements during the past fifty years or so. These successful developments have enabled this technology to be effectively applied in a growing number of everyday applications [6]. However, developing an ASR system is a difficult task involving a variety of disciplines such as signal processing, pattern recognition, statistics, phonetics, linguistics, neuroscience, and computer science [46]. Many milestones have marked the history of ASR; however, the recent emergence of Deep Neural Networks (DNNs) for acoustic modeling in speech recognition has represented a paradigm shift in the area.

Section 1 of this chapter presents background on ASR systems, mainly the definition and the source channel model. Section 2 provides a thorough overview of the architecture and basic components of ASR systems, including a general description of the statistical mathematical formulation. The challenges and difficulties that have faced this field and constrained its progress toward more robust and reliable systems are discussed in detail in Section 3. Finally, Section 4 reviews the major technological developments and milestones in this field, with emphasis on the recent advances in acoustic modeling for speech recognition. In addition, it sets out the historical context in which DNN-based acoustic models have been developed.

2.1 Background of ASR Systems

2.1.1 Definition

An ASR system is defined as the process for decoding and converting speech signals into a set of linguistic units, typically in terms of a sequence of words, by mean of computer programs [82, 81]. The recognized words can then be used to perform several tasks: they can be translated into written text, and/or be used to command or control another system or machine or to access another device or application [46]. This mapping of spoken words into actions is considered as a separate process called speech understanding. In fact, the focus in this thesis is on speech recognition and translation of the acoustic signal into text and not on extracting the meaning of the speech.

2.1.2 The Source-Channel Model of Speech Recognition

The source channel of speech recognition can be schematically illustrated as in Figure 2.1. The speaker's mind is considered the source of the communication; it decides and generates the word sequence W. Then, the speaker's vocal apparatus, which represents the speech producer, pronounces the words. The produced speech passes through the signal-processing component of the speech recognizer, which transforms it into an acoustic signal X. Together, the speech producer and the acoustic processor constitute a noisy communication channel. Finally, the speech decoder transforms the signal X into the most likely word sequence $\widehat{W}$ that best fits the uttered word sequence W.

In ASR, the simplest approach to finding the most probable word sequence that best matches the observed acoustic data is to collect several samples of acoustic data for each possible word sequence in the language, and then compute a statistical similarity metric for the observed acoustic feature vector sequence and the known samples. For a large vocabulary, this approach may be computationally very expensive and hence not feasible due to the large number of possible word sequences. Instead, it is more fruitful to consider the basic sounds, known as phonemes, from which words in the vocabulary are composed. Thus, computing the statistical correspondence between the basic sounds that are considered to be the basic speech recognition units and the given acoustic data becomes easier. Indeed, the problem of finding the appropriate word sequence in speech recognition is often mathematically formulated using a generative statistical model, as discussed in the following sections.

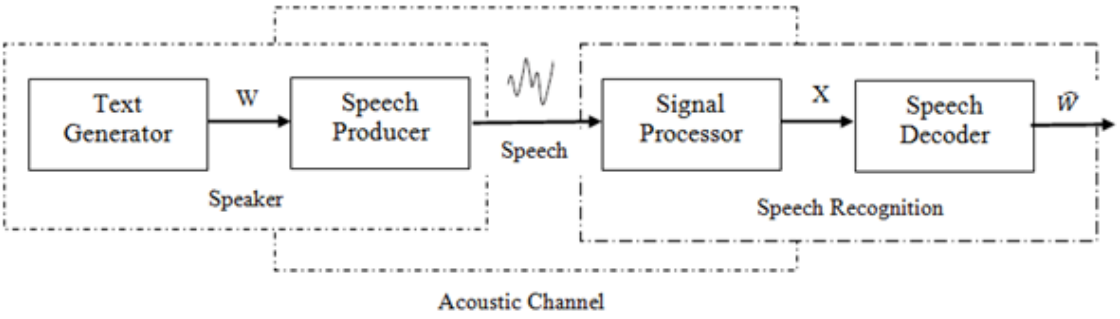

Figure 2.1: Source-channel model of speech recognition [41]

The next section highlights the architecture of a speech recognizer, and describes the components that are of concern in the design of the speech decoder, and their roles.

2.2 Architectural Components of ASR

The architecture of an automatic speech recognition engine, as illustrated in Figure 2.2, typically encloses four basic components: a signal processing component, an acoustic modeling component, a language modeling component, and a decoder. The role of the speech-signal processing component is to extract salient features from the speaker's utterances and feed them to the decoder, which then attempts to find the underlying word sequence using both the acoustic and language models. Acoustic models are designed to model the spectral representation of sound waves and to deal with the variability intrinsic in speech signals, related to the gender, accent, dialect and other differences among speakers, and also with the variability embedded in the speech environment. In contrast, the language models focus on the grammar of the vocabulary used. They are designed to describe how a sequence of linguistic symbols (e.g., phonemes, syllables, and words) is constituted, and how likely these symbols use to appear and co-occur in a specific order.

Next, the role of each component is defined separately.

2.2.1 Speech Signal Processing

Signal processing is a crucial component in speech recognition systems. Its role is to do spectral analysis of the sound waves, eliminate any latent noise, and extract meaningful

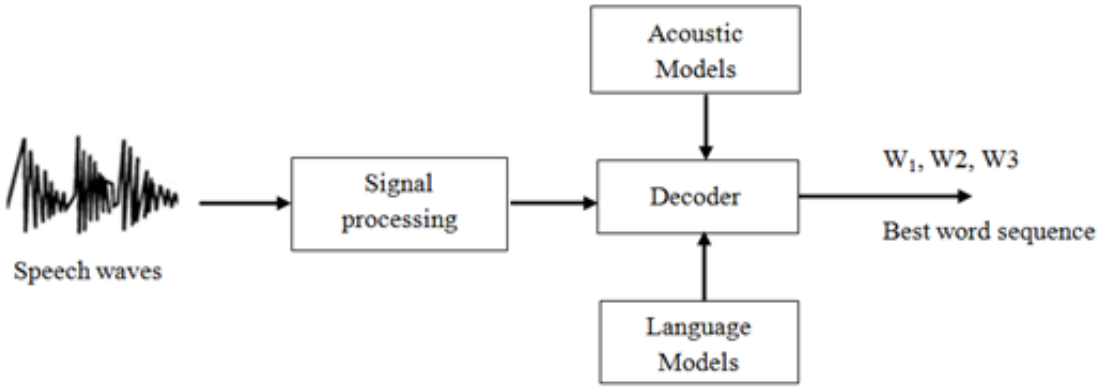

Figure 2.2: Architectural components of ASR system

and salient information from the speaker's utterance that are useful for recognition.
The process involves many steps, as illustrated in Figure 2.2. The first is to capture the speech pressure waves using a microphone. The recorded sound, which represents a time-varying analogue signal, is then transferred to a computer memory as a series of digitalized samples. The sampling is performed with respect to a specific rate. The second step is to detect the end-points of the utterance in order to delete any background noise. This can be done by accurately identifying where speech is present in the captured signal. Different techniques are used for this purpose, some are manual, including the push-to-talk mode where a specific action is needed from the speaker to indicate the beginning and the end of the utterance, and the hit-to-talk technique, where the speaker identifies the beginning of the utterance but the ends are automatically determined. However, in the continuous-listening technique, the system listens to the incoming sound continuously, and both the beginning and the end of any potential speech are automatically detected. The automatic detection is often an energy-based technique, where a threshold is applied to discriminate between the speech region with high energy and the non-speech region with low energy.

The last step in signal processing consists of extracting reliable features from the speech signal. Speech is a time-varying process; hence, the most direct representation is in the form of temporal waveforms [46]. This feature representation is not useful for speech recognition tasks. Indeed, it has been proven that the frequency-domain features are more beneficial and more accurate than the time-domain features for speech recognition. They are capable of discriminating between the elementary sounds of speech, akin to what the human ear

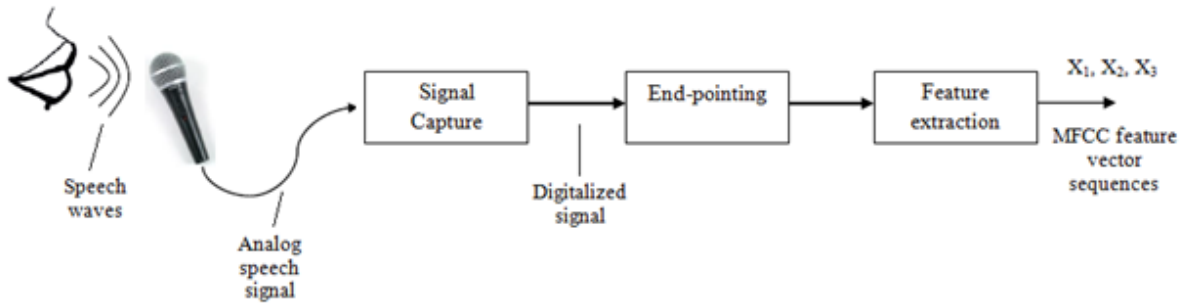

Figure 2.3: Speech signal processing steps

does during the auditory process [46], and also depicting the most useful information, such as formants used to recognize vowels [38]. There are different representations of the frequency-domain features including Linear Prediction Coefficients (LPCs), Perceptual Linear Prediction (PLP), and Mel Frequency Cepstral Coefficients (MFCCs), this last currently represent the most widely used acoustic features. To extract the MFCCs feature, the Discrete Fourier Transform (DFT) is applied to short-time Hamming windows of the signal shifted by a small interval, and a set of Mel scale filters is then applied to generate the Mel filter bank. Later, the log operator is used to get the log-filter-bank output. Finally, the Discrete Cosine Transform (DCT) is used to generate the MFCCs. It has been experimentally proven that the first 13 MFCCs enclose the most important information needed for recognition. In order to determine the temporal change in the spectra, the delta coefficients are calculated and incorporated in the feature vector. The feature vector used in speech recognition is composed of three elements:

$$X_k = \begin{pmatrix} c_k \\ \Delta c_k \\ \Delta\Delta c_k \end{pmatrix} \tag{2.1}$$

where c_k is the 13^{th} order MFCC, Δc_k is the first-order delta MFCC, and $\Delta\Delta c_k$ is the second-order delta MFCC. For a 16-kHz sampling rate of the signal, the 40-msec delta coefficients are calculated as follows [38]:

$$\Delta c_k = c_{k+2} - c_{k-2} \tag{2.2}$$

$$\Delta\Delta c_k = \Delta c_{k+1} - \Delta c_{k-1} \tag{2.3}$$

2.2.2 Acoustic Modeling

Building acoustic models that can deal with speaker variations, context variations, and environment variations plays an important role in the development of an accurate and reliable speech recognition system. In fact, acoustic models are considered as patterns for every speech unit in the language. Thus, selecting the appropriate basic unit to represent the acoustic and language information is the first important step for designing speech recognition systems. For a large vocabulary, it is difficult to consider words as basic modeling units. In fact, doing so may require a significant number of training data, which is difficult to collect; on the other hand, it is difficult to model all of the inherent variations. Actually, three issues must be considered when selecting modeling units: the units must be accurate, trainable, and generalizable so that they can be adapted to any new word in the vocabulary [38]. Alternatively to the word models that may be practical for small-vocabulary speech recognition, phonemes are the most suitable basic units for many speech recognition tasks. They are more generalizable and can be trained with few data [38]. In fact, phonemes are context-independent models. They are considered the most elementary units that encompass the articulatory gestures of the language. However, for large-vocabulary speech recognition systems, tri-phones are often used as basic modeling units. They encompass the left and right neighbouring phonemes in order to model the contextual dependencies. They aim at modeling the most important pronunciations and coarticulatory effects, which make them more suitable and more consistent than context-independent phone models [38].

After selecting the modeling units, acoustic modeling is intended to create efficient representation of the speech feature vector sequences extracted for each unit in the training data. After extracting the useful features, a sequence X of spectral representation (e.g., a sequence of MFCC vectors) is generated. The goal of acoustic modeling is then to estimate the probability, $P(X|W)$, of this sequence of acoustic feature vector given the word (or phonetic) unit, W. Hence, when a new acoustic input observation is present, it can be associated with the appropriate word based on this acoustic probability. In an isolated-word speech recognition system with an N-word vocabulary, each word should have its own acoustic model; thus $P(X|W_i)$ is equivalent to $P(X|\alpha_i)$, where α_i corresponds to the acoustic model for the i^{th} word W_i.

Acoustic modeling of speech units is generally based on the statistical models used to compute the probability $P(X|W)$. Due to their performance and their ease of implementation [44], Hidden Markov Models (HMMs) have been considered the state-of-the-art and

the standard technique for acoustic modeling. An HMM is defined as a finite state machine, where a transition from one state i to another state j is performing every time frame, t, with a probability a_{ij}. On entering the state j, an observation vector x_t is generated with respect to a probability distribution $b_j(x_t)$. Indeed, in a HMM-based speech recognition system, each sequence of an acoustic feature vector corresponding to each speech unit is considered to be generated by a Markov chain. As illustrated in Figure.2.4, an HMM has two states without probability emission, one at the entry of the chain reached before the start of the speech observation vector generation process, the second located at the end and reached when the generation process is finished. The states in between are characterized by an emission probability density often estimated using Gaussian Mixture Models (GMMs). For each emitting state, the HMM is allowed to remain at the same state or move to the next state. For training the HMM's parameters, acoustic training data are needed for each speech unit. These data can be recorded from the same speaker or from different speakers. In the first case, the recognizer is characterized as speaker-dependent, and in the second case, the recognizer is characterized as speaker-independent.

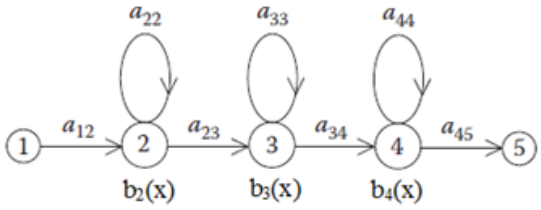

Figure 2.4: Illustration of a five-state left-to-right HMM

More details about the HMM's structure and parameters, the learning algorithm, the evaluation, and the decoding algorithms as well, are discussed in Chapter 4.

2.2.3 Language Modeling

Language modeling is defined as a statistical representation of grammar. It defines a set of syntactical rules that formally describes any permissible combinations of language units (e.g., phonemes, words, etc.). With a word-based unit, the role of language models is to assign a probability estimate, $P(W)$, to every possible meaningful sequence of words, $W = w_1, w_2, w_3, , w_n$, that could appear in the speech signal. A parsing technique is often used to analyze the sentence and to confirm whether its structure is compliant with the grammar.

The earlier method for language modeling was the traditional deterministic grammar that assigns the value of one to the probability of any syntactically correct structure and the value of zero to any impermissible incorrect structure. Actually, the stochastic language models, such as n-gram models that define the probability of occurrence of an ordered sequence of n words, have become the most frequently used technique in speech recognition.

The probability of a word sequence, $P(W)$, can be formulated as a chain of rules as follows:

$$P(W) = P(w_1, w_2, w_3, ..., w_n) \tag{2.4}$$

$$P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, w_2, ..., w_{n-1}) \tag{2.5}$$

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, w_2, ..., w_{i-1}) \tag{2.6}$$

where $P(w_i|w_1, w_2, ..., w_{i-1})$ is the probability that the word w_i will be introduced, given that the word sequence $w_1, w_2, ..., w_{i-1}$ was determined earlier. Hence, the prediction of w_i depends on all the history of the input sequence. It has been proven that it is impossible to determine the probability, $P(w_i|w_1, w_2, ..., w_{i-1})$, even for a small value of i [5], due to the large number of possible combinations for the input history. The N-gram language models came to resolve this problem by limiting the input sequence history and defining the word relationship within a span of n words. Hence, the probability of the predicted word depends only on the $n-1$ previous words, $p(w_i|w_{i-N+1}, w_{i-N+2}, ..., w_{i-1})$.

If $N = 3$, it is then a tri-gram model where w_i depends only on the two preceding words, $P(w_i|w_{i-2}, w_{i-1})$. This kind of model is considered to be the most powerful and convenient language model, since words in any grammar generally depend on the two previous words. In addition, tri-gram models can be estimated with a sufficient number of training data (a corpus). For some speech recognition tasks, it is possible to use bi-gram models that consider only the immediate precedent word, $P(w_i|w_{i-1})$, or uni-gram models, $P(w_i)$.

In a tri-gram model, the probability $P(w_i|w_{i-2}, w_{i-1})$ is calculated as follows:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{Count(w_{i-2}, w_{i-1}, w_i)}{Count(w_{i-2}, w_{i-1})} \tag{2.7}$$

where $Count(w_{i-2}, w_{i-1}, w_i)$ is the number of times the ordered sequence w_{i-2}, w_{i-1}, w_i occurs in the corpus, and $Count(w_{i-2}, w_{i-1})$ is the number of the occurrences of the word pair sequence w_{i-2}, w_{i-1} in the same corpus.

2.2.4 Decoding

The decoding process seeks to find the word sequence that best fits the acoustic observations, and this is done by taking into consideration the trained acoustic models and the language models. This problem can be mathematically formulated using a statistical framework. In fact, as illustrated in Figure 2.5, the salient features extracted from the speech signal are represented by a sequence of observation vectors X, $X = x_1, x_2, ..., x_T$, where x_t is the acoustic observation at time t. The role of the decoder is to figure out the optimal sequence of words $\widehat{W} = w_1, w_2, ..., w_n$, corresponding to the spoken utterance with the minimum errors. Indeed, the determination of the underlying word sequence $\widehat{W}$ can be achieved by maximizing the posterior probability, given the speech signal, $P(W|X)$, as expressed by the following equation(2.8):

$$\widehat{W} = \underset{W}{argmax} P(W|X) \tag{2.8}$$

According to Bayesian decision theory, solving the fundamental equation of speech recognition is equivalent to solving the following equation(2.9):

$$\widehat{W} = \underset{W}{argmax} \frac{P(W)P(X|W)}{P(X)} \tag{2.9}$$

Since the acoustic data X is fixed, maximizing the above equation of the most probable word sequence $\widehat{W}$ is equal to maximizing the following equation(2.10):

$$\widehat{W} = \underset{W}{argmax} P(W)P(X|W) \tag{2.10}$$

where $P(W)$, which is estimated by the language model independently of the acoustic data, is the prior probability of the word W, and $P(X|W)$, which is determined by the acoustic model, is the probability of the acoustic observation sequence X, given that the target word of the speaker is W.

This decoding process can be considered a search process that consists of searching all possible word sequences and selecting the one with the highest posterior probability given the acoustic feature vectors. However, speech recognition is a very difficult task due to the variable-length non-stationary characteristics of the speech waveform. Hence, adopting the most appropriate search algorithms is very important for accurate results. The time-synchronous Viterbi algorithm is a powerful search tool often used in HMM-based speech recognition systems. A description of this search method is presented in Chapter 4.

