MULTI-TASK LEARNING DEEP NEURAL NETWORKS FOR AUTOMATIC SPEECH RECOGNITION

by

DONGPENG CHEN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

PhD Thesis, August 2015, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

DONGPENG CHEN

MULTI-TASK LEARNING DEEP NEURAL NETWORKS FOR AUTOMATIC SPEECH RECOGNITION

by

DONGPENG CHEN

This is to certify that I have examined the above Ph.D. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

PROF. BRIAN MAK, THESIS SUPERVISOR

PROF. SHING-CHI CHEUNG, ACTING HEAD OF DEPARTMENT

Department of Computer Science and Engineering

24 August 2015

ACKNOWLEDGMENTS

First of all, I would like to express my sincere thanks to Prof. Brian Mak, who patiently advises and supports me during my PhD study. His open and optimistic attitude towards research benefits me a lot. I learnt a lot from the way he analyzes problems, and also his great presentation and writing skills. The experience of working with him will influence my whole life.

I would like to thank members of my PhD thesis examination committee, Prof. James Kwok, Prof. Raymond Wong, Prof. Chi-Ying Tsui and Prof. Pak-Chung Ching. I would also like to thank Prof. Dit-Yan Yeung and Prof. Nevin Zhang for serving as the committee members of my thesis proposal defense.

I am grateful to my colleagues, Guoli Ye and Tom Ko, who helped me start my study and research in HKUST. I also want to thank Cheung-Chi Leung, Sunil Sivadas and other colleagues in I2R, Singapore for their useful instruction when I was intern there.

My special thanks go to my parents and my younger brother, for their unconditional support. I have to thank my girl friend, for always accompanying and encouraging me in the past years. Without them I cannot accomplish my research works.

LIST OF PUBLICATIONS

1. **Dongpeng Chen**, Brian Mak. "Distinct Triphone Acoustic Modeling", submitted to IEEE Transactions on Audio, Speech and Language Processing.

2. **Dongpeng Chen**, Brian Mak. "Distinct Triphone Acoustic Modeling By Multi-task Learning Deep Nerual Networks", accepted by Interspeech 2015.

3. **Dongpeng Chen**, Brian Mak. "Multi-task Learning Of Deep Neural Network For Low-resource Speech Recognition", IEEE Transactions on Audio, Speech and Language Processing, vol 23, no. 7, pp. 1172–183, 2015.

4. **Dongpeng Chen**, Brian Mak, Sunil Sivadas. "Joint Sequence Training Of Phone And Grapheme Acoustic Model Based On Multi-task Learning Deep Neural Networks", in Proceedings of Interspeech, 2014, pp. 1083–1087.

5. Tom Ko, Brian Mak, **Dongpeng Chen**. "Modeling Inter-cluster And Intra-cluster Discrimination Among Triphones", International Symposium on Chinese Spoken Language Processing, 2014, pp. 103–107.

6. **Dongpeng Chen**, Brian Mak, Cheung-Chi Leung, Sunil Sivadas. "Joint Acoustic Modeling Of Triphones And Trigraphemes By Multi-task Learning Deep Neural Networks For Low-resource Speech Recognition", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2014, pp. 5592–5596.

7. **Dongpeng Chen**, Brian Mak. "Distinct Triphone Modeling By Reference Model Weighting", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7150–7154.

8. Guoli YE, **Dongpeng Chen**, Brian Mak. "Transition Probabilities Are More Important Than We Once Thought" , in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2012, pp. 4809–4812.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Multi-task learning (MTL) learns multiple tasks together and improves the performance of all the tasks by exploiting extra information from each other with the shared internal representation. Additional related secondary task(s) acts as regularizer(s) to help improve the generalization performance of each singly learning task; the effect is more prominent when the amount of training data is relatively small. Recently, deep neural network (DNN) is widely utilized for acoustic modeling in automatic speech recognition (ASR). The hidden layers of a DNN provide an ideal internal representation for the shared knowledge. The main contribution of this thesis is the proposal of three methods of that apply MTL to DNN for acoustic modeling by exploiting extra information from related tasks, meanwhile imposing the guideline that the secondary tasks should not require additional language resources. The guideline is important when language resources are limited.

In the first method, phone and grapheme acoustic models are trained together within the same deep neural network. The extra information is the phone-to-grapheme mappings, which is confirmed by analysis and visualization of the implicit phone-to-grapheme correlation matrix computed from the model parameters. The training convergence curve also shows that MTL training generalizes better to unseen data than common single task learning does. Moreover, two extensions are proposed to further improve the performance.