However, to combine the acoustic model and the language model, the two probability estimations should be balanced [21]. In fact, the acoustic model probabilities are often underestimated due to HMM independence assumption [38]. If the context dependencies in the speech were considered, higher probabilities would be assigned to the acoustic models. To balance the two probability estimations, a weight called the Language Weight (LW) or Language Model Scaling Factor (LMSF) is added to the language model in order to raise its probability estimation. This weight is typically > 1. In addition, a new Word Insertion Penalty (WIP) is applied. Thus, the new speech coding equation becomes:

$$\widehat{W} = \underset{W}{argmax} P(W)^{LMSF} WIP^{N(W)} P(X|W) \tag{2.11}$$

where $N(W)$ is the number of words in sentence W. The values of the LMSF and WIP are determined empirically to optimize the recognition performance.

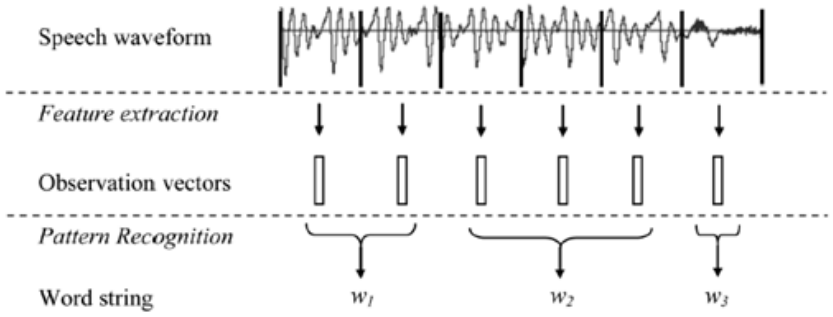

Figure 2.5: Illustration of the speech recognition process [82]

The next section discusses the major difficulties and challenges facing this area.

2.3 Challenges in ASR

Building accurate and robust speech recognition systems remains the major challenge for both researchers and industry. In the laboratory, it is possible to build ASR systems with high performance; however, moving to real-world conditions, these systems dramatically degrade in performance. There are many reasons why existing systems have not delivered results comparable to human performance in recognizing and understanding natural spoken words. Two major problems are encountered when building speech-based applications,

and must be dealt with to reach the expected recognition results: variability in the speech signals and environmental conditions. The noise that characterizes the speech environment is the main reason behind distortion of the captured speech and hence degradation of ASR system recognition performance. Many techniques have been proposed for dealing with the variability inherent in human speech and to build noise-robust ASR systems.

2.3.1 Variability in Speech Signals

With different speakers having different accents and dialects, ASR systems must deal with the problem of speech variability from one speaker to another. In fact, the sex, the age, the emotion and the health of the speaker greatly influence the style and the form of the produced speech, as do physical characteristics such as the size of the vocal tract and the length and width of the neck. It is not even possible for the same speaker to produce exactly the same utterance many times, which make it difficult to build highly accurate acoustic patterns. In addition, spontaneous speech phenomena (e.g. um's, ah's, false starts, out-of-vocabulary words, etc.) can significantly affect recognition performance. Added to that, natural human speech contains dramatic context variability at the word level and/or sentence level. Many words or sequences of words have the same pronunciation but different meaning (e.g., *write* and *right*, *Iscream* and *Ice cream*). Introducing silence between words is an alternative used to reduce the context variability at sentence level when dealing with continuous speech recognition tasks [38]. However, many techniques are needed to deal with the acoustic variability intrinsic in the speech signal with the aim to improve recognition performance.

2.3.2 Environment Conditions

The background noise that characterizes the environment of the speaker can drastically alter system performance. This noise may come from other people speaking or doing noisy actions like closing a window or using keyboards, etc., or from background natural conditions such as rain or wind. Furthermore, many other factors can distort the captured speech, including the quality of the microphone used and its position, or the noise embedded with the transmission tools (e.g., a telephone). In addition, the speakers themselves can produce noise in the speech with lip movement and non-communication words. Indeed, dealing with speech distortion due to environment conditions remains a big challenge in developing robust speech recognition systems.

2.3.3 Noise-Robust Speech Recognition

Noise robustness in speech recognition is related to the non-degradation of the system's accuracy under acoustic distortion conditions. To build robust ASR system capable of dealing with the noise in the acoustical environment, many techniques have been introduced over the last years and have had an important impact in the improvement of these systems toward becoming real applications.

The most obvious way to deal with environmental noise is to train the acoustic models using noisy data collected from all possible environments. This technique is referred to as multi-style training [52], but collecting these data is practically infeasible, and it is also difficult to build a model that can encompass all the variations. Several other techniques have been introduced and showed promising results in dealing with noisy environments.

In [49], the noise-robust existing techniques were categorized following five different criteria. The first category is Feature-Domains vs. Model-Domains. In the model-domain methods, the parameters of the acoustic model are modified to incorporate the effects of the distortion in the speech signal caused by noise. In contrast, feature-domain methods consist of using auditory features that are more robust to noise, either by applying new signal processing methods such as Relative spectral processing (RASTA) [29, 28], or by modifying test features, for example, by removing the noise embedded in the speech signal using different methods such as spectral subtraction [10] or a Wiener filter [51, 63].
The second category of noise-robust techniques involves methods that use a prior knowledge about the speech signal distortion. This prior knowledge can be learned from mapping between clean and noisy speech, using methods like empirical Cepstral compensation [71], or from data collected from different noisy environments.
The third category comprises noise-robust techniques that use an explicit environment-distortion model to predict the distorted speech from the clean speech.
Compensation techniques that focus on processing the uncertainty coming from strong noise and immanent in either the model space or feature space constitute the fourth category stated in [49]. In fact, the idea of these techniques is to incorporate this uncertainty in the statistical Bayesian framework of ASR as introduced in some works like in [42, 5].
The last category involves a joint model training method that is intended to train the acoustic models jointly with the same process used in the testing stage, rather than train the models using clean speech and then noise robustness techniques are used at testing stage to reduce the differences between clean acoustic models and distorted speech. Indeed, either feature enhancement or model adaptation techniques should be jointly applied at training and testing stages. The most well known technique in this category is noise adaptive training (NAT)[17].

Despite the notable success of many noise-robust techniques and the significant progress being made in the field, variability in the acoustic environment remains one of the most critical challenges facing state-of-the-art ASR systems. The next section discusses the major milestones in the history of ASR.

2.4 State of the Art ASR

In this section, the history of ASR is briefly reviewed, and the current state of the art is highlighted. Afterwards, the evolution in acoustic modeling since the emergence of deep neural networks is discussed.

2.4.1 An Overview of the Historical Developments in ASR

Early work on ASR originated in 1952 in the USA, with the work of Davis and his colleagues at Bell Laboratories, that established the first speaker-dependent speech recognizer able to understand isolated digits. Subsequent efforts in the United States, Japan and England in the late 50s and early 60s produced several hardware devices capable of recognizing human speech (for more detailed history about this efforts see [45]). In fact, the early-developed systems were mainly hardware based with limited use of software technology. Later researchers moved towards creating algorithm and mathematical frameworks.

A significant progress was made in the 1970s with the development of the Speech Understanding Research program nurtured by the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defence. This program developed among others, one of the major ASR systems, named HARPY [53], in 1976 at Carnegie Mellon University, which was capable of recognizing more than 1000 words with reasonable accuracy [45]. In the same period, two different works were achieved and made major contributions to the evolution of speech recognition technology. The first work was conducted by Fred Jelinek at IBM and focused on the statistical grammatical rules for language modeling, and introduced the n-gram models that evolved in the 1980s and have become the most powerful technique for language modeling in large vocabulary speech recognition systems [45]. The second work was conducted at AT &T Laboratories and focused on the spectral representation and the acoustic modeling of speech signals [45]. In fact, these two different efforts were considered as the first step toward the establishment of the statistical framework and the mathematical formalism for speech recognition.

This decade was also marked by two important milestones in speech recognition technology. One was the understanding of large vocabulary continuous speech; the other was the establishment of pattern recognition template-based methods. Furthermore, different additional techniques, such as dynamic-programming-based time wrapping algorithms and the Linear Predictive Coding (LPC) for speech analysis, were introduced to improve the process of speech recognition and they still have maintained their effectiveness [46].

Fundamental progress in the area of speech recognition was made in the 1980s, with the establishment of the statistical framework for speech recognition based on HMMs mainly derived from the work of Ferguson in 1980, and Rabiner in 1989 [81, 46]. This statistical framework came to substitute the template-based methods that showed difficulties generating representative templates for speech units. Hence, it represented a paradigm shift in the field. Many algorithms were introduced to improve the HMM-based statistical framework including the Expectation Maximization (EM) algorithm and the Forward-Backward or Baum-Welch algorithm for training HMMs [81]. Indeed, the trainability and the power of modeling the speech variability have allowed HMMs to remain the predominant technique in speech recognition so far.

Nevertheless, alternative approaches to HMM-based systems were introduced in the following years and showed competitive recognition performance, such as neural-networks-based systems. In fact, Artificial Neural Networks (ANNs) were first used to [75] emulate the work of HMMs in speech recognition and to carry out the recognition task using the discriminative algorithm to classify phonemes or words. Some early works in this approach are presented in these papers [77, 76, 24, 23, 73, 67]. In order to handle the temporal dependencies in the speech signal, two major classes of neural networks were used [75]: time-delay neural network and recurrent neural network. In spite of the successful results in phonemes classification tasks, ANNs were not able to effectively recognize continuous speech with long time-sequences, or to deal with large vocabulary recognition tasks. Although ANNs failed as a general framework for ASR [75], they proved their ability to contribute to the development of this field due to their strength at modeling highly nonlinear data. This fact motivated researchers to use ANNs in different ways within the system. In the early 1990s, the approach of combining HMMs and ANNs within a single hybrid model [22, 61] was introduced and showed significant improvements in the performance of the ASR systems in different tasks. In fact, ANNs were introduced to estimate the posterior probability of HMM states given the acoustic observations (see a comprehensive review in [11]).

During the last decades, ASR has shown notable progress and achieved impressive results even in real-world environments, despite the challenges and difficulties summarized in the previous section. In [82, 81, 38], it was asserted that this large-scale success of ASR

systems was derived from four major sets of factors working together at many levels to help this technology reach widespread commercial application. The first set involves the evolution of computation technologies and low-cost memory, the availability of a wide range of speech corpora for training and building acoustic and language models, and the regularization and distribution of these corpora by different organizations and communities such as the Linguistic Data Consortium (LDC), National Institute of Standard and Technology (NIST), and the European Language Resources Association (ELRA). Added to these factors, the adoption of more-rigorous specific evaluation measures and standards had a great impact in the evolution of ASR. The second set arose with the development of speech signal representations techniques such as Mel Frequency Cepstral Coefficients (MFCCs) and Perceptual Linear Prediction (PLP) coefficients, which all played an important role in the improvement of acoustic modeling in speech recognition. The third set implies a particularly significant achievement consisting in the establishment of sophisticated modeling tools and algorithms including HMM-based acoustic models and N-gram-based language models. Finally, the fourth set involves the application of advanced search strategies such as the Viterbi algorithm and stack decoding (A*search), which has notably contributed to more robust systems.

Indeed, thanks to these advances at different levels, speech recognition technology has been an ongoing topic of keen interest, and many research labs and industries have emerged in the field, leading to the introduction of a growing number of software tools such as CMU Sphinx, Hidden Markov Model Tool Kit (HTK) from Cambridge University, and Microsoft Whisper. However, the accuracy, robustness and speed of ASR systems remain the big challenge for both the industry and researchers aiming at building more-reliable and cost-effective commercial applications.

2.4.2 The Recent History of Acoustic Modeling with DNNs for Speech Recognition

Despite much progress, both researchers and industry continue working to further improve the accuracy, robustness and performance of ASR systems for real usage. They have focused their efforts on acoustic modeling since it is arguably the central part of the speech recognition engine.

For decades, Gaussian Mixture Model-Hidden Markov Models (GMM-HMMs) have been the dominant technique for acoustic modeling. In this hybrid approach, HMMs are used to model the sequential structure of the sound wave, and GMMs are used for modeling the emission distribution of HMM states.

In the late 1980s and the early 1990s, Artificial Neural Networks (ANNs) were introduced as an alternative to GMMs. ANN-HMM type hybrid models were proposed in which a feed-forward neural network was trained to estimate the posterior probabilities of continuous density HMM's states, given the acoustic features. The main goal of such hybrid architecture was to improve the robustness and the recognition performance of ASR systems by taking advantage of the properties of both ANNs and HMMs [75]. Theoretically, neural networks, in particular Multilayer Perceptrons (MLPs), seemed to be very competitive with GMMs. The advantages of neural networks were thought to be the availability of discriminative training algorithms, the efficiency of their output computation and probability estimation, and the efficiency in their hardware implementation, in addition to their needing fewer estimation parameters than GMM-HMMs usually do [75]. However, the contribution of ANNs were not as expected due to the limitation of the back-propagation learning algorithm in training networks with many hidden layers and the large output layer used to fit the numerous HMM states, on large amounts of data. With back-propagation, it was not possible to model networks with more than two hidden layers, especially with the hardware restrictions at that time. In fact, the performance benefits of using neural network were not important enough to challenge GMMs. In consequence, the hybrid model ANN-HMMs fell out of favour with both researchers and industry for many years.

In 2006, a new approach for training deep neural networks (DNNs) with many hidden layers of nonlinear units was introduced [33, 34]. This approach has proven efficient and is known as a notable success in a wide range of applications (e.g., digit recognition, 3-D object recognition, information retrieval, etc.), triggering the interest of researchers to apply deeper architecture for acoustic modeling. The early attempt to use DNNs for acoustic modeling was made by Mohamed et al. [55] in 2009 at the University of Toronto. The work consisted of using Deep Belief Networks (DBNs) [33] to model the spectral variability in speech. In fact, the network was trained to estimate the posterior probabilities over HMM states given a window of n successive frames of speech feature vectors. Unlike the former method of using ANNs, the network was much deeper and larger and trained in a new fashion. Applied to phonemes recognition, the proposed DBN-HMM model beat the best reported recognition results on the TIMIT core test set. More extensive experiments in the same direction were conducted in [56] in order to explore variations in the architecture of the neural network (number of layers and number of units per layer), and the representation of the acoustic input (number of frames of speech feature vectors). It was proved that with many hidden layers, better recognition results were reached on the TIMIT dataset [32]. Subsequent works in [57, 15, 59] have strengthened the idea that deeper model are capable of consistently outperforming GMMs at modeling the spectral variability in the speech signal for phoneme recognition tasks.

These promising results on phoneme recognition have motivated many other research groups in academia and in industry as well, such as Microsoft Research (MSR), Google, and IBM research, to exploit the DNN-HMM hybrid architecture for large-vocabulary continuous-speech recognition tasks. The first successful work was done in 2011 by Dahl et al. [13], who introduced a context-dependent DBN-HMM hybrid acoustic model (CD-DBN-HMM) that greatly outperformed GMM-HMM systems on large data collected from the Bing mobile voice search (BMVS) applications (more details about this work were reported in [14]). The network was trained to produce distribution over senones (tied triphone HMM states). Subsequent efforts have proved fruitful, and experiments on different large vocabulary speech recognition benchmark tasks such as Switchboard corpus [70], Google Voice Input [40], English Broadcast News [69]demonstrated that DNN-HMMs are capable of outperforming GMM-HMMs by a large margin [32].

Over 2013, new applications of deep learning technologies in acoustic modeling for speech recognition have been carried out and showed successful results. One of the most interesting application, is the usage of DNN acoustic models for multi-task learning [18, 19]. Being trained in data from one task, the DNNs were able to benefit in other related tasks. Actually, DNNs have shown excellent performance in two separate works: the first one was on mixed-bandwidth training where the DNN was trained with wideband (16-kHz sampling rate) and narrowband (8-kHz sampling rate) acoustic data jointly [50]. This approach proved that the mixed-band DNN can improve wideband speech recognition accuracy in the CD-DNN-HMM framework. The second work was on multilingual speech recognition [19, 27, 37]. Being trained on multiple languages simultaneously, DNNs have proved to have an excellent capability to extract acoustic features shared across all languages.

In fact, DNNs are considered as universal learners because their intermediate hidden layers play the role of high-level feature extractors, allowing them to "handle heterogeneous data from different acoustic sources and languages" [19]. Indeed, the powerful architectures and learning procedures for DNNs have allowed them to be used for multi-task learning more effectively than GMMs.

Another interesting accomplishment of DNN in acoustic modeling for speech recognition in the same year was the establishment of DNN performance for noisy speech. The work in [19] has empirically proved the noise-robustness of DNN-based acoustic models due to their capacity to extract stable representations from the variations, such as environmental noise, immanent in the acoustic data.

2.5 Conclusion

This chapter has provided a thorough overview about the process of speech recognition, the architecture, and the main components needed to effectively design such complicated systems. It also discussed the major milestones that marked the history of this field.

Fueled by many technological developments, ASR systems have known a significant progress and have invaded a wide range of real applications. Furthermore, the big advance in the area of artificial intelligence and machine learning algorithms has played a critical role in the improvement of the accuracy and robustness of ASR systems. Recently, the adoption of DNNs as an alternative paradigm to GMMs in HMM-based acoustic modeling has shown promising results in many speech recognition tasks.

Actually, DNN-HMMs are considered to be a new generation of acoustic models in ASR. The aim of this thesis is to experimentally confirm the power of DNNs in acoustic modeling for phoneme recognition. To this end, it is crucial to understand the architecture and the deep learning algorithms that have been the key success of such deep connectionist models. Thus, the next chapter is devoted to DNNs.

Chapter 3

Deep Neural Networks

A Deep Neural Network (DNN) is defined as a feed-forward Artificial Neural Network (ANN) that has multiple layers of hidden units stacked on top of each other between the input and the output layers [20, 32]. The main goal of DNNs is the discrimination between classes; however, they are often generatively pre-trained in order to initialize their parameters.

The focus in this thesis is on DNNs and their use for acoustic modeling in speech recognition. The architecture and the learning algorithm of DNNs are analysed in this chapter.

3.1 Motivations for Training DNNs

ANNs are connectionist models composed of highly connected computational units (neurons) organized in a specific manner that makes them able to perform a nonlinear transformation to the input data. In a feed-forward ANN, the units in one layer are connected to the units in the next layers using unidirectional connections as shown in Figure 3.1. In fact, each hidden unit j, in layer l, is connected to all the units in the layer $l-1$, and has an activation function, typically logistic, that is responsible for summing the weighted input from the layer below, and for processing the result to the layer above as a single output. The output of the units i can be calculated using the following equation(3.1):

$$o_i = logistic(\sum_{i}^{N} w_{ij}x_i + b_j) \tag{3.1}$$

where N is the number of units in the layer $l-1$, w_{ij} is the weight of the connection between unit j and unit i of the layer below, and b_j is the bias of the hidden unit j.

$$logistic(x) = 1/(1+e^{-x})) \tag{3.2}$$

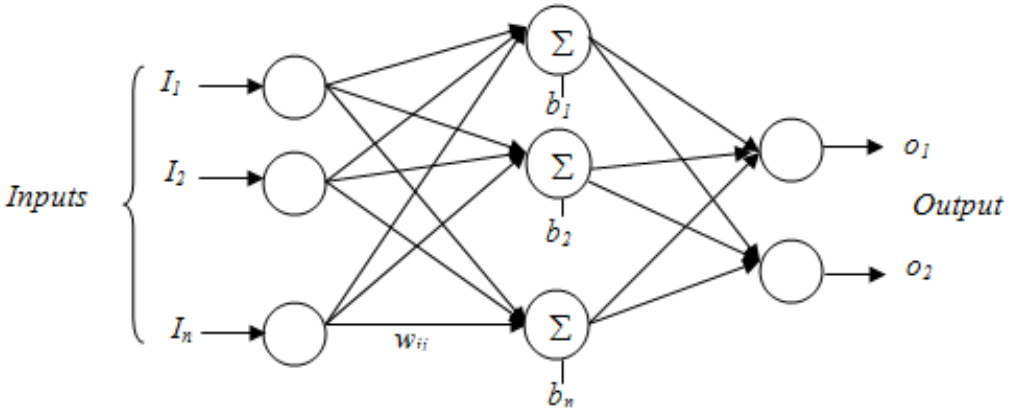

Figure 3.1: Feed-forward ANN with multiple inputs, two outputs and one hidden layer

ANNs are discriminatively trained in a supervised way using the back-propagation learning algorithm that measures the error between the network outputs and the desired outputs, and updates the network parameters (the weights and biases) by applying the gradient descent optimization technique with the aim of reducing the local error [47].

The motivation behind designing ANNs is to mimic the computational power of the human brain. In fact, ANNs have shown great success in the areas of machine learning, pattern recognition and classification. Moreover, they have proven a notable strength in modeling nonlinear systems. They have been used in a wide range of applications including robotics, medical, communication systems, data mining, sales and marketing [47]. However, only shallow-structured architectures were exploited, and researchers have failed, for many years, to train networks with more than one hidden layer.

The main difficulty that arises is the "curse of dimensionality", which states that the variations in the input data grow exponentially with a linear increase in their dimensionality [7, 39]. In consequence, the network may require a huge number of training data in order to model all these variations, and the learning process is inherently very complex and computationally very expensive.

This problem begs the need for different new approaches to train deep-layered neural networks. The clue might lie in understanding how the human brain processes this vast

amount of information, received in a very high frequency, with such an ease. It has been argued that the brain processes the sensory input through multiple layers in a hierarchical fashion. Each layer extracts new features from its inputs and creates a higher level representation to pass it on to the next higher level [7, 39].

Thus, designing deep architectures is an attempt to emulate the representational power and the information modeling ability of the human brain. Learning a DNN to extract useful features from its input data and eliminate unnecessary variations is considered the key solution to preventing the curse of dimensionality. As a result, the number of training examples required for learning high dimensional data would drastically decrease. A DNN can achieve this features extraction and transformation using hierarchical processing of the input data akin to what the brain does. The idea is to feed the input data to the first layer which learns to extract a new representation of its input and provides its output to the layer above. The next layer then derives a new higher-level representation from its inputs. The process continues all the way to the top layer, resulting in meaningful features and a new description of the input data.

Despite the big development in the area of computer hardware and machine learning algorithms, researchers have failed to train a network with many layers of nonlinear feature detectors. The back-propagation learning algorithm has been the efficient and traditional way for discriminatively training ANNs due to its ease of implementation. However, attempts to train a DNN that learns to derive features and build internal representations from complex inputs using the back-propagation algorithm have led to dismal results. In fact, training deep networks seems to be very difficult and hard to optimize; these networks perform worse than shallow ones in terms of generalization [7]. As networks become deeper, they tend to get stuck more often in poor local minima. This trend is attributed to the random initialization of the weights in the back-propagation algorithm. Moreover, the large number of layers drives the network to learn rare dependencies and variations in the input data, which makes it very prone to the problem of overfitting [20, 32, 7, 16].

Due to these generalization problems, learning deep-layered architectures for tackling very complicated applications fell out of favour amongst both researchers and industry. Hence, many experiments were made to overcome these problems and to find a good learning algorithm able to mimic the robustness of the brain in representing information, and able to produce good classification results in terms of time and accuracy.

3.2 Breakthrough of DNNs

In 2006, simultaneous and independent ground-breaking achievements in training deep models were made by Hinton et al. in [33], Bengio et al. in [8] and Le Cun et al. in [65], and have demonstrated successful results. These researchers proposed different approaches that all consist of two phases learning:

- Greedy layer-wise unsupervised pre-training, and
- Global supervised fine-tuning.

The intention behind the unsupervised pre-training is to optimize the weights, and hence the network converges toward a better final local minimum during the supervised learning. In fact, by pre-training the network layer by layer, the difficulty associated with the random initialization of the weights in the back-propagation algorithm is empirically alleviated. On the other hand, each layer learns to extract features from its inputs and to create a higher-level representation that is transferred to the next higher level. Then, the global fine tuning with the standard back-propagation algorithm is used as an optimization process to adjust the weights found during the pre-training and to make them more useful for discrimination [32, 56]. This technique has also drastically cut down the time required for the global supervised learning.

In fact, many approaches have been proposed for pre-training deep architectures, but the generative learning approach based on Deep Belief Networks (DBNs) proposed in [33] has become a standard one. Other techniques have been proposed for accomplishing the pre-training phase of DNNs and have proven the same success. For example, Bengio et al [8] proposed a pre-training technique using a non-probabilistic and non-generative model that consists of stacking layers of auto-encoder.

Currently, DNNs are booming in both the research and industry. They have been used in many applications with notable success, such as dimensionality reduction [34], image recognition and classification [33, 8], 3-D object recognition [62], information retrieval [35], and language modeling [48].
Recently, generatively pre-trained DNNs have been used for acoustic modeling in a wide variety of speech recognition and have produced successful results in particular.

3.3 Generative Pre-Training of DNNs

In contrast to discriminative learning on neural networks, which aims at modeling the nonlinear relationship between the input and the output data, the main goal of generative learning is to model the structure of the sensory input [32] and to extract salient information about the input distributions. In fact, the learning objective for generative models is to maximize the probability of the observed data, $p(data)$, and not to maximize the labels, given the inputs, $p(label/data)$.

The idea behind pre-training a deep architecture in a generative fashion is to create internal higher-level representations of the input at each layer. These higher representations contain the most useful information needed for good predictions. Indeed, the different units in the different layers play the role of nonlinear feature detectors. Furthermore, the ability to exploit the large amount of unlabeled data to initialize the weights of the neural network represents the major benefit of generative pre-training. Thus, the weights found during the pre-training serve as the starting point for the discriminative fine-tuning phase [32, 56, 74].

In fact, the unsupervised pre-training consists of training the network, layer by layer, from bottom to top where the output of one layer is the input to the next layer, in order to extract features and correlations from the observed data. Doing so allows the weights in the lower layers to move to regions that correctly represent the input [32]. Hence, the supervised learning phase using the back-propagation learning algorithm comes to adjust the weights and fine-tune the whole network, and thereby it discriminates between the different classes.

The generative pre-training can be done in two different ways: either by using directed causal models or by using undirected energy-based models [32, 56].

The directed models consist of using Sigmoid Belief Networks(SBNs) that are composed of many layers of binary stochastic latent units connected in a directed acyclic graph, as depicted in Figure 3.2 .

To generate data from this model, the binary values of the units at the top layer are determined based on their biases. Given the states of the top layer, a stochastic decision is made about the units in the layer below. This process is performed until the values of the visible layer, which represent the input, are determined from their conditional distributions, given the latent states. In fact, the binary state of the unit i is generated from the inferred binary states of its parents, with a probability p_i calculated using the following equation (3.3)

$$p_i = P(s_i = 1) = \frac{1}{1 + exp(-b_i - \sum_j s_j w_{ji})} \tag{3.3}$$

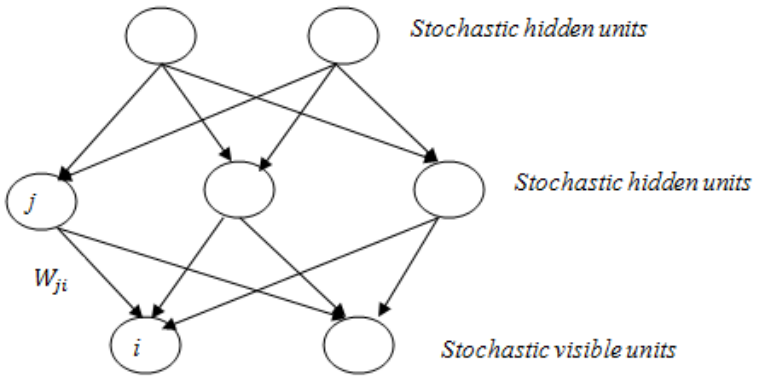

Figure 3.2: Sigmoid Belief Network composed of binary units

where p_i is the probability that the parents of unit i turn i on (i.e., assign the value 1 to i), b_i is the bias of unit i, s_j is the j^{th} element in the layer above, and w_{ji} is the weight of the connection between unit j and unit i.
The update rule for the weights is simply $\Delta(w_{ji}) = \epsilon(s_j(s_i - p_i))$, where ϵ is the learning rate.

The main problem of this model is the difficulty of getting the posterior distributions over the hidden variable configurations, given the visible states. This problem arises from the phenomenon of "Explaining away", which asserts that even if two hidden units are independent in the prior knowledge, they become conditionally dependent when they both have correlated effects on a visible variable. Moreover, the posterior distribution over the latent variables is not factorial; it also depends on the prior distribution created by the layer above, as well as on the likelihood created by the layer below. For example, in Figure 3.3, the hidden variables of the second hidden layer create a prior distribution over the first hidden layer, which will cause correlations between the hidden variables. Thus, to learn the weights, W, it is needed to know the weights in the higher layers even if the posterior distributions over the hidden configurations, given the data, are approximated [32, 56]. In addition, all possible configurations in the higher layers must be integrated to get the prior distribution for the first hidden layer.

The undirected model consists of using a network where the binary stochastic units of each layer are connected using symmetric connections. This kind of model uses an energy function to determine the joint configuration over the hidden and visible units, rather than trying to determine the prior distribution over the hidden variable, and the conditional distribution over the visible variable given the states of the hidden variables [32]. To fit

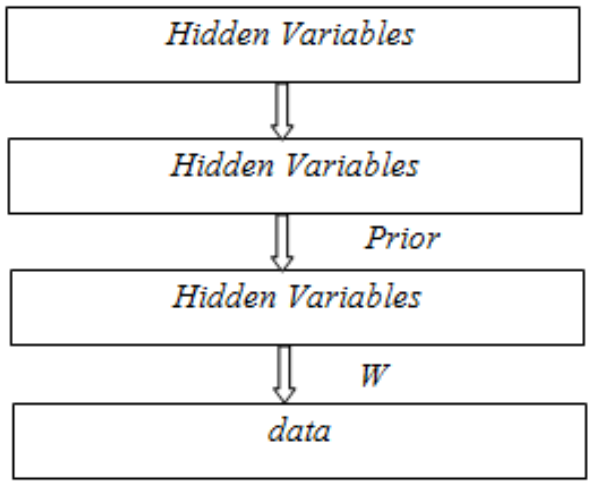

Figure 3.3: Sigmoid Belief Network with many hidden layers

the input data, this generative model uses simple sub-models learned sequentially from the bottom up. These sub-models named Restricted Boltzmann Machines (RBMs) are composed of only one layer of visible units and one layer of hidden units. To create the generative model, many RBMs are stacked on top of one other, where the output of one RBM serves as the input to another RBM, and thus a DBN is created where each hidden layer is a new representation of the input data.

Due to their efficiency as generative models, DBNs are the most useful models for pre-training DNNs. They will also be used in this thesis for building acoustic models with DNNs. The next section details the learning algorithm for DBNs.

3.4 Deep Belief Networks(DBNs) for Generative Pre-Training

DBNs are multilayer, probabilistic, nonlinear generative models, where each pair of layers is an RBM. By stacking RBMs, as many as needed, on the top of others to create a DBN, the layers represent feature detectors that create complex correlations in the input data in a progressive way. In DBNs, starting with the lowest layers, each layer is considered individually, and trained such that the weights form a suitable representation of the input.

Once a layer is trained, the weights are frozen and the outputs are used as inputs to train the next level RBM. This process is repeated until all the layers are trained. To generate data from this model, the top two hidden layers form an undirected associative memory and the lower hidden layers form a directed acyclic graph that transforms the representation held in the associative memory into observable data [33].

3.4.1 Restricted Boltzmann Machines and Contrastive Divergence Learning

An RBM is a particular type of Markov Random Field(MRF). It has one layer of stochastic binary hidden units and one layer of stochastic binary visible units. As shown in Figure 3.4, an RBM is represented as a bipartite graph in which the visible units are connected to the hidden units using undirected weighted connections. The visible units represent the observed values (i.e., the input data), whereas the hidden units represent the feature detectors that learn to extract features from the input data. In RBMs, there are no visible-visible or hidden-hidden connections.

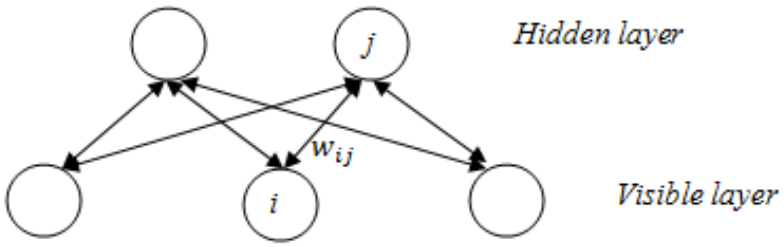

Figure 3.4: Illustration of an RBM

An RBM is an energy-based model. It uses an energy function E in order to assign a probability to every possible configuration of visible and hidden state vectors [31], according to the following equation (3.4)

$$p(v,h) = \frac{1}{Z}e^{-E(v,h)} \tag{3.4}$$

Z is a partition function that ensures the valid distribution of the probability, and that

the sum of the probabilities of all possible pairs (v, h) is equal to 1,

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{3.5}$$

When both the hidden and visible units are binary, the result is a Bernoulli-Bernoulli RBM, and the energy of the joint configuration over the visible and hidden units is given by the following equation (3.6)

$$E(v, h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{i,j} \tag{3.6}$$

where v_i is the binary state of the visible unit i, h_j is the binary states of the hidden unit j, a_i is the bias of the unit i, b_j is the bias of the unit j, and $w_{i,j}$ is the symmetric weight between the two units.
The probability that an RBM assigns to an input vector v (visible vector) is the sum over all possible hidden vectors [31] and is given by equation (3.7)

$$p(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{3.7}$$

Considered as a generative model, an RBM can be trained using an unsupervised learning algorithm to accurately reconstruct the input patterns. The objective function is to maximize the probability that an RBM assigns to the training input data. In fact, this probability can be raised by adjusting the weights and biases to lower the energy of that observed input and to raise the energy of the reconstructed [31]. The derivative of the log probability of a training vector with respect to a weight can be simply determined as:

$$\frac{\partial log p(v)}{\partial w_{i,j}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{3.8}$$

where $\langle v_i h_j \rangle_{data}$ is the expectation observed in the training set. It measures the frequency that the visible unit v_i and the hidden unit h_j are on together, and $\langle v_i h_j \rangle_{model}$ is that same expectation under the distribution defined by the model.
Thus, to perform the stochastic steepest ascent in the log probability of the training data, the updating rule for the weights is determined by:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \tag{3.9}$$

where ϵ is the learning rate.

Since there are no direct connections within the hidden units, given a random training vector v, the binary state h_j of each hidden unit j is set to one with a probability of:

$$p(h_j = 1|v) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{3.10}$$

where σ is a logistic sigmoid function $\sigma(x) = 1/(1 + exp(-x))$. This results in an unbiased sample $\langle v_i h_j \rangle_{data}$ when the data are clamped on the visible units, and the states of the hidden units are sampled from their conditional distribution.

The fact that there are no direct connections within the visible units also results in an unbiased sample $v_i h_j$. Given a hidden vector h, the state v_i of each visible unit i is set to one with a probability of:

$$p(v_i = 1|h) = \sigma(a_i + \sum_j h_j w_{ij}) \tag{3.11}$$

Indeed, the visible units are conditionally independent given the hidden unit states and vice versa.

To get an unbiased sample of $\langle v_i h_j \rangle_{model}$, the states of the visible units are set to random values, and alternating Gibbs sampling is performed until the network reaches equilibrium, which can take a very long time. This process consists of updating all the hidden units in parallel using equation (3.10), followed by updating all the visible units in parallel using equation (3.11) [31].

In order to overcome the difficulty of getting an unbiased sample of $\langle v_i h_j \rangle_{model}$, Hinton proposed in [30] the Contrastive Divergence(CD) learning procedure. In this procedure, the visible units are clamped on a given training vector, and the states of all the hidden units are stochastically updated in parallel given the current states of the visible units (positive phase). Then the input data is reconstructed by stochastically updating all the visible units in parallel given the current states of the hidden units (negative phase).
Running a Markov chain using alternating Gibbs sampling for n full steps (CD_n), as shown in Figure 3.5, where n is a very large number, results in determining the statics for the second term in the learning rule $\langle v_i h_j \rangle_{model}$.