State tying, to some extent, relieves the data scarcity problem in context-dependent acoustic modeling. However, quantization errors are inevitably introduced. The second MTL method in this thesis aims at robust modeling of a large set of *distinct* context-dependent acoustic units. More specifically, distinct triphone states are trained with a smaller set of tied states, benefiting from better inductive bias to reach a better optimum. In return, they embed more contextual information into the hidden layers of the MTL-DNN acoustic models.

Our last method works in a multi-lingual setting when data of multiple languages are available. Multi-lingual acoustic modeling is improved by learning a universal

phone set (UPS) modeling task together with language-specific triphones modeling tasks to help implicitly map the phones of multiple languages to each other.

MTL methods were proved to be effective on a board range of data sets. The contributions of this thesis include the three proposed MTL methods, and the heuristic guidelines we impose to find helpful secondary tasks. With the successful explorations, we hope to stimulate more interest of MTL in improving ASR, and our results show that it is promising for wider applications.

CHAPTER 1

INTRODUCTION

Since the emergence of human civilization, speech is indispensable to human-human communication. It is also considered as an important communication method in human-computer communication.

Research on automatic speech recognition (ASR) has been very active for more than six decades and has made tremendous progress. At the beginning, speech recognizers were only able to recognize a small number of isolated words spoken in a quiet environment. In 1980s, the use of hidden Markov model with Gaussian mixture model as state output distribution (GMM-HMM) for acoustic modeling makes speech recognizers capable of conducting large-vocabulary continuous speech recognition. Thanks to its ease of training and decoding, for the following twenty years, GMM-HMM was the mainstream acoustic model in ASR systems, and acoustic modeling research focused on improving GMM-HMM by better model structure or training algorithm. Significant works include state tying [2], discriminative training [3, 4, 5], and maximum likelihood linear transformation [6].

During the period dominated by GMM-HMMs, researchers also explored many other models for acoustic modeling such as high-density discrete HMM which uses a discrete distribution with large codebooks to model the state output distribution [7], hybrid artificial neural network (ANN) HMM [8] and the segment models [9, 10]. However, none of them can be shown to outperform GMM-HMM.

Development of ASR was slow and a bit boring until the second decade of the new century. The past five years saw the great success of deep learning architectures and techniques on many computer vision, language and speech learning tasks. Deep neural network (DNN) and its variants finally replaced GMM and nowadays hybrid DNN-HMM is used as the acoustic model in most ASR systems. The advancement can be attributed to the following factors:

- deep learning architectures and algorithms;

- evolution of general purpose graphical processing units (GPGPU);
- thousands of hours of well-transcribed training data, and far more unlabeled data from the crowd;
- the use of weighted finite state transducer in ASR decoder [11];
- mobile Internet and cloud computing;
- the great personal and commercial needs for speech recognition applications.

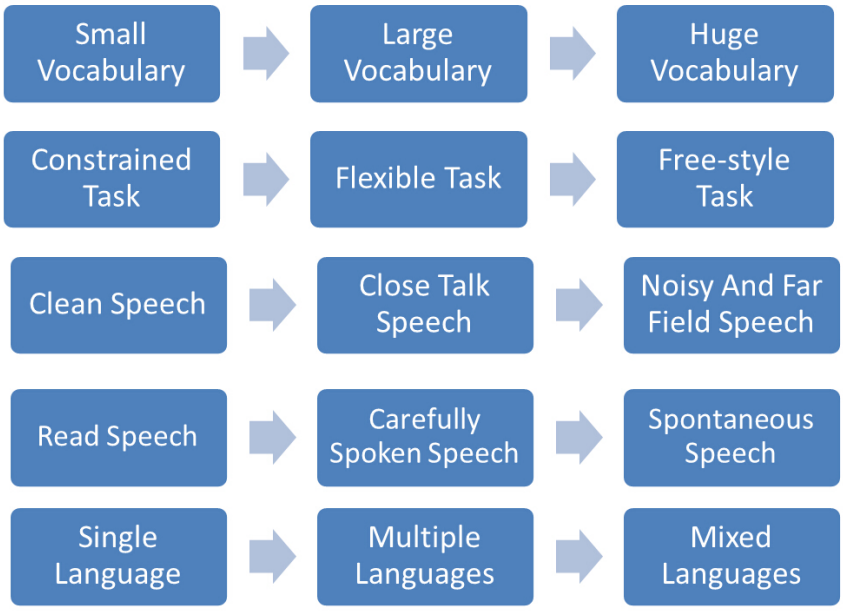

Figure 1.1: ASR problems.

Today the ASR techniques are mature enough for many real-world applications. However, many efforts still need to be paid to catch up with and surpass the speech recognition ability of human beings. [12] categorizes the subproblems that ASR addresses into different aspects and different difficulty levels, as is shown in Fig. 1.1. The authors point out that we are facing the problems in the right-most column: ASR on huge vocabulary, free-style task, noisy far-field speech, spontaneous speech and mixed languages. Research interests have moved to the following aspects of DNN-HMM ASR systems:

- parallelizing and accelerating the training and decoding process;
- speaker adaptation, noise robustness, etc. ;

- regularization methods, such as the dropout method [13]
- different deep learning architectures such as the deep convolutional neural network [14] and the deep recurrent neural network [15];

This thesis takes a multi-task learning approach, which can be regarded as a regularization method to improve state-of-the-art DNN-HMM acoustic model.

1.1 Why Multi-task Learning (MTL) for ASR?

For thousands of years, humans have been learning from the nature during. Even though in modern times, we are surrounded by artificial products, we can still see the signs of many inspiration from the nature on many industrial products. For example, most initial designs of airplanes and submarines were copied from birds and fishes — from their appearance to mechanism. Without the hints from nature, the human civilization would not be able to evolve so fast. How to learn from nature even becomes a complex science: bionics. It applies biological methods and systems observed in nature to the design of engineering systems.

In computer science, one of the most straightforward and influential imitations is the artificial neural network (ANN) [16]. Similar to biological neural network, ANN is composed of millions of neurons and their connections. Neurons can communicate with each other, and the connection weights between them can be trained to learn certain knowledge from the training data. More recently, people observe that biological brains use both shallow and deep circuits from brain anatomy [17]. Thus, an ANN was later enhanced by adding more hidden layers to form a deep neural network (DNN).

Table 1.1: Two real life examples of MTL.

Task	Object recognition	Typing English words and Chinese words by Pinyin
Shared Input	Pixels	Words to type
Shared Internal representation	Shapes or textures	Keyboard layout
Output	Target object seen?	Finger movements to type English or Chinese words

Multi-task learning (MTL) [18] is a machine learning method that learns multiple *related* tasks together to better learn the primary task we aim to improve. The idea of MTL is also motivated from human behavior on learning real tasks. Humans tackle a new task with the prior knowledge gain from previous similar learning tasks. Moreover, humans have the capability to learn multiple tasks simultaneously to achieve better learning effect. Table 1.1 lists the shared input features, internal representations and outputs for the two MTL examples:

- Recognition of multiple objects are related tasks. Children learn to recognize all objects at the same time by the shapes or textures of the objects in an MTL manner. They don't learn one by one.
- Typing words of different languages by a keyboard are related tasks. To type Chinese characters by the Pinyin input method, people need to learn the keyboard layout first, which is the same as that for typing English.

As a real life example that is more related to automatic speech recognition, humans usually learn a language by reading, listening and speaking it at the same time. Learning multiple language skills together accelerates the process of mastering a foreign language, while languages without a formal writing system are usually much harder to learn for foreigners since the trick of MTL cannot work.

Applying these observations from real life to engineering is natural. In machine learning, multi-task learning is known to be particularly effective when training data is rare. Data scarcity is one of the largest obstacles for the development of human language technologies, especially for low-resourced languages with only a few hours of training data.

Actually, MTL has been applied successfully in many speech, language, image and vision tasks with the use of neural network (NN) because the hidden layers of an NN naturally capture learned knowledge that can be readily transferred or shared across multiple tasks. For example, [19] applies MTL on a single convolutional neural network to produce state-of-the-art performance for several language processing predictions; [20] improves intent classification in goal-oriented human-machine spoken dialog systems which is particularly successful when the amount of labeled training data is limited; in [21], the MTL approach is used to perform multi-label learning in an

image annotation application, which is exactly motivated from the object recognition example given above.