The pseudo code in Figure 3.6, reported in [74], highlights the learning algorithm of one RBM layer using the contrastive divergence with k iterations. H_D and V_D represent, respectively, hidden and visible units when the network is clamped on input data, H_M

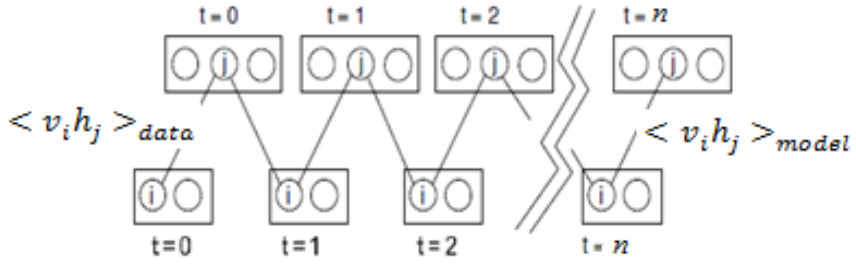

Figure 3.5: Illustration of the Contrastive Divergence learning procedure for RBMs [33].

and V_M represent the models reconstructions, and W is the weight matrix. The stochastic binarization of a vector is performed using the *stoch* function, while *sigm* indicates the logistic function.

```
# positive phase
H_D = sigm(V_D * W)

# negative phase
H_M = stoch(H_D)
for i = 1 to k:
    V_M = sigm(H_M * W^T)
    V_M = stoch(V_M)
    H_M = sigm(V_M * W)
    H_M = stoch(H_M)
ΔW = η(H_D V_D^T - H_M V_M^T)
```

Figure 3.6: A pseudo code for training an RBM with K iterations CD [74]

However, it has been proved that CD_1, which consists on one single full step of alternating Gibbs sampling after the initial update of the hidden units, is sufficient [32, 30]. The updating rule for the weights is then given by:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{CD_1}) \tag{3.12}$$

To model real-valued input data, such as MFCCs to represent speech, the visible units in an RBM are replaced by linear units with Gaussian noise, and form a Gaussian-Bernoulli RBM (GRBM). The energy of the joint configuration of visible and hidden units is determined as:

$$E(v,h) = \sum_{i \in visible} \frac{(v_i - h_j)^2}{2\sigma_i^2} - \sum_{j \in hidden} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{i,j} \tag{3.13}$$

where σ_i is the standard deviation of the visible unit i. The two conditional distributions $p(h_j|v)$ and $p(v_i|h)$ are factorials and can be calculated using equation (3.14) and equation (3.15) respectively.

$$p(h_j|v) = \sigma(b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij}) \tag{3.14}$$

$$p(v_i|h) = N(a_i + \sigma_i \sum_j w_{ij} h_j + b_i, \sigma_i^2) \tag{3.15}$$

where $N(\mu, \sigma^2)$ is a Gaussian with mean μ and variance σ^2.

3.4.2 Learning DBNs by Stacking RBMs

Combining many RBMs to make a DBN starts with clamping the input data on the visible layer of the first RBM. After training with contrastive divergence, the states of the units in the hidden layer are copied and used as input for training the second RBM. This step is repeated as many times as there are RBMs in the stack. Then all the RBMs are combined such that only the top two layers that form the last RBM in the stack have undirected connections; the rest of the lower layers have top-down directed connections just as in a Sigmoid Belief Network. Thus, the DBN created by the whole stack is considered as a hybrid generative model, as shown in Figure 3.7.

By applying equation (3.7) and inversing the roles of the visible and hidden variables, the set of weights, W, define both the $p(v|h, W)$ and the prior distribution over the hidden vectors, $p(h|W)$, [32, 55]. Thus, the RBM model can be expressed as the probability of generating a visible vector v as following:

$$p(v) = \sum_h p(h|W) p(v|h, W) \tag{3.16}$$

This model requires improvement of $p(h|W)$, while $p(v|h, W)$ is kept fixed, by replacing the former with a better model that defines a better prior that fits the aggregated posterior

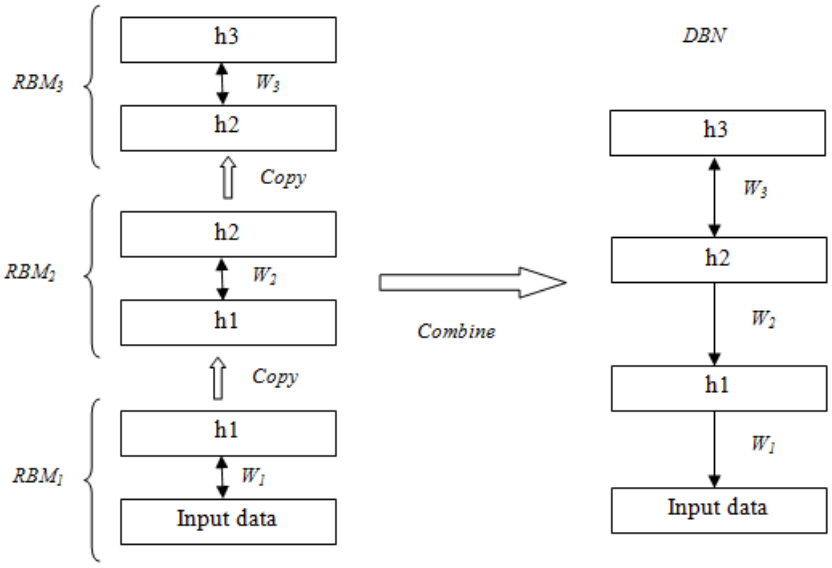

Figure 3.7: Creating a DBN by stacking three RBMs

over the hidden variables [32, 55]. This model is obtained by training another RBM using the states of the hidden units of the first RBM as inputs. In fact, by clamping the data on the visible units, the bottom level RBM produces the aggregated posterior distribution over the hidden vectors, and the second level RBM is used to build a better model of this aggregated posterior than the first RBM.

3.5 Discriminative Training of DNNs

After training a DBN as stacked RBMs, the weights are used as a starting point for learning a DNN. In fact, the generative weights are reversed into bottom-up direction, and a Softmax output layer that represents the labels is added on the top of the DBN. Hence, the whole network forms a deterministic feed-forward neural network. This network is then discriminatively fine tuned using the back-propagation learning algorithm, as depicted in Figure 3.8.

Each unit in the output layer is used for prediction and can be used to estimate the

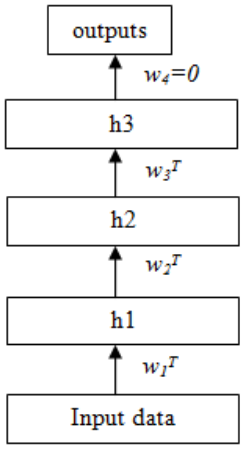

Figure 3.8: Creating a DNN from a DBN by adding a Softmax output layer on the top

probability that an input x is assigned to label l_i, derived as :

$$p_i = P(l_i|x) = \frac{e^{(b_{l_i} \sum_j h_j^k w_{jl_i})}}{\sum_i^L e^{(b_i + \sum_j h_j^k w_{ji})}} \tag{3.17}$$

where b_{l_i} is the bias of the label l_i, k is the last hidden layer in the DNN, h_j^k is the state of the unit i at layer k, w_{jl_i} is the weight of the connection between unit i at label l_i and unit j, and L is the total number of labels.
Hence, the objective function of the supervised learning is to minimize the error between the desired output and the network output.
Finally, a DNN that is generatively pre-trained as a DBN is called a DNN-DBN [32].

3.6 Conclusion

Recently, the establishment of deep learning technique for training ANNs with many hidden layers represents a paradigm shift in the area of machine learning and artificial intelligence. This technique consists of two-phase procedure. The network is first generatively pre-trained; then, it is discriminatively fine-tuned using the back-propagation learning algorithm.

Training DNN intends to create new representation of the input data at each layer. In fact, the hidden neurons play the role of feature detectors. They intend to extract salient information from the data. The representation learning in DNN is carried out using generative learning algorithm, which trains the network in an unsupervised way without any knowledge about the data labels. This approach has proven efficient in dealing with the curse of dimensionality problem.

Different techniques have been proposed for generatively pre-training DNNs. However, in this thesis, the technique based on DBNs is studied. DBNs are probabilistic generative models that are created by stacking RBMs. In fact, each pair of layer in DBN represents an RBM that is trained using the contrastive divergence algorithm; the output of one RBM is the input to the next higher RBM. Once the DBN is trained, an output layer is added on the top to create a DNN. This DNN is then discriminatively trained using the back-propagation learning algorithm. The generative pre-training phase has proven efficient in initializing the weights helping the network to rapidly converge during the discriminative phase.

In this chapter, the architecture of DNNs and the learning algorithm are studied. In recent years, DNNs have been used to replace GMMs in HMM-based acoustic models. The next chapter is devoted to DNN-HMM hybrid architecture for acoustic modeling.

Chapter 4

DNN-HMM for Acoustic Modeling

Acoustic modeling in speech recognition refers to the process of creating statistical representations for the feature vector sequences computed from the speech waveform. As discussed in chapter one, these acoustic models should be robust and capable of modeling not only the variability intrinsic in human speech but also the noise that characterises the background environment. In fact, the acoustic models should determine the probability $P(A|W)$ of an acoustic vector sequence A, given that the uttered word sequence is W. Due to the large number of different possible pairings of W with A, sophisticated statistical models are needed [41]. HMMs are typically the most useful acoustic models in speech recognition.

In ASR, HMMs are typically used to model the sequential structure of speech signals [55, 58], with Gaussian densities at each state in order to model the local spectral variability in the sound wave [55]. Using the Expectation Maximization (EM) algorithm, GMMs are easy to fit to data. Indeed, they have proved effective in modeling the relationship between HMM states and the acoustic input represented more often as a set of MFCCs. With enough components, GMMs can accurately model the probability distribution over the acoustic input feature vectors associated with each state of an HMM [32]. Nonetheless, GMMs in HMM-based acoustic models suffer from some major limitations that have motivated researchers to find other models.

This chapter addresses the major issues in using GMM-HMMs for acoustic modeling. It describes the new DNN-HMM hybrid architecture, and discusses the main advantages of this architecture for acoustic modeling in speech recognition.

The first section introduces HMMs parameters and the associated learning and decoding algorithms. In fact, this chapter does not intend to present an in-depth description

of HMM theory or to widely cover its application to speech recognition, but only to give the preliminary body of knowledge necessary to later understand how interfacing DNNs with HMMs. In the second section, the limitations of GMM-HMM-based speech recognition systems are tackled, and especially the shortcomings of GMMs that have led many researchers to use neural networks with the hope of alleviating some of them. Then, the DNN-HMM hybrid architecture for acoustic modeling is discussed, as well as the role and the advantages of the DNNs in ASR systems.

4.1 HMMs for Acoustic Modeling in Speech Recognition

Current state-of-the-art speech recognition systems are mainly based on HMMs for acoustic modeling. HMMs are the most efficient approach developed so far to dealing with both the temporal and spectral variability of speech signals. Essentially, HMMs are probabilistic generative models presented under the form of a stochastic finite state machine in which each state has an output probability distribution that describes the probability of occurrence of certain feature vectors.

Although the speech signal is defined as a non-stationary process, it can be modeled as a stationary process under the assumption that over a short period of time, the statistical characteristics of the signal do not differ from sample to sample. The speech pre-processing technique consists of extracting salient features from the sampled speech waveform at regular small intervals that are generally spread between 8 and 16 msec [11]. Then the speech signal associated with each utterance is represented as a sequence of acoustic feature vectors $X = \{x_1, x_2, ..., x_N\}$. These feature vector sequences are considered to be piecewise stationary processes, and modeled using the HMMs as a sequence of stationary states with instantaneous transitions between these states. Thus, HMMs involves two different and concurrent stochastic processes [11]: the first represents the state sequence and aims at modeling the temporal structure of the speech, and the second is related to the stochastic output process associated with each state, and aims at modeling the local spectral characteristics of the speech signal. In HMM, the state sequence is not directly observable, but it affects the observed sequence of events since each state contains salient information about the modeled data, and hence the HMM is called "hidden".

Generally, for a small-vocabulary isolated speech recognition task, there should be one HMM for each word (or sentence for a continuous recognition task) allowed in the vocabulary. However, for large vocabulary, this is not feasible, since it would require a

large number of models with many parameters, and a big amount of training data that is difficult to collect. In addition, as soon as a new word is introduced in the vocabulary, new training is required. In consequence, it is more efficient to use sub-words, such as phones, di-phones, demi-syllables, and syllables,as speech recognition units. Phonemes are the most common units, and are used in this work as basic speech recognition units for investigating the performance of DNNs in acoustic modeling. However, to model a sequence of speech units, an HMM can be built by concatenating individual trained HMMs associated with the elementary speech units. For example, considering phonemes as the basic units, words can be modeled by concatenating phone-based HMMs, but a challenge exists in dealing with pronunciation variations.

In HMM-based systems, the speech feature vector sequences are assumed to be generated from a finite state automate according to the output probability function associated with each state. In fact, an HMM used to model a speech unit is composed of N distinct emitting states labelled by $\{1, ..., N\}$, with the state at time t denoted as s_t. The states in the HMM are introduced in a specific topology; the left-to-right is the standard one for modeling the speech signal.

Fundamentally, an HMM is defined by the set of parameters $\Phi = (A, B, \pi)$, where:

- $A = \{a_{ij}\}$ represents the transition probability matrix; a_{ij} is the probability of taking a transition from state i to state j, $P(s_t = j|s_{t-1} = i)$, and should satisfy the condition $\sum_{j=1}^{N} a_{ij} = 1 \forall i$.
- $B = \{b_j(x)\}$ represents the output probability matrix. For continuous speech feature vector such as MFCCs, $b_j(x)$ defines the probability $p(x|s_t = j)$ of emitting the feature vector x when the state j is entered, and should satisfy this condition: $\sum_{k=1}^{M} b_j(k) = 1 \forall j$.
  Multivariate Gaussian Mixture Models(GMMs) are often used to estimate this probability. Hence, with M Gaussian mixture density functions:

$$b_j(x) = \sum_{k=1}^{M} c_{jk} N(x, \mu_{jk}, \Sigma_{jk}) \tag{4.1}$$

$$b_j(x) = \sum_{k=1}^{M} c_{jk} b_{jk}(x) \tag{4.2}$$

  where both $N(x, \mu_{jk}, \Sigma_{jk})$ and $b_{jk}(x)$ define a single Gaussian density function with mean vector μ_{jk} and covariance matrix Σ_{jk} for state j; M is the number of mixture-components, and c_{jk} is the weight for the k^{th} mixture component, which should satisfy the condition $\sum_{k=1}^{M} c_{jk} = 1$.

- $\pi = \{\pi_i\}$ determines the initial state distribution where $\pi_i = P(s_0 = i)$, $1 \leq i \leq N$.

To model the acoustic characteristic of the vocabulary, each single speech unit is modeled with a separate HMM, resulting in a set $M = \{M_1, M_2, ..., M_U\}$ of possible elementary speech unit HMMs with the set of associated parameters $\Phi = \{\Phi_1, \Phi_2, ..., \Phi_U\}$. That is, the goal of the acoustic modeling is to estimate $P(X|M)$, the probability that the model M has generated the acoustic vector sequence X. This estimation is mainly performed in the parameter space of the probability of the likelihood and is known as the Maximum Likelihood Estimation(MLE). Hereinafter, each model M is referred to by its associated parameter Φ.

Given the above definition, three major problems should be addressed when using HMMs for acoustic modeling in speech recognition:

- **The probability estimation (or the evaluation) problem:** The determination of the probability $P(X|\Phi)$, given the model Φ and a sequence of observed feature vectors $X = (X_1, X_2, ..., X_T)$.
- **The decoding problem:** The determination of the most likely state sequence $S = (S_1, S_2, ..., S_T)$ that generates the observed sequence of feature vectors $X = (X_1, X_2, ..., X_T)$, given a model Φ.
- **The training problem:** Tuning the model parameters $\hat{\Phi}$ in order to maximize the joint probability(likelihood) $\prod_X P(X|\Phi)$, given a model Φ, and a set of observation sequences.

By solving the estimation problem, it becomes possible to evaluate how well the HMM associated with a particular speech unit matches a given acoustic observation sequence. Since it is possible to compute the posterior probability $P(\Phi|X)$ using the likelihood $P(X|\Phi)$, the HMM can be used for pattern recognition. The desired pattern is identified by the HMM that has the highest posterior probability [38]. Solving the decoding problem reveals the hidden state sequence that best fits the observed sequence. Finally, by solving the learning problem, the model parameter Φ can automatically be estimated from the set of training data.

To resolve these problems, two main assumptions with HMMs have been made. First, it is assumed that the HMMs are first-order Markov models; hence, the probability that the Markov chain is in a particular S_t at time t depends only on the state S_{t-1} of the Markov chain at the previous time $t - 1$, and is conditionally independent of the past.

Second, it is assumed that the acoustic observations are conditionally independent. In another word, the probability of emitting a particular vector at time t depends only on the transition taken at that time. Despite of the unrealistic property of these assumptions, tractable-decoding and learning can be effectively performed even with large amount of speech data [55].

Next, the solutions for each problem are discussed.

4.1.1 The Forward Algorithm to Evaluate an HMM

Evaluating an HMM model Φ implies the computation of the probability $P(X|\Phi)$ of an observation sequence $X = (X_1, X_2, , X_T)$, given the model. The most straightforward way is to enumerate all the possible state sequences of length T, and then sum their output probabilities.

For a particular state sequence $S = (S_1, S_2, , S_T)$, where S_1 is the initial state, the output probability of the associated observation sequence X is calculated by applying the independent assumption as follows [38]:

$$P(X|S, \Phi) = P(X_1^T|S_1^T, \Phi) = \prod_{t=1}^{T} P(X_t|S_t, \Phi) \tag{4.3}$$

$$P(X|S, \Phi) = b_{S_1}(X_1).b_{S_2}(X_2)...b_{S_T}(X_T) \tag{4.4}$$

The probability of the state sequence S is calculated by applying the first-order Markov chain assumption, as follows:

$$P(S|\Phi) = P(s_1|\Phi) \prod_{t=2}^{T} P(S_t|S_{t-1}, \Phi) \tag{4.5}$$

Hence, the probability of the observation sequence given the model, $P(X|\Phi)$, is calculated by summing the joint probabilities over all possible state sequences, as follows:

$$P(X|\Phi) = \sum_{allS} a_{s_0 s_1} b_{s_1}(X_1) a_{s_1 s_2} b_{s_2}(X_2)...a_{S_{T-1} S_T} b_{S_T}(X_T) \tag{4.6}$$

However, the enumeration of all possible state sequences is not feasible, due to the exponential computational complexity of $O(N^T)$. Hence, the forward algorithm is an

efficient solution to this problem. This algorithm stores intermediate results about partial state-sequence, and uses them for the subsequent-sequence calculations [38]. For this, it computes a forward probability $\alpha_t(i)$ defined in equation (4.7):

$$\alpha_t(i) = P(X_1^T, S_t = i|\Phi) \tag{4.7}$$

$\alpha_t(i)$ is also defined as the probability of generating the partial observation sequence $X = (X_1, X_2, ..., X_t)$ when the HMM is in state i at time t. This probability can be calculated inductively as follows [38]:

1. Initialisation:
   $\alpha_1(i) = \pi_i b_i(X_1), 1 \leq i \leq N$
2. Induction:
   $\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(X_t), 2 \leq t \leq T; 1 \leq j \leq N$
3. Termination:
   $P(X|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$
   when the final state is the end $P(X|\Phi) = \alpha_T(S_F)$

The computation of the forward probability should be performed in a time-synchronous approach, from left to right of the HMM. Before proceeding to time $t + 1$, all state-computations at time t should be completely done.