With the recent success of DNN for acoustic modeling in ASR, we believe MTL may further improve DNN training. Multi-task learning deep neural network (MTL-DNN) is essentially an imitation of human brain, where most neurons are functioning for all basic human abilities, while some are exclusive for particular behaviors. There are many related secondary tasks that are promising to improve the primary speech recognition task. Some of them have been proved to be helpful. For example, in [22], phone and state context classification tasks are trained together to benefit phone recognition. Therefore, there are plenty of reasons to believe MTL can be a helpful way to improve ASR performance.

1.2 How to Apply MTL to ASR?

Before applying MTL to any task, one needs to answer the question of how to choose the secondary tasks to benefit the training of the primary task.

The main contribution of this thesis is answering the question in ASR in the context of acoustic modeling. Firstly, we have the following guidelines on choosing the secondary task(s):

(a) It should be positively related to the primary phonetic modeling task so that they can share an internal representation, which generalizes better to unseen observations.

(b) It shares the same set of inputs with the primary task;

(c) It should not require extra language resources;

(d) It should be easy to set up and train.

Under these guidelines, we choose and investigate three kinds of secondary tasks through experiments on different recognition tasks. They are acoustic modeling tasks of

- graphemes: basic units in a writing system. Writing systems preserve speech over time and distance. MTL training of phone and grapheme acoustic units is expected to exploit the extra phone-to-grapheme mapping information from the writing systems.

- distinct triphones: phonetic units that exhibit finer contextual information. Joint training with distinct triphone states will provide more contextual information to the tied-state acoustic modeling task, while keeping the estimation robust.

- universal phones: a set of common phones that can be used to describe the speech sounds in multiple languages. Universal phone set of multiple languages is usually derived from International Phonetic Alphabet based on the linguistic knowledge of experts. We believe utilizing the multi-lingual phone mapping is able to benefit current multi-lingual acoustic modeling.

1.3 Thesis Outline

The thesis is organized as follows:

In Chapter 2, we present an overview of automatic speech recognition (ASR) and state-of-the-art ASR systems together with details of DNN-HMM that are used for acoustic modeling. The data scarcity problem in ASR will also be described.

In Chapter 3, a literature review of both theoretical and experimental works on multi-task learning is given. We also expound our MTL formula in the thesis together with the structure and the objective function of MTL-DNN.

In Chapter 4, the first proposed method is illustrated under a mono-lingual ASR setting. A phone acoustic modeling task is estimated with a grapheme acoustic modeling task in a DNN acoustic model, sharing part of the DNN parameters. It does not need extra language resources like explicit phone-to-grapheme mapping, which is usually not easy to obtain.

In Chapter 5, to model distinct triphones and reduce quantization errors brought by state tying, our second method estimates a large group of distinct triphone states with a smaller set of tied states in an MTL-DNN. Again the parameters in the hidden

layers of the MTL-DNN are shared by the two tasks. In this way, the estimation of the distinct triphones is more robust even if they do not have sufficient training data.

In Chapter 6, when multi-lingual data are available, we further model the implicit multi-lingual phone-to-phone mapping by an additional *universal phone set* (UPS) [23] modeling task in an MTL-DNN to benefit the acoustic modeling of all languages. The method can be combined with the first method to gain further improvement.

Finally, in the last chapter we summarize our contributions and findings in this thesis. Furthermore, we look into various prospective future works, expecting that MTL will benefit ASR more.

CHAPTER 2

REVIEW OF AUTOMATIC SPEECH RECOGNITION

This chapter aims to provide the reader the background knowledge of automatic speech recognition (ASR). The definition of ASR, and important components in ASR system are illustrated. More importantly, we describe details of how deep neural network is adopted for acoustic modeling in a hybrid system with a hidden Markov model.

2.1 Automatic Speech Recognition

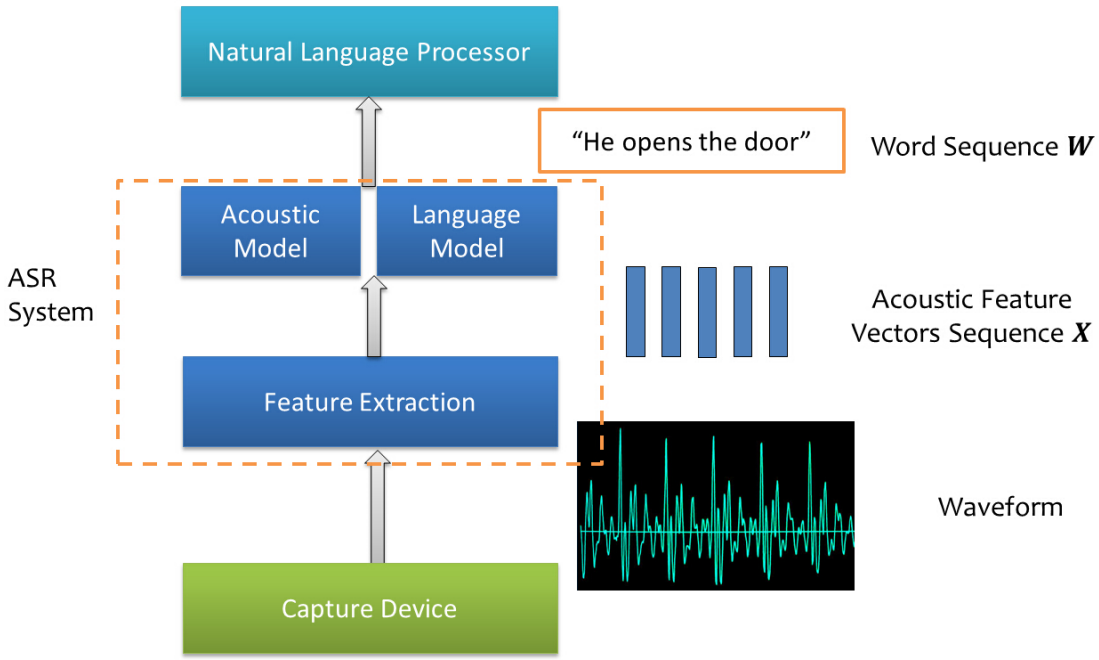

Figure 2.1: Overview of an ASR system.

The aim of automatic speech recognition, also known as computer speech recognition or speech to text, is to find the corresponding word sequences given a speech signal using computing devices. Generally it can be viewed as a machine learning and classification problem, or a sequential labeling task in which we are to learn acoustic patterns from speech training data and figure out the unknown word labels of new

speech signals. Given a sequence of acoustic feature vectors, the aim is to discover the unobserved word sequence behind them.

From the statistical point of view, the ASR task can be rephrased as: given a sequence of acoustic vectors $\mathbf{X}$, find the word sequence $\hat{\mathbf{W}} = w_1, w_2, \ldots, w_M$ that maximizes the posterior probability $P(\mathbf{W}|\mathbf{X})$. Applying the Bayes's rule, we have:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{X}) = \arg\max_{\mathbf{W}} \frac{P(\mathbf{W})P(\mathbf{X}|\mathbf{W})}{P(\mathbf{X})}$$

Since $P(\mathbf{X})$ is invariant of $\mathbf{W}$, we have:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X}|\mathbf{W}) \tag{2.1}$$

In Eq. (2.1), $P(\mathbf{X}|\mathbf{W})$ and $P(\mathbf{W})$ have the following meanings in a speech recognition system:

- $P(\mathbf{W})$ is called the language model, which represents the prior probability of the word sequence $\mathbf{W}$ independent of the observed signal sequence $\mathbf{X}$.
- $P(\mathbf{X}|\mathbf{W})$ is called the acoustic model, which represents the probability of observing the speech feature sequence $\mathbf{X}$ given the word sequence $\mathbf{W}$.

Fig. 2.1 shows the overview of an ASR system. Acoustic feature vectors are extracted from the speech waveform, and then decoding is performed to obtain the word sequence given the acoustic model and language model.

2.2 Language Model

Knowledge of a language is important for recognizing and understanding natural speech of the language. Linguistic knowledge is usually embedded in a language model (LM), which is a statistical model that assigns a probability to a sequence of m words $P(w_1, \ldots, w_m)$ by means of a probability distribution. Language modeling is used in many natural language processing applications such as speech recognition, machine translation, part-of-speech tagging, parsing and information retrieval.

In speech recognition, such a model tries to capture the properties of a language, and to predict the next word in a speech sequence. Generally, when evaluating the

probability of a word sequence given an utterance, it gives a probability of the word sequence, which is combined with the acoustic score that will be introduced in the following section.