4.1.2 The Viterbi Algorithm for Decoding an HMM

Using the forward algorithm described above, the determination of the HMM model that best fits a given observation sequence can be done by summing up the probabilities of all possible state-paths. However, in speech recognition, it is important to determine the optimal state sequence associated with an acoustic observation $X = (X_1, X_2, ..., X_T)$. The most efficient way is to find the state sequence $S = (s_1, s_2, ..., s_T)$ that maximizes the probability $P(S, X|\Phi)$. A formal technique based on dynamic programming, called Viterbi algorithm, can be efficiently used to find the best state sequence for an HMM. Unlike the Forward algorithm that sums up probabilities from different paths coming to the same destination state, Viterbi algorithm uses and remembers only the best path. Hence, it defines the variable $V_t(i)$:

$$V_t(i) = P(X_1^t, S_1^{t-1}, S_t = i|\Phi) \tag{4.8}$$

$V_t(i)$ determines the probability of the most likely state sequence at time t, which accounts for the first t observations X_1^t , and ends in state i. The inductive procedure of the Viterbi algorithm is described as follows [38]:

1. Initialization:
   $V_1(i) = \pi_i b_i(X_1); 1 \leq i \leq N$
   $B_i(i) = 0$

2. Induction:
   $V_t(j) = Max_{1 \leq i \leq N} \left[V_{t-1}(i) a_{ij}\right] b_j(X_t)$; $2 \leq t \leq T; 1 \leq j \leq N$
   $B_t(j) = ArgMax_{1 \leq i \leq N} \left[V_{t-1}(i) a_{ij}\right]$; $2 \leq t \leq T; 1 \leq j \leq N$

3. Termination:
   The best score is equal to : $Max_{1 \leq i \leq N} \left[V_t(i)\right]$
   $s_t^* = Argmax_{1 \leq i \leq N} \left[B_T(i)\right]$

4. Backtracking:
   $s_t^* = B_{t+1}(s_{t+1}^*)$; $t = T-1, T-2, ..., 1$
   $S^* = (s_1^* s_2^*, ..., s_T^*)$ is the best sequence.

The computation in the Viterbi algorithm is also performed in a time-synchronous fashion from left to right of the HMM.

4.1.3 The Baum-Welch Algorithm for Estimating the HMM Parameters

The model parameter $\Phi = (A, B, \pi)$ of HMM should be adjusted in order to maximize the probability of an observation sequence given the model. Due to the absence of any known analytical method that maximizes the joint probability of the training data, this problem is by far the most difficult of the three problems in HMMs. However, the parameter Φ can be tuned by locally maximizing the probability $P(X|\Phi)$ using the iterative Baum-Welch algorithm that is based on the EM method; also known as the forward-backward algorithm.

The Baum-Welch algorithm requires the definition of the backward variable $\beta_t(i)$ in a similar manner to the forward algorithm:

$$\beta_t(i) = P(X_{t+1}^T | s_t = i, \Phi) \tag{4.9}$$

$\beta_t(i)$ determines the probability of a partial observation sequence, X_{t+1}^T, from $t+1$ to the end, given that the HMM is in state i at time t. The inductive algorithm for calculating $\beta_t(i)$ can be described as follows [38]:

1. Initialization:
   $\beta_T(i) = 1/N \; ; \; 1 \leq i \leq N$

2. Induction:
   $\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)\right] \; ; \; t = T-1...1 \; ; \; 1 \leq i \leq N$

In fact, the computation of $\beta_t(i)$ should be performed recursively from right to left.

In order to iteratively update and improve the parameters of the HMM, $\gamma_t(i,j)$ is defined as the probability of transition from state i to state j at time t, given the model and the observation sequence.

$$P(s_{t-1} = i, s_t = j | X_1^T,) \tag{4.10}$$

$$\gamma_t(i,j) = \frac{P(s_{t-1} = i, s_t = j, X_1^T | \Phi)}{P(X_1^T | \Phi)} \tag{4.11}$$

$$\gamma_t(i,j) = \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)} \tag{4.12}$$

The set of HMM parameters $\Phi = (A, B, \pi)$ is adjusted iteratively according to the EM method. The re-estimated model derived from the model Φ is defined as $\hat{\Phi}$. Thus, the parameter re-estimation is computed directly by maximizing the auxiliary function:

$$\varrho(\Phi, \hat{\Phi}) = \sum_{allS} \frac{P(X, S|\Phi)}{P(X|\Phi)} log P(X, S|\hat{\Phi}) \tag{4.13}$$

where

$$P(X, S|\Phi) = \prod_{t=1}^{T} a_{s_{t-1} s_t} b_{s_t}(X_t) \tag{4.14}$$

and

$$P(X, S|\Phi) = \sum_{t=1}^{T} log a_{s_{t-1} s_t} + \sum_{t=1}^{T} log b_{s_t}(X_t) \tag{4.15}$$

The estimate of the parameter $\hat{a}_{ij}$ are computed as follows:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t(i,k)} \tag{4.16}$$

where $\hat{a}_{ij}$ is the expected number of transition from state i to state j normalized by the expected number of transition from state i.

For continuous-density HMM, the output distribution at each state is derived from Gaussian Mixture Model (GMM) with K components. Hence, the re-estimation of the mean, covariance, and mixture weight are computed as follows:

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \tau(j,k) x_t}{\sum_{t=1}^{T} \tau(j,k)} \tag{4.17}$$

$$\hat{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \tau(j,k)(x_t - \hat{\mu}_{jk})(x_t - \hat{\mu}_{jk})^t}{\sum_{t=1}^{T} \tau(j,k)} \tag{4.18}$$

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \tau(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \tau(j,k)} \tag{4.19}$$

where $\tau(j,k)$ is defined as

$$\tau(j,k) = \frac{P(X, s_t = j, k_t = k|\Phi)}{P(x|\Phi)} \tag{4.20}$$

Actually, the re-estimation formula for $\hat{c}_{jk}$ can be defined as the ratio between the expected number of times the system is in state j using the k^{th} mixture component, and the expected number of times the system is in state j. Similarly, the re-estimation formula for the mean vector $\hat{\mu}_{jk}$ defines the expected value of the portion of the observation vector associated to the k^{th} component. It is computed by weighting the value of the expected number of times the system is in state j by the observation. The re-estimation formula of the covariance matrix $\hat{\Sigma}_{jk}$ can be explained is a similar way.

Finally, similarly to the EM algorithm, the Baum-Welch algorithm for estimating the HMM parameters is described as follows [38]:

1. Initialisation: choose an initial estimation Φ.

2. Estimation-step : Compute auxiliary function $\varrho(\Phi, \hat{\Phi})$ based on Φ.

3. Maximization-step: Compute $\hat{\Phi}$ according to the re-estimation of the parameters $\hat{a}_{ij}$ and $\hat{b}_j(k)$ to maximize the auxiliary function ϱ.

4. Iteration: set $\Phi = \hat{\Phi}$ and repeat from step 2 until convergence

4.2 Limitations of GMMs in GMM-HMM Based Acoustic Models

In GMM-HMM acoustic models, GMMs are used to model the relationship between the states of the HMM and the acoustic input. These models have proven efficient in dealing with the acoustic variations related to speaker accents, pronunciation variations, and environmental noise, etc. In fact, due to these variations, modeling the state densities of HMM using a mixture of Gaussians is more accurate than using a single form of density function. In addition, the availability and the efficiency of the EM algorithm for estimating the parameters of the model have played an important role in the success of GMMs in HMM-based acoustic models. Hence, it was difficult to find a new method that can outperform GMMs.

Despite their efficiency, GMMs do suffer from several shortcomings that should be addressed. In [68], three of the major GMM problems are identified. First, GMMs assume that the data distribution is necessarily Gaussian. Second, the parameters of GMMs at each HMM state are not trained using the whole data of all states but with the subset associated with that state given the alignment. On the other hand, the number of GMM parameters needed to be estimated across all states is very big, especially in context-dependent acoustic models, and may require a large amount of training data [54]. Finally, techniques used for feature dimensionality reduction may significantly reduce the accuracy in estimating the GMM parameters due to the potential elimination of some useful information.

Another critical shortcoming of GMMs in acoustic modeling is that they may require a large number of diagonal Gaussians or full-covariance Gaussians to model highly nonlinear data; whereas, other models exist and can fit such kind of data with only a few parameters [32]. In [56], it was argued that GMMs are also statistically inefficient at modeling high-dimensional data with componential structure. This inefficiency was attributed to the fact that for two significantly different sub-bands of independent patterns, when the first contains N different patterns and the second contains M different patterns, a GMM requires $N.M$ components to model such data. In fact, each data has only one single latent cause, and hence each component must fit both sub-bands. However, only $N + M$ components are necessary to explain such data for a model that uses multiple causes, in which

each component is specific for a single sub-band. This shortcoming of GMM may affect the efficiency of GMM-HMM based ASR system where a large number of Gaussians at each HMM state must be estimated from a sub-set of the data derived from the alignment. Furthermore, in GMMs, every single Gaussian aims at modeling a partition from the input space. Having a large number of Gaussians with independent means may lead to local generalization [14].

Many approaches have been proposed to overcome some of the limitations of GMMs. Since GMMs are typically trained as generative models using the EM algorithm, applying a subsequent stage of discriminative training was a first attempt to significantly improve the GMM-HMM acoustic models. The objective function of the discriminative training has a close relationship with the main goal of the ASR system. Maximum Mutual Information Estimation (MMIE) is one of the most common discriminate estimation methods. It aims at maximizing the separation between acoustic models by taking into account not only the likelihood of the training word strings given the labels, but also the probability of other possible word string hypotheses [80].

Using feed-forward ANNs to replace GMMs in continuous density HMM for acoustic modeling was an alternative means to address the problem of GMMs discussed above. However, using only a one- or two-layer network trained with back-propagation learning algorithm provided limited results that were not sufficient to challenge GMMs. The introduction of an effective new procedure for learning deep neural networks has motivated researchers to apply deeper architecture for acoustic modeling.

On the other hand, it has been asserted that [32] the information embedded in speech can be represented with lower-dimensional data. However, GMMs are incapable of exploring highly correlated features. In consequence, they cannot handle the latent information from a large window of frames. For this purpose, they often require feature extraction techniques that may remove some information useful for speech recognition. In light of the representational power of DNNs, it is useful to investigate them as a potential alternative to GMMs. Using the generative pre-training, the intermediate layers of the deep network play the role of feature extractors.
The next section details the advantages of DNNs over GMMs and the implementation procedure of DNN-HMM hybrid architecture for acoustic modeling.

4.3 DNN-HMM Hybrid Architecture for Acoustic Modeling

Early attempts to use feed-forward ANNs as a substitute for GMMs in HMM-based acoustic models offered several advantages over GMMs. In [56], three major ones were reported. First, in contrast to GMMs, ANNs do not require detailed assumptions about the data distribution in order to estimate the posterior probabilities over HMM states. Second, when using ANNs, it is possible to apply different types of data, including the combination of discrete and continuous features. Finally, yet importantly, ANNs make use of all the training data to model their parameters. Furthermore, ANNs have shown great capabilities at modeling highly nonlinear data.

The effectiveness of the new deep learning algorithms developed recently has strengthened the idea that neural networks are suitable for acoustic modeling. As reviewed in the previous chapter, the algorithm for training deep neural networks consists of a two-steps procedure: first, a generative pre-training step that aims at initializing the weights of the network and extracting higher new representations of the input at each layer, followed by a discriminative fine-tuning step with the back-propagation learning algorithm. DBNs have been the most common technique for generatively pre-training deep networks where each layer extracts a new higher representation and new structures from the input. This pre-training stage has the great advantage of reducing over fitting and reducing the computation cost during the discriminative fine-tuning with the back-propagation algorithm, a cost that was considered the major impediment with deep networks. It also helps the network to rapidly converge towards better local minima. In earlier literature, DNNs generatively pre-trained as DBNs were often called DBNs. In order to eliminate any ambiguity, Hinton et al. introduced the new name DBN-DNNs in [32].

In combining pre-trained DNNs with HMMs within a single hybrid architecture for acoustic modeling, researchers intended to combine the representational power of DBN-DNNs and the sequential modeling capability of HMMs.

Actually, there are few techniques available for using DNN as an alternative to GMM for acoustic modeling. The same technique has been preserved so far, with some minor variations. Recently, the three pioneer groups in this field (University of Toronto, Microsoft Research, and IBM) admitted, in [18], that the approach was initially based on intuitive guesses, and the aim was to outperform GMMs in any reasonable way. In fact, the first attempts were surprisingly successful, and people were not able to conduct in-depth investigations due to the slow training even with highly powerful hardware such as Graphics Processing Units (GPUs). However, since 2013, many studies have been carried out and

have revealed some factors behind the success of this approach.

The following subsection describes the standard technique for applying DNNs to acoustic modeling, and sheds light on its advantages.

4.3.1 Interfacing DNNs with HMMs

To interface DNN with HMM for modeling acoustic data, the network is trained to estimate the probability distribution over the states of the HMM. Considered as static classifiers with fixed dimensionality input vectors, DNNs are unable to perform sequence recognition with variable dimensionality of the inputs and outputs such as speech recognition [16]. However, HMMs are the most powerful tool that can handle sequential patterns using dynamic programming. Thus, combining HMMs and DNNs fruitfully takes advantage of both static and sequential pattern recognition, which makes such hybrid architecture very useful for speech recognition.

When first used for phone recognition in [55, 56], DNNs were trained to predict the posterior probability over mono-phone HMM states. The training data consist of a window of n successive frames of speech coefficient. A network composed of many layers of nonlinear units was first generatively pre-trained using RBMs and contrastive divergence to extract new features of the input at each layer. Then, the network was discriminatively trained with the back-propagation algorithm to predict the label of the central frame. In fact, the output layer that represents the states of the HMMs provides a probability distribution over the possible labels of the central frame. Figure 4.1 illustrates the architecture of DNN for phone recognition.

As described in the first section of this chapter, an HMM is defined by 3 parameters:

- the initial state distribution $\pi_i = P(s_0 = i)$,
- the transition probability $a_{ij} = P(s_t = j|s_{t-1} = i)$, which is the probability of taking a transition from state i to state j at time t,
- the emission probability $b_j(x_t) = P(x_t|s_t = j)$ defined as the probability that the state j generates the acoustic observation x_t at time t. In GMM-HMM, this emission probability is estimated using the GMMs.

In DNN-HMM, DNN is used instead of GMM to estimate the emission probability at each state. In phoneme recognition, the DNN produces the posterior probability, $P(s_t|x_t)$,

of the mono-phone HMM states given the acoustic observation x_t. However, $p(x_t|s_t) = p(s_t|x_t).p(x_t)/p(s_t)$. The probability $p(s_t)$ can be estimated from an initial state-level alignment on the coustic training data using a Viterbi decoder. The probability $p(x_t)$ is assumed to be "independent of the word sequence and can be ignored during decoding" [13]. Hence, to get the emission probability $P(x_t|s_t)$, the posterior probability $p(s_t|x_t)$ should be divided by the prior $p(s_t)$. However, it is asserted that this division may not provide improvement in recognition accuracy under some conditions [13].

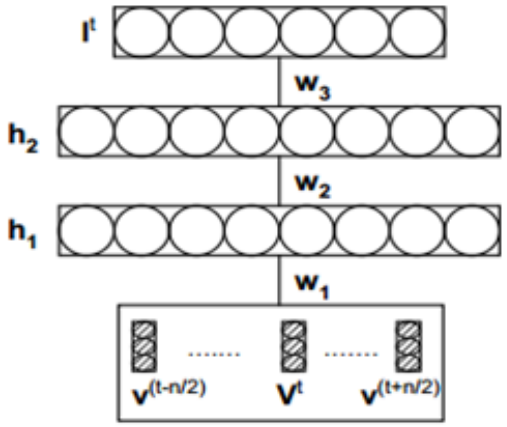

Figure 4.1: Frame-level DNN for phoneme recognition [55]

The DNN-HMM hybrid architecture can also be used for acoustic modeling in continuous speech recognition task, as presented in [13, 14]. In contrast to the context-independent (CI) DNN-HMM hybrid architecture described above for phoneme recognition, the DNN can be trained to predict probability distributions over tri-phone HMMs, forming thus a context-dependent (CD) DNN-HMM hybrid architecture for large-vocabulary continuous speech recognition task. The idea is to use senones as modeling units rather than monophones. The method involves two main steps [13]. First, the Viterbi algorithm is used to generate the senone-level alignment on a tied tri-phone GMM-HMM baseline. Then, the DNN-HMM is trained to predict the senones in each frame or sequence of frames. It has been proven that the better the baseline system used during forced alignment, the better the final results of the CD-DNN-HMM system [32, 13]. An illustration of DNN-HMM hybrid architecture for continuous speech recognition is presented in figure 4.2.

DNN-HMM hybrid architecture has shown successful results and good recognition performance in both isolated and continuous speech recognition tasks. In fact, DBN-DNNs have proven capability to outperform GMMs in acoustic modeling. The following subsec-

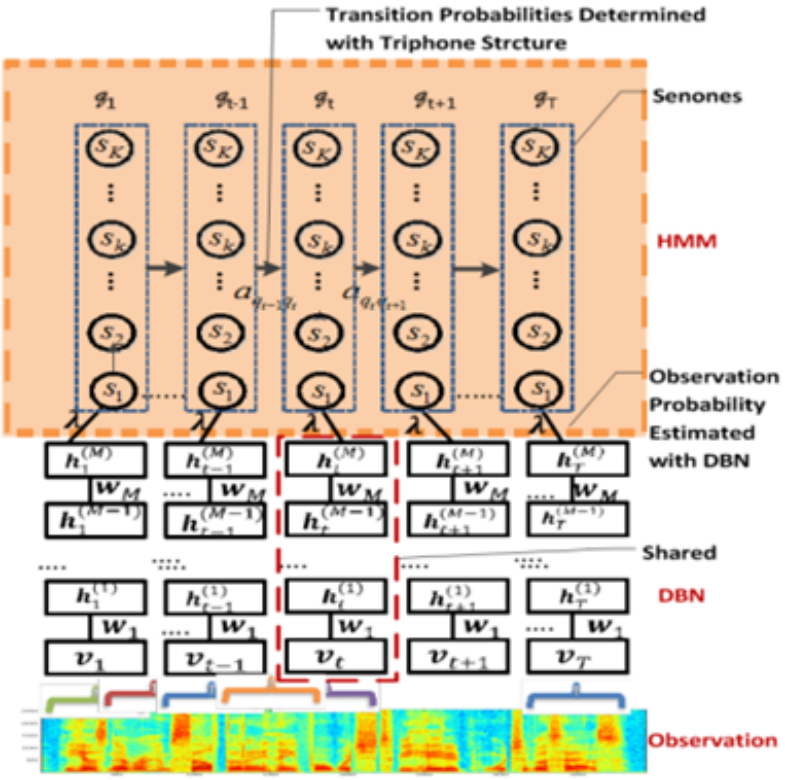

Figure 4.2: Interfacing DNN and HMM for continuous speech recognition [84]

tion explores the main differences between GMMs and DNNs, and the main strengths of both deep-layered networks and the new learning algorithm for acoustic modeling.