The most commonly used LM is the n-gram LM. It is assumed that the probability of the current word only depends on the most recent $n-1$ words. The model stores all the n-gram probabilities $P(w_i|w_{i-1}, w_{i-2}, ..., w_{i-n+1})$, and the probability of any word sequence can be computed by

$$P(\mathbf{W}) = P(w_1, w_2, ..., w_m) = \Pi_{i=1}^{m} P(w_i|w_{i-n+1}, ..., w_{i-1})$$

Generally, an n-gram language model is trained by counting the occurrences of all n-grams in a text data set. However, generally many n-gram items are infrequent or unseen in the training data. [24] proposed a back-off scheme named Katz smoothing to address this problem. If an n-tuple is not observed frequently enough in the training text, its probability will be backed off by another probability computed from the occurrence count of a shorter $(n-1)$-tuple context. Kneser-Ney smoothing [25] further improved language modeling by combining the lower-order distribution and higher-order distribution. The authors interpolated lower-order distribution for all words (not just for words that have zero counts) in the higher-order distribution.

2.3 Acoustic Model

Although there were knowledge-based acoustic models in the history, since the late 1980s, most speech recognition systems had turned to statistical acoustic models. Generally, hidden Markov model (HMM) is trained to model the likelihood $P(\mathbf{X}|\mathbf{W})$. In this section, the definition and assumptions of HMM are first presented, followed by an introduction to two popular approaches of modeling the state output probability distribution in HMM.

2.3.1 Hidden Markov Model (HMM)

For ease of description, let us define:
λ: an HMM model (normally means all the parameters in the model),
a_{ij}: the transition probability from state i to state j,

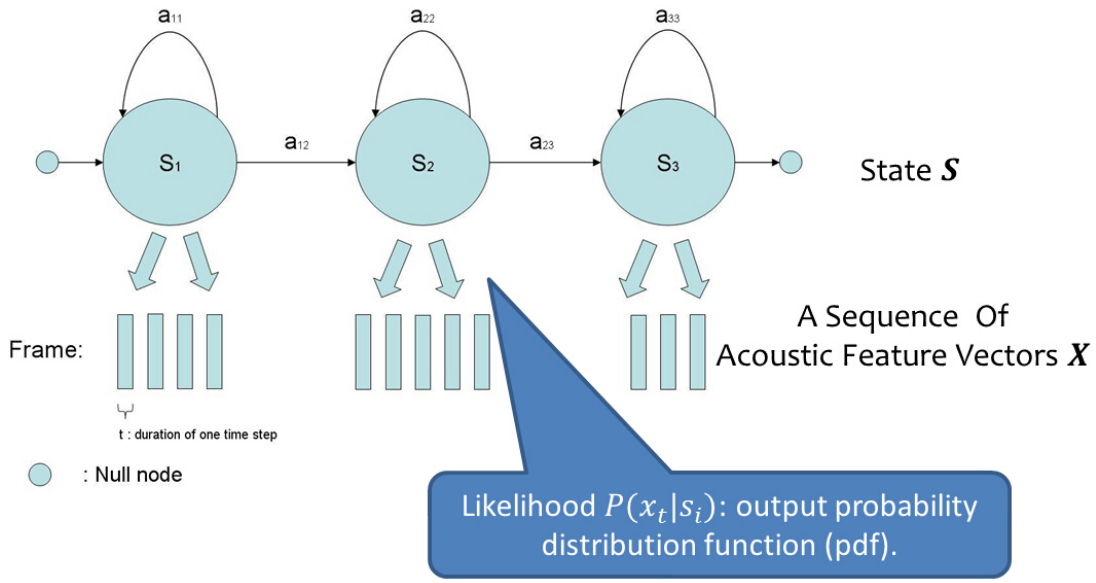

Figure 2.2: An example of left-to-right HMM with 3 states used for acoustic modeling.

J: the total number of states in the HMM λ,

T: the total number of frames in the observation vector sequence X.

$\mathbf{x}_t$: an observation vector at time t,

$\mathbf{X}$: a sequence of T observation vectors, $[x_1, x_2, \ldots, x_T]$,

s_t: the state at time t,

$\mathbf{S}$: the state sequence, $[s_1, s_2, \ldots, s_T]$.

The hidden Markov model (HMM) is a finite state machine in which the state sequence is not observable whereas only the observations generated by the model is directly visible. Transitions among the states are associated with a probability a_{ij} representing the transition probability from state i to state j. HMM is a generative statistical model. In each time step t, the system transits from a source state s_{t-1} to a destination state s_t and an observation vector $\mathbf{x}_t$ is emitted. The distribution of this emitted $\mathbf{x}_t$ is governed by the probability density function in the destination state. In the case of continuous-density HMM, each state is associated with a probability density function (PDF), which is crucial to the performance of an ASR system.