4.3.2 Advantages of DNNs for Acoustic Modeling

DNNs have proven efficient in acoustic modeling and shown results competitive with GMMs. However, many differences exist between GMMs and DNNs, which have made DNNs more powerful for speech recognition. Unlike GMMs, DNNs are more efficient at modeling highly non-linear data. Indeed, [32] with many hidden units, DNN can easily model multiple simultaneous events within one frame or window. However, doing so is very difficult for GMM because each data point is assumed to be generated by a single Gaussian in the mixture. In addition, DNN can efficiently be trained using multiple frames

of acoustic features; whereas GMM, which requires uncorrelated data, is unable to exploit multiple frames [32].

Despite the advantage of the EM algorithm for training GMMs, which can be easily parallelized on a cluster machine [32], DNNs present many advantages that have made them the most prominent technique for acoustic modeling in speech recognition. In [58], the success of DNNs over GMMs in acoustic modeling was attributed to three main facts. First, neural networks are very flexible at modeling the data without any preliminary assumptions about its distribution. In addition, DNNs are exponentially more compact than GMMs due to the "distributed representation" of the input, which can be explained by the big number of units that together can represent a single input vector. Second, the deep architecture of DNNs that are composed of many nonlinear layers plays an important role in improving the flexibility and the modeling capacity of the network. In fact, extracting higher representation and a highly non-linear statistical structure at each layer, akin to the human speech recognition, has a great impact on improving the recognition accuracy of the system. Finally, the generative pre-training phase using an unsupervised layer-wise training algorithm represents the most important advantage of DBN-DNNs. On one hand, this phase aims at extracting regularities in the input data so as to adjust the weights for a good generative model; on the other hand, it initializes the weights of the network for a good discrimination between the labels during the supervised learning stage. Indeed, the pre-training phase plays an important role in reducing over fitting, improving the generalization, and reducing the computation during the discriminative fine-tuning with back-propagation learning algorithm.

Due to these advantages, generatively pre-trained DNNs are competitive to GMMs for acoustic modeling, sometimes with better recognition results.

4.4 Conclusion

This chapter has provided an overview about HMMs, their parameters and their learning and decoding algorithms. It also discussed the limitations of GMMs in GMM-HMM acoustic models for speech recognition.

Replacing GMM with DNN in HMM-based acoustic model has shown successful results in many speech recognition tasks. In this chapter, the technique for interfacing DNN with HMM in both isolated and continuous speech recognition systems is detailed. In addition, the advantages of DNNs are discussed. In fact, combining pre-trained DNNs with HMMs within a single hybrid architecture for acoustic modeling intendes to combine the representational power of DBN-DNNs and the sequential modeling capability of HMMs.

In order to empirically confirm the power of DNNs in modeling speech data, DNN-HMM acoustic models for phoneme recognition are implemented in the next chapter. A systematic implementation procedure is also provided. Finally, the next chapter experimentally investigates the major factors behind the success of DNNs in acoustic modeling.

Chapter 5

Implementation of DNN-HMM Acoustic Models for Phoneme Recognition

This chapter describes the implementation of DNN-HMM hybrid acoustic models for phoneme recognition. The DNN is trained with the new deep learning procedure to produce posterior probabilities over the states of mono-phone HMMs. The approach is the same as that successfully used in [56] with minor variations. The main goals are (i) to empirically confirm the potency and effectiveness of DNN in acoustic modeling, and (ii) to investigate the major factors behind its power. Essentially, a step-by-step implementation procedure is provided, including the development of the GMM-HMM baseline, which sheds light onto the understanding of major details in successfully applying DNN for acoustic modeling in speech recognition.

The implemented DNN-HMM hybrid system shows good improvement over the GMM-HMM baseline in phoneme recognition, which provides experimental evidence that generatively pre-trained DNNs can replace GMMs in HMM-based acoustic models. Several experiments are conducted to evaluate and analyze the DNN performance and capability of modeling speech data. This work also confirms that DNN-HMM hybrid models provide accurate acoustic estimates in comparison to GMM-HMMs, which often underestimate them.

5.1 DNN-HMM Acoustic Models for Phoneme Recognition

5.1.1 Implementation Procedure of DNN-HMM

In the following, the steps for developing DNN-HMM hybrid acoustic models for recognizing phonemes are outlined.

1. Speech data pre-processing

   The first step to building a speech recognizer is to analyse the speech data files of interest as well as the transcription files. Actually, the speech signals must be converted into the appropriate forms by extracting the most useful features for the recognition task. Different type of features can be used such as PLP, filter-banks, LPC, etc. However, in this work, the Cepstral features are chosen. Speech waveforms are transformed into a sequence of MFCC vectors, each vector is of length 39. The MFCCs are derived from a window of 25ms advanced by 10ms. Each frame is represented by 12^{th}-order MFCCs and energy along with their first and second temporal derivatives. To extract these features, the HCopy tool in HTK [1] is used, having as input the configuration file depicted in Figure 5.1. The file contains values of the parameters required for coding the speech. In fact, the speech was analyzed using Fast Fourier Transform (FFT) with Hamming windows. The signal should have first order pre-emphasis applied using a coefficient of 0.97, and the filter-bank should have 26 channels.

```
SOURCEFORMAT = WAV
TARGETKIND = MFCC_E_D_A
TARGETRATE = 100000.0
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS  26
CEPLIFTER = 22
NUMCEPS = 12
```

Figure 5.1: Parameters for coding the speech data

   On the other hand, the transcription files may also need preparation. Typically, the structure of the original phone-level transcription files may not be exactly as

required, for example, because of the way of determining the phoneme boundaries (i.e., in terms of samples or time). Finally, a file that defines the grammar of the language and another that contains the list of the phonemes in the vocabulary should both be created.

2. Implementing the GMM-HMM acoustic models

   In the beginning, the topology of the HMM should be defined. For phoneme recognition, 5-states left-right HMMs are often used, whose entry and exit states are non-emitting, as shown in Figure 5.2. For each phoneme in the vocabulary, an HMM acoustic model is defined. Thus, the HMMs are mono-phones with continuous-density mixture-Gaussian states. All the phoneme models are initialized to be identical. In fact, all the Gaussians in HMM states should initially have the same mean and variance equal to the global speech mean and variance. For the transition probabilities, sensible values should be assigned (i.e., the sum should be equal to one). Initially, each HMM state should have a single Gaussian. Then, the GMM-HMM models are refined incrementally using a Gaussian splitting strategy. Better recognition performance is expected from increasing the number of mixtures in the Gaussian at each state. The Baum-Welch algorithm is jointly applied in order to compute the re-estimates of the HMM parameters. Better results are obtained from running the algorithm three consecutive times, using the whole speech training data each time.

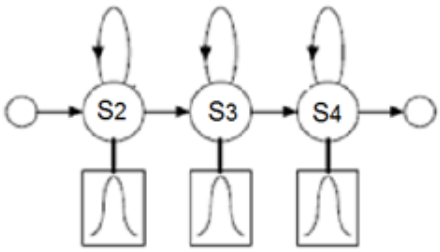

Figure 5.2: Mono phone 5-states HMM with single Gaussian at each state

3. Building the language models

   From the training dataset, bi-gram language models over phonemes are estimated using the HLStats and HBuild tools in HTK.

4. Evaluating the GMM-HMM phoneme recognizer

With all the speech recognition components now complete, the GMM-HMM baseline can be tested on speech test files. To this end, the Viterbi decoder HVite in HTK is used, which matches the speech data against the set of acoustic and language models, and outputs a transcription file. During the recognition, the language model scaling factor and the word insertion penalty have to be determined empirically.
The recognition performance of the GMM-HMM baseline is determined using the standard Phone Error Rate (PER) metric defined as follows:

$$PhoneErrorRate(PER) = \frac{(S + D + I)}{N} * 100 \tag{5.1}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the total number of phonemes in the reference. In fact, the number of substitution, deletion and insertion errors is computed using the best alignment between the manually defined phone sequence (reference translation) and the recognized sequence (decoder output translation).
The HResults tool in HTK usually provides two measures: the Correctness and the Phone Accuracy Rate (PAR) defined as follows:

$$Correctness = \frac{C}{N} \tag{5.2}$$

where C is the number of correct phonemes, and N is the total number of phonemes in the reference.

$$PhoneAccuracyRate(PAR) = 100 - PER \tag{5.3}$$

5. Generating the forced HMM alignments

   After training the GMM-HMM and achieving some reasonable recognition results, the Viterbi algorithm is then used to generate the state-level alignments on the acoustic training data. These HMM-state alignments are used to determine the labels of the DNN input data during the discriminative training. In fact, instead of providing the phoneme sequence that best fit the speech data according to the acoustic and language models, the Viterbi decoder takes as input the exact transcription of the speech data and provides a new frame-level label file that identifies the time boundaries of a particular phoneme in the acoustic data. Hence, the new output transcription file contains two levels of transcription: the first level determines the phoneme's name and the second level determines the state number of the associated

mono-phone HMM.
Example:

start segment	end segment	phoneme[state]
5400000	5500000	$sh[2]$
5500000	5700000	$sh[3]$
5700000	5800000	$sh[4]$

where the first and the second parameters correspond to the time segments in 100 nanosecond , sh is the phoneme's name, and the number in '[]' identifies the number of the HMM state. Since the first and the last states of the 5-state HMM are non-emitting, they will not be included in the transcription.

6. Building and training the DNN

   A deep neural network with a fixed input and fixed Softmax output layer and many hidden layers should be defined. Nevertheless, the number of intermediate layers and the number of units per each layer are determined empirically to optimize the recognition performance.
   The input data for training the DNN correspond to a window of n successive frames of Cepstral coefficients. The data labels correspond to the HMM states of the central frame; they are produced by forced alignment using the GMM-HMM baseline as described in step 5. The number n should be odd. Thus, for training the network, the input data should be prepared. For an MFCC feature vector corresponding to the frame at time t, the $n/2$ preceding MFCCs vectors and the $n/2$ following MFCCs vectors are concatenated to the left and right, respectively. The speech feature vectors corresponding to the frames at the beginning and at the end of each speech data are padded with zeros.
   The input layer of the DNN is composed of $n * 39$ units (i.e., each feature vector is composed of 12^{th} order MFCCs and energy both augmented by their first and second temporal derivatives). The output layer is composed of the total number of phonemes in the vocabulary multiplied by the number of emitting HMM states (i.e., 3 states for each phoneme). A mapping file is needed to store the mapping of each mono-phone HMM state number to the corresponding DNN output label, as depicted in Table 5.1. The first and last HMM states are not included since they do not emit any probability distribution. This file is then used when interfacing DNN and HMM for speech decoding.

   The DNN is first generatively pre-trained as a DBN, where each pair of layers formed an RBM. Then, it is discriminatively fine-tuned with the back-propagation learning algorithm to estimate a posterior probability over the HMM state of the central

Table 5.1: Partial list of HMM state ID -DNN class label ID mapping

Mono-phone HMM state id	**DNN class labl id**
sh[2]	171
sh[3]	172
sh[4]	173
m[2]	174
m[3]	175

frame.

7. Predicting probability distributions over the labels of the testing data

   The testing data should also be pre-processed the same way as the training data in order to fit the input layer of the DNN. For each test input, the DNN produces probability estimations over the different labels of the central frame. Indeed, the labels represent the states of the mono-phone HMMs. The label with the highest probability defines the predicted state.

8. Speech decoding using the DNN-HMM acoustic models

   To generate the phoneme sequences from the speech data, the mono-phone HMM definitions that contain the transition probabilities between the states, the DNN output probabilities, and the previously defined language models are fed to a Viterbi decoder implemented in Python. In fact, at each state of the HMM, instead of using the output probability defined by the GMM, the decoder uses the probability defined by the DNN. Typically, the HMM defines a prior over the states; hence, the posterior probabilities over the HMM states defined by the DNN should be divided by the respective prior [22]. However, for this type of architecture, that step is asserted to be unnecessary [56].
   When combining the DNN-HMM acoustic models and the language models, the same values for the language model scaling weight and the word insertion penalty used with the GMM-HMM baseline may not be used. Yet, new values are empirically determined to optimize the recognition performance.

9. Recognition evaluation of DNN-HMM acoustic model for phoneme recognition

   Once the transcription files are generated from the Python Viterbi decoder, they are compared to the original transcription file using the HResults tool in HTK in order to evaluate the DNN-HMM recognition performance. The same evaluation metrics are used as for the GMM-HMM baseline and defined earlier in step 4.

5.1.2 Challenges

To implement DNN-HMM acoustic models for phoneme recognition, two major challenges are faced.
The first is related to the huge amount of speech data after pre-processing, which leads to very slow training. In consequence, it is hard to perform extensive experimentations to investigate the behaviour of the DNN and its impact on phonetic recognition performance. The second major barrier to the implementation of DNN is on how to choose appropriate values for hyper-parameters, such as the learning rates, the number of layers, and the number of units per layer, that enable DNN to outperform GMM. There are no formal and clear standards available, and many attempts should be carried out using different values. However, testing one setting of these parameters is computationally very expensive and time consuming. On the other hand, a sensible value for one hyper-parameter may potentially affect the performance of the DNN on acoustic modeling, and may lead to very poor results.
To tackle the problem of slow training, it is worthwhile to explore parallel programming on powerful Graphical Processing Units (GPUs).

5.2 Experimental Setup

The Hidden Markov Model Tool Kit (HTK) version 3.4 [1] is used to implement the GMM-HMM baseline, and Python 2.7.8 is used to implement the DNN. This section provides a detailed description of the speech corpus, as well as the hardware and software computation setup.

5.2.1 Dataset Description

The experiments were conducted on the TIMIT corpus [2], a dataset composed of 6300 sentences recorded from 8 different English dialects in the United States. Every speaker from a total of 630 speakers spoke 10 sentences: 2 dialect sentences (the SA sentences), 5 phonetically-compact sentences (the SX sentences), and 3 phonetically-diverse sentences (the SI sentences). Following the work in [56], the SA records were eliminated, because they are used to expose the dialectal variants of the speakers, thus they may bias the results. The TIMIT corpus is decomposed into a training set and a testing set. In all experiments, the phonetic recognition results are reported using the standard core test set of 24 speakers of whom each spoke 5 SX sentences and 3 SI sentences.

Using the HTK toolkit for extracting the MFCC features and for building the HMM-based acoustic models and the phoneme recognizer, first required modification of the wave-form file format of the TIMIT dataset into the supported HTK file format [83]. For the HTK requirement, the phone-level transcription files of the TIMIT were also subject to some modifications. In fact, the boundaries of the phonemes in each sentence are given in the sample number; however, the HTK label format is in 100 nanoseconds. The speech in the TIMIT was sampled at 16 kHz. Thus, equation (5.4) and equation (5.5) were respectively applied to the start and the end of each segment in the phoneme transcription file.

$$htk\_phone\_start = \frac{timit\_phone\_start}{16000} * 10^7 \tag{5.4}$$

$$htk\_phone\_end = \frac{timit\_phone\_end}{16000} * 10^7 \tag{5.5}$$

Finally, in using the HTK tools, it was found that replacing the non-speech markers 'h#' at the start and the end of each sentence in the TIMIT phoneme translation files by 'ENTER' and 'EXIT' makes the implementation process relatively easy. Hence, the total number of phoneme labels will be 62 instead of 61.

The speech waves were analyzed using 25-ms Hamming window advanced by 10-ms. The acoustic data was then represented by the first 12 MFCCs (excluding the coefficients of order zero) and by the energy both augmented by their first and second temporal derivatives. The extracted features were then normalized to have zero mean and unit variance.

Typically, to measure phonetic recognition performance after decoding, only 39 phonemes are used. Thus, the 61 TIMIT phonemes are mapped, after decoding, to the standard 39 phonemes. Table 5.2 lists the phonemes that were replaced, the other phonemes were kept unchanged.

5.2.2 Computational Setup

The development of the GMM-HMM baseline was carried out on a Lenovo notebook computer, which has an Intel i7 processor with a CPU clock speed of 2.4GHz, and 8GB RAM. Accelerating the training of the DNN, which was computationally very intensive, and building the DNN-HMM speech recognizer necessitated exploration of GPUs with sufficient memory to handle the large amount of training data. However, this kind of hardware

Table 5.2: Mapping the TIMIT phonemes into the standard phonemes

Standard phonemes	**TIMIT phonemes**
aa	aa, ao
ih	ih, ix
l	l, el
m	m, em
sh	sh, zh
uw	uw, ux
ah	ah, ax, ax-h
n	n, en, nx
er	er, arx
hh	hh, hv
ng	ng, eng
cl	bcl, pcl, dcl, tcl, gcl, kcl, epi, pau, h#

was not immediately available. To solve this problem, access was obtained to the Sharcnet Monk cluster system [3] from the Sharcnet organization [4] which provides High Performance Computing(HPC) resources in the form of online clusters. The Monk system is composed of 54 nodes each containing 8 CPUs Intel E5607, with 4 cores and a CPU clock speed of 2.26 GHz, and 48 GB memory. Each node also has two Nvidia Tesla M2070 General Purpose Graphical Processing Units (GPGPUs). This kind of GPU contains 6GB of GDDR5 memory and 448 processor cores, a 1.15 GHz processor core clock, and a 1.566 GHz memory clock.

Sharcnet was useful not only in accelerating the training of the DNN by exploring the GPUs, but also in the possibility of launching the many jobs at the same time, each job handling the training of a DNN with a different configuration. However, Scharcnet system has the major drawback of waiting time when the system is busy. The jobs may not run immediately but may hold in the queue sometimes for many days.

The Theano Python library is used to perform computations at a functional level. This library dynamically performs the code generation and compilation with transparent use of CPU and GPU. It internally uses the CUDA library to perform matrix operations on the GPU.

5.3 Experiments

5.3.1 Building the GMM-HMM Baseline System

To generate the frame-level labels for training the DNN and to compare the performance of the DNN with that of the GMM in HMM-based phonetic acoustic models, it is crucial to build a mono-phone GMM-HMM baseline system. The entire TIMIT standard training set was used for training the GMM-HMM baseline. This training was carried out using the HERest tool of the HTK. Initially, each state of the HMMs had a single Gaussian. The baseline system was then optimized by increasing the number of Gaussians in the mixture using a splitting strategy. The splitting was performed using the HHed tool of the HTK, according to the strategy explained in [83]. Every time the number of mixture components was increased, the baseline system was re-trained three consecutive times. The parameters of the Viterbi decoder were fixed to 5.0 and 2.5 for the language model scaling factor and word insertion penalty, respectively. The performances of the different configurations of the GMM-HMM baseline on the TIMIT core test set are summarized in Table 5.3.
As expected, increasing the number of components in the GMM provides better phone recognition accuracy. With 17 mixture components at each state, the phone accuracy rate reaches 69.84%.

Table 5.3: Recognition Performance on TIMIT core test set of the GMM-HMM baseline as a function of the number of Gaussian mixture components

GMM-HMM	**Correctness**	**Phone Accuracy Rate (PAR)**
mono Gaussian	55.67%	41%
2-mixture Gaussians	66.34%	62.25%
3-mixture Gaussians	68.78%	65.3%
5-mixture Gaussians	69.95%	66.31%
9-mixture Gaussians	71.55%	68.13%
17-mixture Gaussians	**73.09%**	**69.84%**

However, as the number of Gaussian components increases at each state, the HMM training becomes computationally expensive in time. As shown in Figure 5.3, the training time exponentially increases with a growing number of Gaussians. This rise is due to the heavy computation performed in re-estimating the GMM-HMM parameters.

The parameters of the Viterbi decoder were not arbitrarily chosen; instead, a set of experiments was conducted in order to empirically determine the values of the language

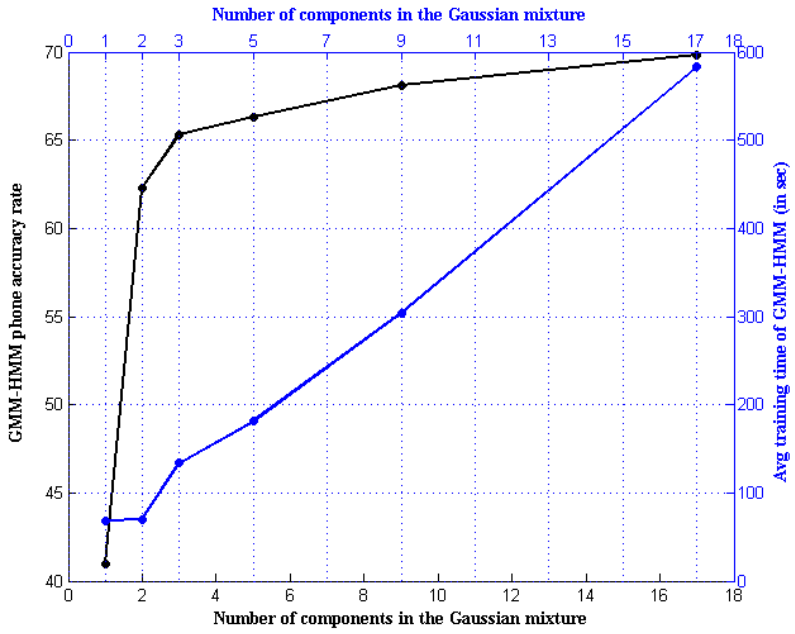

Figure 5.3: GMM-HMM phone accuracy rate and average training time vs. the number of Gaussian mixtures at each HMM state

model scaling factor and the word insertion penalty such that the recognition performance was optimized. Table 5.4 reports the accuracy of the GMM-HMM baseline with different values of the Viterbi parameters. It clearly shows that the performance increases with the rise in assigned values.

In fact, HMM-based acoustic models usually underestimate the acoustic probabilities, due to the unrealistic Markov independence assumptions [38]. To balance between the acoustic model scores and language model scores, a scaling weight is applied to the language model probability [21]. This weight is typically > 1 .

On the other hand, the language model probability also functions as a penalty for inserting words. Hence, modifying the language model weight to balance the underestimated acoustic probability has a side effect of adjusting the insertion penalty. In fact, if the penalty is large (small scaling weight), the decoder will prefer fewer longer words, and if the penalty is small (large scaling weight), the decoder will prefer a greater number of shorter words. To address this problem, a separate word-insertion penalty is often used [38].

Table 5.4: GMM-HMM performance with different values for Viterbi parameters

Language Model Scaling Factor	**Word Insertion Penality**	**TIMIT test core Phone Accuracy Rate (PAR)**
1.0	0.0	66.57%
1.5	2.0	68.21%
2.5	5.0	69.84%

5.3.2 Training the DNN

For all experiments, context windows of 11 frames (5-1-5) of MFCC vectors were used as input features to the DNN. Hence, the input layer was composed of 429 units (i.e., each frame is represented using the 12^{th}-order MFCCs and energy with their first and second derivatives). The DNN output layer was composed of 186-target class labels, since each of the 62 phonemes was modeled with 3-states HMM.

In all experiments, the network was composed of a stack of RBMs. It was first generatively pre-trained by using unsupervised learning that learned one layer of stochastic binary features at a time. The output of one layer was the input to the next layer. The unsupervised learning algorithm based on contrastive divergence aimed at initializing the weights of the network by exploring the data without having any knowledge of the labels. Then, the standard discriminative back-propagation learning algorithm with gradient descent was used to adjust the weights and predict the labels of the central frame. In fact, the weight updates were calculated using mini-batch learning and not fully iterative (on-line). The size of the mini-batches was fixed to 120 training patterns.

All experiments used the TIMIT standard training set for training the DNN. However, 15% of this set was excluded for validation in order to investigate and tune the parameters (i.e., learning rates, number of pre-training and fine tuning epochs) and the meta-parameters of the DNN(i.e., the number of hidden layers and the size of each layer). Throughout the following discussion, the number of layers takes into account the input and output layers; however, when the number of hidden layers is used to describe the architecture of the DNN, this number means that the input and output layers are not counted.

After processing the speech data and extracting the MFCC feature vectors, the size of the data used for training the DNN was up to 2GB, which made the training computationally very expensive and hence not feasible on a standard computer. It was crucial to use a high-performance computer to explore this huge set of data and the large number of model parameters. However, the training was still very slow.

The best solution was to use parallel computing on GPUs that were able to efficiently accelerate the training up to 10 times.

Table 5.5 summarizes the difference between the Sharcnet Monk system CPU and the same system's GPU in training DNNs with 3 and 4 hidden layers, the size of each layer was fixed to 512 units. It can be seen that training a 3-hidden layer network with 100 epochs for both pre-training and fine-tuning may take up to 6 days with the CPU. This time length will fall to 15 hours only, using GPUs.

Table 5.5: Summery of DNN training time with CPU and GPU

Processing Unit	**Number of Hidden Layers**	**Pre-training time per epoch**	**Fine-tuning time per epoch**
CPU	3	57 min	24.68 min
GPU	3	8.3 min	0.6 min
CPU	4	83 min	33 min
GPU	4	12 min	0.72 min

In fact, with implementation on GPUs, it was possible to take advantage of parallelization at two levels [74]: data level and layer level. At the data level, the patterns of each mini-batch were distributed and processed in parallel across the graphic processing cores. However, at layer level, the neurons at each layer were processed in parallel, and separate graphic cores handled the activation of different neurons.

Before training the DNN, many decisions needed to be made related to the parameters and meta parameters of the network. The next subsection examines the impact of these choices on the accuracy of the DNN-HMM speech recognizer.

5.3.3 DNN-HMM Results and Analysis

After training, the DNN provided for each frame a posterior probability over the states of the HMMs. These probabilities were used to replace the ones estimated using GMMs. They were fed to the Viterbi decoder, along with the mono-phone HMMs definitions that contained the transition probabilities between the states. In all experiments, the bi-gram language models created earlier for the GMM-HMM baseline were used. After investigations, the language model scaling factors and the word insertion penalty were fixed to 1.0 and 0 respectively. The phoneme recognition performance of the DNN-HMM hybrid system is reported on the TIMIT core test set.

With different architectures and learning parameters, all the developed DNN-based acoustic models are able to outperform the GMM-HMM baseline. Deploying only 2-hidden layer DNN, where each layer has 1024 units (i.e., neurons), the DNN-HMM-based speech recognizer achieved 73.76% Phone Accuracy Rate, while the baseline provided 69.84%.

To investigate the success of DNNs in acoustic modeling, the following three criteria were studied:

- the effect of varying the number of hidden layers and the number of neurons per hidden layer in the DNN on phoneme recognition accuracies,
- the role of pre-training and fine-tuning in improving the recognition results,
- the efficiency of the DNN at acoustic estimation.

Finally, the DNN-HMM recognizer is evaluated with comparison to other reported results.

Sensitivity Analysis of the Number and Size of Hidden Layers in DNN

A series of experiments were conducted on DNNs with different architectures. The first test consisted of fixing the number of hidden layers to 3 and varying their respective number of neurons. Note that the number of neurons is always the same in all the hidden layers of the DNN. The number of pre-training epochs was fixed as follows:

- 220 for the Gaussian-Bernoulli RBM, i.e., first hidden layer, with a learning rate equal to 0.002,
- 80 for the Bernoulli-Bernoulli RBMs, i.e., second and third hidden layers, with a learning rate equal to 0.02.

The fine-tuning was performed using 100 epochs and a 0.02 learning rate. Table 5.6 reports the DNNs test accuracy in predicting the labels of the central frame, and the DNN-HMMs speech recognition performance versus the number of neurons per hidden layer. It can be observed that the DNN test accuracy and the phone accuracy rate of the DNN-based recognizer are correlated. Additionally, Table 5.6 demonstrates that when the number of neurons is increased from 512 to 800 per hidden layer, the best phone accuracy rate is reached. In fact, if the number of neurons is increased beyond 800, an over-fitting problem arises, degrading the DNN performances. Indeed, the number of neurons in the hidden layers is an important parameter to the DNN to avoid this over-fitting problem.

Table 5.6: The performance of 3-hidden layer DNN vs. the number of hidden neurons

3-hidden layer DNN	**DNN test accuracy**	**DNN-HMM Phone Accuracy Rate**
512 units per hidden layer	60.51%	73.60%
800 units per hidden layer	**61.07%**	**73.70%**
1024 units per hidden layer	60.85%	73.53%
1500 units per hidden layer	58.86%	72.50%
2048 units per hidden layer	55.73%	71.64%

In other words, it is crucial to find the ideal number of neurons to effectively model the speech data, i.e., to achieve the optimal acoustic modeling performance.

It is worth mentioning that different tests were performed outside the scope of this report, and the optimum configuration was recorded with a 2-hidden layer DNN, each layer having 1024 neurons. Consequently, the second sensitivity analysis consisted of varying the number of hidden layers while fixing their sizes to 1024.

The DNN was pre-trained with a fixed learning rate of 0.002 for the first hidden layer (i.e., the Gaussian-Bernoulli RBM) and 0.02 for the others (i.e., Bernoulli-Bernoulli RBMs). The network was fine-tuned with a fixed number of epochs and a learning rate equal to 100 and 0.02, respectively. In addition, the number of pre-training epochs was varied. As depicted in Table 5.7, while the number of pre-training epochs was fixed, increasing the number of hidden layers from 2 to 3 slightly reduces the performance of the DNN. However, the DNN-based speech recognizer still outperform the GMM-HMM baseline. The test accuracy of the DNN with 4 hidden layers was very poor, less than 10%. Nevertheless, it is worth mentioning that only 120 epochs were used in the pre-training of the first RBM.

Increasing the number of hidden layers and the number of hidden neurons was expected to result in an improvement in DNN performance and in speech recognition accuracy. However, as it was shown by the above analysis, larger networks render the DNN prone to over-fitting problem. This later drives the network to learn rare dependencies and variations in the speech input data at the expenses of its generalizability to non trained data.

In the following, the speech recognition performances dependency on the learning parameters during the pre-training and fine-tuning is given.

Table 5.7: Performance of the DNN with different number of hidden layers and different number of pre-training epochs

Number of hidden layers	**Number of hidden units**	**Number of pre-training epochs for the first layer**	**Number of pre-training epochs for other hidden layers**	**DNN test accracy**	**PAR**
2	1024	220	80	59.98%	73.76%
3	1024	220	80	59.85%	73.53%
4	1024	120	80	7.07%	22.72%

The Role of Pre-Training and Fine-Tuning

For the three DNN architectures presented in Table 5.7, it can be observed that a DNN-based speech recognizer with only two hidden layers achieves the optimal phone accuracy performances of 73.76%. This confirms the power of the deep learning algorithm at modeling the speech data even with shallow architecture. However, in the case of 4-hidden layer DNN, when the pre-training was performed with only 120 epochs for the first hidden layer, the phonetic recognition accuracy degrades to 22.72%. These results clearly explains the role of the pre-training to extract the most useful information from the speech data, which is performed mainly in the first hidden layer where the visible units (i.e., neurons) of the RBM were fed with the original MFCC features. In fact, the pre-training phase based on generative learning plays an important role in creating a powerful new statistical representation of the input at each layer.

As previously mentioned in the sensitivity analysis of the number and size of hidden layers in DNN, the performance of the DNN at predicting the labels of the central frame of the input has a direct impact on the performance of the DNN-HMM-based phoneme recognizer. Figure 5.4 shows the DNN error rate for the validation set at each epoch during the fine-tuning. This describes the prediction performance of the three DNN architectures presented in Table 5.7. It is clear that the accuracy of the 3-hidden layer DNN in assigning the right label to the central frame of the 11 frames MFCC features was better than that of the 2-hidden layer DNN. This remained true for the 66 first epochs, yet at epoch 67, the validation error rate of the 3-hidden layer DNN started to increase, while for the 2-hidden layer DNN it continued to decrease. Conversely, the 4-hidden layer DNN performances was very poor, and the discriminative error did not drop below 90% even after 100 epochs fine-tuning.

Figure 5.4 shows that the 2-hidden layer and 3-hidden layer DNN achieve already an acceptable validation error when using only one epoch. This confirms the importance of the pre-training phase in initializing the weights of the DNN, which enables a rapid convergence of the fine-tuning phase. However, further optimization of the fine-tuning step is needed by carefully choosing the learning parameters.

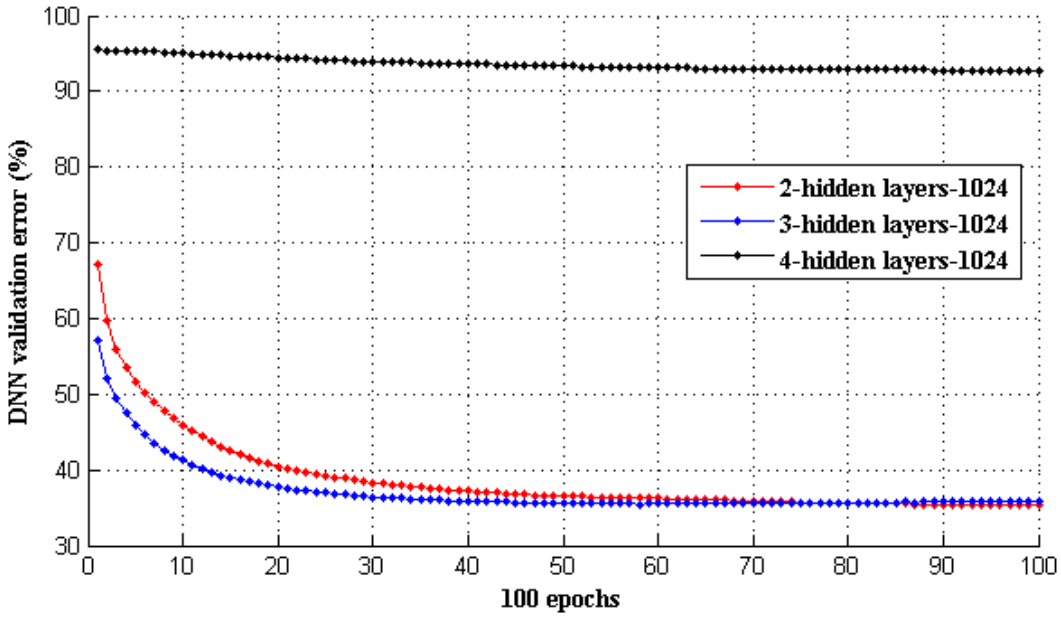

Figure 5.4: Validation error of DNN with different number of hidden layers vs. the number of fine-tuning epochs

Investigation the fine-tuning process dependency on the learning rate is performed as it follows. A DNN with 3 hidden layers, where the number of neurons at each hidden layer is equal to 1024, is fine-tuned using different learning rates as shown in Figure 5.5. It can be observed that with a learning rate set to 0.1, the DNN achieved 52.32% error at the first epoch. This error decreases with the number of epochs until the epoch 36 where it started increasing. A similar behaviour is noted when the learning rate was set to 0.05, where the error started at 57.7%, dropped to 35.6%, and then increased again at epoch 70. However, when the learning rate was set to 0.02, one can observe that the error of the DNN was very high during the first 10 epochs, but it continued to readily decrease to reach 35.36% at epoch 100. One can conclude that it is better to start the fine-tuning with the learning rate set to 0.1 and to dynamically adjust it during the training with respect to the value of the error on the validation set [56].

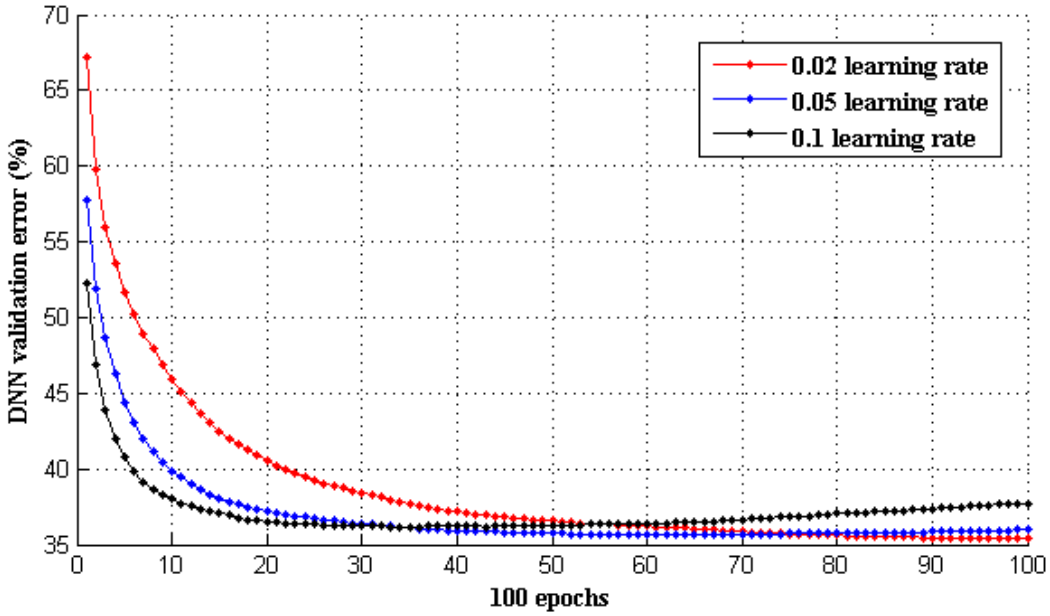

Figure 5.5: Validation error of DNN with different fine-tuning rates vs. the number of fine-tuning epochs

The Efficiency of the DNN at Acoustic Estimation

As asserted in the beginning of this section, in all the previous experiments on the DNN-HMM acoustic models, the values for the language model scaling factor and the word insertion penalty were set to 1.0 and 0, respectively, during the decoding. However, for the GMM-HMM baseline, larger values were used in order to optimize the speech recognition performance. This was explained by the fact that the acoustic probabilities were underestimated. Applying the same values used in the GMM-HMM decoding to the DNN-HMM provided poor results, and showed that the recognizer was not able to outperform the GMM-HMM baseline, as reported in Table 5.8. However, without using the language scaling and the word insertion penalty, good phonetic recognition results were achieved. This affirms the capability of the DNN-HMM acoustic models at providing good acoustic estimations, unlike GMM-HMMs, which often underestimate them.