An example of HMM which is most commonly used to model a phone is shown in Fig. 2.2. It is a 3-state straightly left-to-right HMM in which only left-to-right

transitions are allowed in order to capture the sequential nature of speech. The first and the last nodes are null nodes, they are non-emitting states which will not generate any observations and are used to indicate the entry and exit states. This specific structure makes it easy to connect one HMM with another HMM to form a longer HMM. For example, several phone HMMs may connect together to form a bigger phonetic or linguistic unit such as a syllable, a word or even a sentence.

There are three major issues in hidden Markov modeling:

- **The Evaluation issue** : As an HMM is a generative model, any sequence of observations can be generated by an HMM. Given the HMM parameters λ, it is possible to determine the probability $P(\mathbf{X}|\lambda)$ that a particular sequence of observation vector $\mathbf{X}$ is generated by the model. In this case, the model parameters λ and the observation vectors $\mathbf{X}$ are the inputs, and the corresponding probability is the output.

- **The Training issue** : From a training/learning perspective, the sequence of observation vectors $\mathbf{X}$ is given to train the model parameters λ which are unknown. The observed data $\mathbf{X}$ are the inputs, and the estimated model parameters λ are the outputs.

- **The Decoding issue** : In a decoding process, the model parameters λ and the sequence of observation vector $\mathbf{X}$ is given whereas the sequence of states $\mathbf{S}$ is unknown. The goal is to look for the most likely sequence of underlying states $\mathbf{S}$ which maximizes $P(\mathbf{S}|\mathbf{X}, \lambda)$. In this case, the model λ and the observation vectors $\mathbf{X}$ are the inputs, and the decoded sequence of states $\mathbf{S}$ is the output.

ASR systems are based on the first-order HMM, which has the following assumptions:

- **The first order Markov assumption**: It is assumed that in first-order HMMs the transition probabilities from one state to the next state only depend only on the current state and not on the past state history. Given the past k states,

$$P(s_{t+1} = j|s_t = i_1, s_{t-1} = i_2, \ldots, s_{t-k+1} = i_k) = P(s_{t+1} = j|s_t = i_1), \quad (2.2)$$

where $1 \leq i_1, i_2, \ldots, i_k, j \leq J$.

On the other hand, the transition probabilities of a k^{th}-order HMM depend on the past k states. The first-order Markov assumption enables recognizers to perform fast decoding on large decoding network using dynamic programming Viterbi algorithm, with effective path pruning.

- **The output independence assumption**: It is assumed that given its emitting state, an observation vector is conditionally independent of the previous vectors as well as the neighboring states. Hence, we have

$$P(\mathbf{X}|\mathbf{S}, \lambda) = \prod_{t=1}^{T} P(\mathbf{x}_t|s_t, \lambda). \tag{2.3}$$

If the states are stationary, the observations in a given state are assumed to be independently and identically distributed (i.i.d.).

2.3.2 Gaussian Mixture Model (GMM)

For a long time, the probability density functions (PDF) associated with each state in HMM are commonly estimated as mixtures of Gaussian densities with diagonal covariances due to its simplicity and trainability. The PDF function is written as a weighted sum of the probabilities of an observation vector $\mathbf{x}$ given the Gaussian distributions:

$$P(\mathbf{x}|s) = \sum_i w_i \mathcal{N}(\mathbf{x}; \mathbf{m}_i, \mathbf{v}_i) \tag{2.4}$$

where w_i, $\mathbf{m}_i$, $\mathbf{v}_i$ are the weight, mean and covariance of the i-th Gaussian component.

Hidden Markov model using Gaussian mixture model (GMM-HMM) are usually estimated by the expectation-maximization (EM) algorithm which maximizes the likelihood of the observations sequences given the label sequences $P(\mathbf{X}|\mathbf{W})$ for all utterances in the training data set:

$$F_{MLE} = \sum_{n=1}^{N} logP(\mathbf{X}_n|\mathbf{W}) \tag{2.5}$$

If the HMM model assumptions always match the real speech statistics, and an infinite training set was used, MLE would be enough to ensure speech recognition accuracy. However, the assumptions are not true and the amount of training data is always

limited. When it became hard to get more improvement from the conventional MLE scheme, people start to explore discriminative training on HMMs. Unlike MLE, discriminative training also considers other possible competing hypotheses during HMM training and tries to reduce the probabilities of incorrect hypotheses. In other words, it attempts to improve the probability of a correct hypothesis while keeping those of incorrect hypotheses low.