In fact, it has been asserted that posterior probabilities provided by the neural network can generate good acoustic confidence measures that confirm how well the model fits the speech data [79, 78]. Indeed, the local posterior probability estimates of the mono-phone

Table 5.8: Effect of varying the values of LMSF and WIP on recognition accuracy

3-hidden layer DNN	LMSF	WIP	Phone Accuracy Rate
512 units per hidden layers	5.0	2.5	60.8%
	1.0	0.0	73.67%
800 units per hidden layers	5.0	2.5	60.02%
	1.0	0.0	73.69%
1024 units per hidden layers	5.0	2.5	60.39%
	1.0	0.0	73.53%

HMM states, given a frame of acoustic data, can be combined to produce a Viterbi estimate of the global posterior probability of a word sequence, given the acoustic observations, as carried out in [78]. Hence, a DNN-HMM hybrid model can be used as a speech decoder, using a modified Forward-backward algorithm.

Comparison with Other Works

Table 5.9, compares the results achieved in this work to previously published results based on DNN for acoustic modeling in phoneme recognition, in terms of phone error rate (PER).

Table 5.9: Reported results on TIMIT core test set

Method	PER
Interpolating Conditional Restricted Boltzmann Machines (ICRBMs) [57]	26.7%
DBN-DNN (this work)	26.24%
Full-Sequence DBNs [60]	22.5%
DBNs with Mean-Covariance RBM [15]	20.5%
DBNs [56]	20.7%

Although the implemented DNNs did not outperform other works that used neural networks with the deep learning technology for phoneme recognition on the TIMIT core test set, the developed DNN-HMM recognizer has performed at a reasonable level. The results obtained are interesting but further improvements are required. These improvements should include the optimization of the baseline since it has a big influence on the DNN-HMM performance, as asserted in [13, 14]. Moreover, more extensive experiments are required to optimize the pre-training and fine-tuning steps. Good decisions for setting

the hyper-parameter are able to improve the DNN performance. By going deeply into understanding all the steps for implementing DNN-HMM hybrid architecture for speech recognition, this work represents a guide toward using DNN for more powerful acoustic models.

5.4 Conclusion

In this chapter, DNN-HMM hybrid acoustic models for phoneme recognition are implemented. Initially, a systematic implementation procedure is provided, including the development of a GMM-HMM baseline system. This procedure intends to outline all the steps in combining DNN with HMM for acoustic modeling in phoneme recognition. Two major challenges were faced to implement the DNN-HMH-based phoneme recognition system: first, the slow training of DNN due to the large amount of speech data; second, the sensitivity in setting the values for the DNN parameters. To accelerate the DNN training, parallel computing using GPUs was exploited.

To optimize the performance of the GMM-HMM baseline, the number of Gaussian components at each HMM state was increased. However, this increase resulted in heavy computation. The deployed DNN-HMM phoneme recognition system outperformed the GMM-HMM baseline on the TIMIT core test set. Investigating the performance of DNNs has followed three different axes: first, the effect of varying the depth and the width of the network on modeling the speech data; second, the role of pre-training and fine-tuning in improving the recognition results; finally, the efficiency of the DNNs at acoustic estimation.

In the next chapter, a summary of the work is presented, and potential future research directions are discussed.

Chapter 6

Conclusion and Future Works

Conclusion

The main goal of this thesis was the development of Deep Neural Network-Hidden Markov Model (DNN-HMM) hybrid acoustic models for phoneme recognition, with an emphasis on providing a systematic implementation procedure. This thesis also aimed to empirically confirm the capability of DNNs to outperform Gaussian Mixture Models (GMMs), and to investigate the performance of DNNs in acoustic modeling.

In this thesis, a thorough overview of the fundamentals of speech recognition was presented, including the description of the different components required to build Automatic Speech Recognition (ASR) systems. Theories about DNNs, their architecture and learning algorithm were reviewed. The problems of GMMs in HMM-based acoustic models were discussed, and the advantages of DNNs were exposed.

To interface a DNN with HMMs for phoneme recognition, a DNN was trained to estimate a posterior distribution over the states of the mono-phone HMMs, given a window of 11 successive frames of speech data. The DNN was first generatively pre-trained layer-by-layer as a stack of Restricted Boltzmann Machines (RBMs), and then fine-tuned using the back-propagation learning algorithm in order to predict the label of the central frame. Training a neural network with many hidden layers and a large output layer on a large amount of data was computationally very expensive. To accelerate the training, parallel computing using Graphic Processing Units (GPUs) was exploited. In fact, GPUs were capable of performing parallelization at two levels: the data level and the layer level.

To generate the frame-level labels for training the DNN, and to compare the performance of DNN with that of GMM in HMM-based phonetic acoustic models, a mono-phone GMM-HMM baseline system was built. The developed DNN-HMM phoneme recognition system outperformed the GMM-HMM baseline with relatively 5% improvement, and achieved 73.76% Phone Accuracy Rate on the TIMIT core test set.

The investigations proved that by increasing the number of layers and the number of units per layer, the recognition accuracy does not necessarily increase. However, both the unsupervised pre-training phase and the fine-tuning phase had a great impact on the performance of the recognizer. During the generative pre-training, the units in the hidden layers played the role of feature extractors, and the intermediate layers detected new representation of the input speech data at each layer. In addition, this phase was very efficient at initializing the weights, which helped the network to rapidly converge towards better local minima during the discriminative phase. The choice of values for the learning parameters for fine-tuning has an impact on the final results of the DNN-based recognizer. It is better to dynamically update these values during the training so as to optimize the accuracy. Finally, it was proved that DNNs are capable of providing better acoustic estimations than GMMs, which often underestimate. Hence, when combining the acoustic model and the language model estimations during the Viterbi decoder, there was no need to apply the language model scaling factor and the word insertion penalty to balance the two estimations.

Future Works

Training a DNN on large amount of data is extremely computational expensive and is also slow process, even on GPUs. Given the preliminary investigative scope of this thesis and given the constraint of time and computational resources, it was not possible to perform much more extensive experimentation and outperform other reported results on the TIMIT core test set for phoneme recognition, which have been carried out by noted research centers at Microsoft, IBM and Google [32]. Further investigations are required to identify other major factors behind the performance of DNNs in acoustic modeling. The investigation could consist of varying the size of the input window for training the DNN. In fact, the goal of concatenating the frames, whereas the DNN predicts only the label of the central frame, is to take into consideration the context and temporal variability in speech data. However, when increasing the size of the input data, training DNNs can become costly.

In this work, the DNN was generatively pre-trained as a stack of RBMs. Other pre-training methods exist such as stacking auto-encoders [25]. It is also proven that the dropout technique can significantly avoid over-fitting and improve generalization error in DNNs for speech recognition tasks. This technique consists of randomly omitting a fraction of neurons in hidden layers during the pre-training [36]. However, dropout technique can drastically slow down the learning [12]. Using rectified linear units instead of the standard sigmoid units has proven efficient in accelerating the training and also improving the discriminative performance [12, 18]. Finally, planned future work will be directed towards applying DNNs for large-vocabulary continuous speech recognition tasks.

References

[1] `http://htk.eng.cam.ac.uk/`.

[2] `https://catalog.ldc.upenn.edu/LDC93S1`.

[3] `https://www.sharcnet.ca/help/index.php/Monk`.

[4] `https://www.sharcnet.ca/my/`.

[5] J.A. Arrowood and M.A. Clements. Using observation uncertainty in HMM decoding. In *INTERSPEECH*. ISCA, 2002.

[6] J. Baker, L. Deng, S. Khudanpur, C.H. Lee, J. Glass, and N. Morgan. Updated MINDS Report on Speech Recognition and Understanding, Part 2. *IEEE Signal Processing Magazine*, 26(4), July 2009.

[7] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[8] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.

[9] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Trans. Neural Networks*, 3(2):252–259, 1992.

[10] S.F. Boll. Suppression of Acoustic Noise in Speech Using Spectral Subtraction. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 27:113–120, 1979.

[11] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[12] G. Dahl, T. Sainath, and G. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP*, 2013.

[13] G. Dahl, D. Yu, L. Deng, and A. Acero. Large Vocabulary Continuous Speech Recognition With context-dependent DBN-HMMS. In *Proc. ICASSP, Prague*. IEEE, May 2011.

[14] G. Dahl, D. Yu, L. Deng, and A. Acero. Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing (receiving 2013 IEEE SPS Best Paper Award)*, 20(1):30–42, January 2012.

[15] G.E. Dahl, M. Ranzato, A. Mohamed, and G. Hinton. Phone Recognition with the Mean-Covariance Restricted Boltzmann Machine. In *Advances in Neural Information Processing Systems 23*, pages 469–477, 2010.

[16] L. Deng. Three Classes of Deep Learning Architectures and Their Applications: A Tutorial Survey. *APSIPA Transactions on Signal and Information Processing*, 2012.

[17] L. Deng, A. Acero, M. Plumpe, and X. Huang. Large-Vocabulary Speech Recognition under Adverse Acoustic Environments,. In *Proc. Int. Conf. on Spoken Language Processing*, volume 3, October 2000.

[18] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *in Proc. Int. Conf. Acoust., Speech, Signal Process*, 2013.

[19] L. Deng, J. Li, J-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero. Recent Advances in Deep Learning for Speech Research at Microsoft. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), May 2013.

[20] L. Deng and D. Yu. *Deep Learning for Signal and Information Processing*. Microsoft, 2013.

[21] L. R. Bahl et al. Language-Model/Acoustic Channel Balance Mechanism,. 23(7B):3464–3465, 1980.

[22] M.A. Franzini, K.F. Lee, and A. Waibel. Connectionist Viterbi training: a new hybrid method for continuous speech recognition. *International Conference on Acoustics, Speech and Signal Processing*, pages 425–428, 1990.

[23] P. Frasconi, M. Gori, and G. Soda. Recurrent networks for continuous speech recognition. *Computational Intelligence 90*, 1990.

[24] M. Gori, Y. Bengio, and R. De Mori. BPS: A learning algorithm for capturing the dynamic nature of speech. *Proceedings of the International Joint Conference on Neural Networks*, 2:417–423, 1989.

[25] A. Graves, A. Mohamed, and G. Hinton. Speech Recognition with Deep Recurrent Neural Networks. *ICASSP*, 2013.

[26] P. Haffner, M. Franzini, and A. Waibel. Integrating time alignment and neural networks for high Performance continuous speech recognition. *International Conference on Acoustics, Speech and Signal Processing*, pages 105–108, 1991.

[27] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean. Multilingual Acoustic Models using Distributed Deep Neural Networks. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), May 2013.

[28] H. Hermansky and N. Morgan. RASTA processing of speech. *IEEE Transactions on Speech and Acoustics*, 2(4):587–589, October 1994.

[29] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn. Compensation for the effect of the communication channel in auditory-like analysis of speech (RASTA-PLP). In *EUROSPEECH*, page 13671370. ISCA, 1991.

[30] G. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Comput.*, 14(8):1771–1800, 2002.

[31] G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2012.

[32] G. Hinton, L. Deng, D. Yu, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, G. Dahl, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012.

[33] G. Hinton, S. Osindero, and Y-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comput.*, 18(7), 2006.

[34] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504 – 507, 2006.

[35] G. Hinton and R. Salakhutdinov. Discovering Binary Codes for Documents by Learning Deep Generative Models. In *Topics in Cognitive Science*, pages 1–18, 2010.

[36] G. Hinton, N. Srivastava, A. Krizhevsky, I.Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. 2012.

[37] J-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. Cross-Language Knowledge Transfer Using Multilingual Deep Neural Network With Shared Hidden Layers. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013.

[38] X. Huang, A. Alex, and H-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[39] A. Itamar, R. Derek, and T. P. Karnowski. Deep Machine Learning - A New Frontier in Artificial Intelligence Research. *IEEE Computational Intelligence Mag.*, 5(4):13–18, 2010.

[40] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke. Application Of Pretrained Deep Neural Networks To Large Vocabulary Speech Recognition. In *Proceedings of Interspeech 2012*, 2012.

[41] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA, USA, 1997.

[42] H. Jiang and L. Deng. A robust compensation strategy against extraneous acoustic variations in spontaneous speech recognition. *IEEE Transactions on Speech & Audio Processing*, 10(1):9–17, January 2002.

[43] B-H. Juang and S. Furui. Automatic Recognition and Understanding of Spoken Language - a First Step Toward Natural Human-Machine Communication. *Proceedings of The IEEE*, 88:1142–1165, 2000.

[44] B. H. Juang and L.R. Rabiner. Hidden Markov Models for Speech Recognition. *Technometrics*, 33:251–272, August 1991.

[45] B. H. Juang and L.R. Rabiner. Automatic Speech Recognition - A Brief History of the Technology Development. *Elsevier Encyclopedia of Language and Linguistics*, 2005.

[46] J.C. Junqua and J.P. Haton. *Robustness in Automatic Speech Secognition: Fundamentals and Applications*. Kluwer international series in engineering and computer science. Kluwer Academic Publishers, 1996.

[47] F.O. Karray and C. De Silva. *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications.* Addison Wesley, 2004.

[48] H-S. Le, I. Oparin, A. Allauzen, J-L. Gauvain, and F. Yvon. Structured output layer neural network language models for speech recognition. *IEEE Transactions on Audio, Speech & Language Processing*, 21(1):195–204, 2013.

[49] J. Li, L. Deng, Y. Gong, and R. Haeb-Umbach. An Overview of Noise-Robust Automatic Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):745 – 777, April 2014.

[50] J. Li, D. Yu, J-T. Huang, and Y. Gong. Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM. In *IEEE Workshop on Spoken Language Technology*, 2012.

[51] J. S. Lim and A. V. Oppenheim. Enhancement and bandwidth compression of noisy speech. *Proceedings of the IEEE*, 67(12):15861604, December 1979.

[52] R. Lippmann, E. Martin, and D. Paul. Multi-style training for robust isolated-word speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 705–708. IEEE, April 1987.

[53] B. T. Lowerre. *The Harpy Speech Recognition System.* PhD thesis, Carnegie Mellon University, 1976.

[54] L. Lu. *Subspace Gaussian Mixture Models For Automatic Speech Recognition.* PhD thesis, Institute of Language, Cognition and Computation, School of Informatics, Unversity of Edinburgh, 2013.

[55] A. Mohamed, G. Dahl, and G. Hinton. Deep Belief Networks for Phone Recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[56] A. Mohamed, G. Dahl, and G. Hinton. Acoustic Modeling Using Deep Belief Networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):14 –22, jan. 2012.

[57] A. Mohamed and G. Hinton. Phone recognition using Restricted Boltzmann Machines. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP, Dallas, Texas, USA*, pages 4354–4357, 2010.

[58] A. Mohamed, G. Hinton, and G. Penn. Understanding How Deep Belief Networks Perform Acoustic Modelling. In *ICASSP*, 2012.

[59] A. Mohamed, T. N. Sainath, G. E. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny. Deep Belief Networks using discriminative features for phone recognition. In *ICASSP*, pages 5060–5063, 2011.

[60] A. Mohamed, D. Yu, and L. Deng. Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition. In *Interspeech 2010*. International Speech Communication Association, September 2010.

[61] N. Morgan and H. Bourland. Continuous speech recognition using multilayer perceptrons with Hidden Markov Models. *International Conference on Acoustics, Speech and Signal Processing*, pages 413–416, 1990.

[62] V. Nair and G. Hinton. 3-D Object Recognition with Deep Belief Nets. In *NIPS*, pages 1339–1347, 2009.

[63] T.F. Quatieri. *Discrete-Time Speech Signal Processing : Principles and Practice*. Prentice-Hall signal processing series. Prentice Hall PTR, Upper Saddle River, N.J., London, 2001.

[64] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[65] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. In *Advances in Neural Information Processing Systems 19*, pages 1137–1144. MIT Press, 2007.

[66] C. Ris, J. Hennebert, H. Bourlard, S. Renals, and N. Morgan. Estimation Of Global Posteriors And Forward-Backward Training Of Hybrid Hmm/ann Systems. In *in Proc. Europ. Conf. Speech Communication and Technology*, pages 1951–1954, 1997.

[67] T. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Trans. Neural Networks*, 5(2):298–305, 1994.

[68] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed. Making Deep Belief Networks effective for large vocabulary continuous speech recognition. In *ASRU*, pages 30–35, 2011.

[69] T. N. Sainath, B. Ramabhandran, M. Picheney, D. Nahamoo, and D. Kanevsky. Exemplar-based sparse representation features: From TIMIT to LVCSR. *IEEE Trans. Audio Speech Lang. Processing*, 19(8):2598–2613, May 2011.

[70] F. Seide, G. Li, and D. Yu. Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. In *Interspeech 2011*. International Speech Communication Association, August 2011.

[71] R. Stern, A. Acero, F. H. Liu, and Y. Ohshima. Signal Processing for Robust Speech Recognition. In *Automatic speech and speaker recognition : Advanced Topics*, page 357384. Kluwer Academic Publishers, 1996.

[72] V. Stouten, H. Van Hamme, and P. Wambacq. Model-based feature enhancement with uncertainty decoding for noise robust ASR. *Speech Communication*, 48(11):1502–1514, 2006.

[73] J. Tebelskis, A. Waibel, B. Petek, and O. Schmidbauer. Continuous speech recognition using linked predictive networks. *Advances in Neural Information Processing Systems*, 1:61–64, 1991.

[74] A. Testolin, I. Stoianov, M. D. F. D. Grazia, and M. Zorzi. Deep unsupervised learning on a Desktop PC: a primer for cognitive scientists. *Frontiers in Psychology*, 2013.

[75] E. Trentin and M. Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*, 37:91–126, 2001.

[76] A. Waibel. Modular construction of time-delay neural networks for speech recognition. *Neural Compution*, 1:39–46, 1989.

[77] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions. Acoustics. Speech Signal Processing.*, 37(3):328–339, 1989.

[78] F. Wessel, R. Schlter, K. Macherey, and H. Ney. Confidence Measures for Large Vocabulary Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 9:288–298, 2001.

[79] G. Williams and S. Renals. Confidence measures for hybrid HMM/ANN speech recognition. In *EUROSPEECH*, 1997.

[80] P.C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech & Language*, 16(1):25–47, 2002.

[81] X.Huang and L. Deng. An Overview of Modern Speech Recognition. In *Handbook of Natural Language Processing, Second Edition, Chapter 15 (ISBN: 1420085921)*, pages 339–366. Chapman & Hall/CRC, 2010.

[82] H. Xiaodong and L. Deng. *Discriminative Learning For Speech Recognition: Theory and Practice*. Morgan & Claypool, October 2008.

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

[84] D. Yu, L. Deng, and G. Dahl. Roles of Pre-Training and Fine-Tuning in Context-Dependent DBN-HMMs for Real-World Speech Recognition. In *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, December 2010.