Denote $\mathbf{X}_n$ and $\mathbf{W}_n$ as the observation sequence and the truth transcription of the n-th utterances. There are two popular discriminative training criteria[4]:

- **Maximum Mutual Information (MMI)** [26] : the MMI criterion is to maximize the mutual information between the training word sequences and the observation sequences during HMM training:

$$F_{MMI} = \sum_{n=1}^{N} \log P(\mathbf{W}_n|\mathbf{X}_n) = \sum_{n=1}^{N} \log \frac{P(\mathbf{X}_n|\mathbf{W}_n)P(\mathbf{W}_n)}{\sum_{\mathbf{W}'} P(\mathbf{X}_n|\mathbf{W}')P(\mathbf{W}')} \tag{2.6}$$

  where $\mathbf{W}'$ is a hypothesis word sequence.

- **Minimum Phone Error (MPE)** [5]: the MPE criterion is to minimize the phone errors on training utterances by optimizing:

$$F_{MPE} = \sum_{n=1}^{N} \frac{\sum_{\mathbf{W}'} P(\mathbf{W}'|\mathbf{X}_n)A(\mathbf{W}', \mathbf{W}_n)}{\sum_{\mathbf{W}'} P(\mathbf{W}'|\mathbf{X}_n)} \tag{2.7}$$

  where the function $A(\mathbf{W}', \mathbf{W}_n)$ measures the accuracy of hypothesis $\mathbf{W}'$ given transcription $\mathbf{W}_n$, and it equals the number of reference phones in $\mathbf{W}_n$ minus phone errors in hypothesis $\mathbf{W}'$. Intuitively MPE training maximizes the probabilities of hypotheses that have higher accuracy.

In 1990s and 2000s, most ASR systems employed GMM as state output distribution function in their HMMs. However, GMM is inefficient to model data distributed on a non-linear manifold such as a simple sphere — even for such simple distribution, a large number of components are needed! During the period, researchers kept proposing many new models beyond GMM-HMM, but none of them succeeded until the resurgence of deep neural networks.

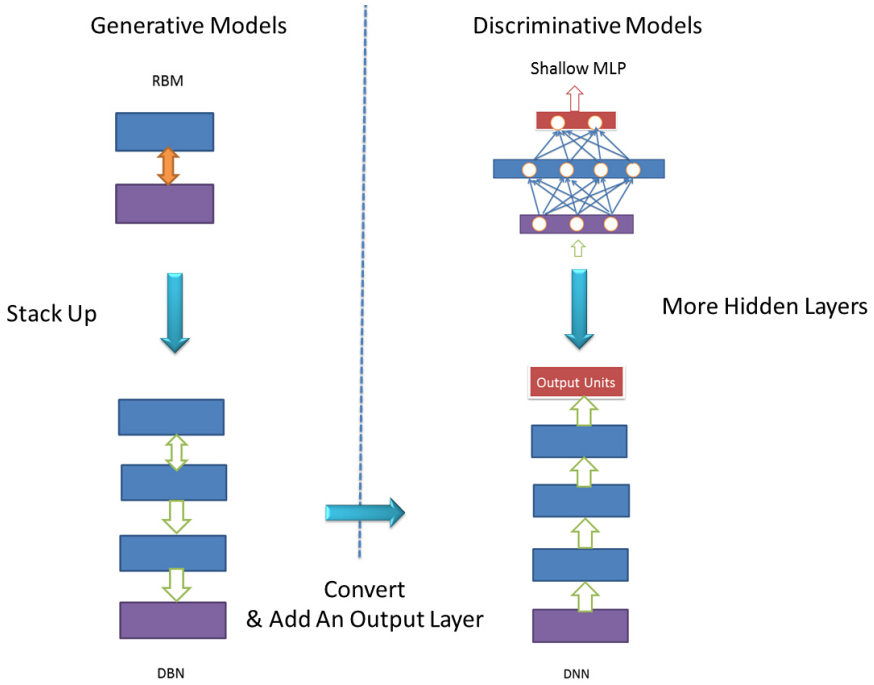

Figure 2.3: Relationship between the four ANNs in this section.

2.3.3 Deep Neural Network (DNN)

To better illustrate the definition, model structure and training algorithm of deep neural network, we first describe three other popular graphical models. They are multilayer perceptron, restricted Boltzman Machine and deep belief network. All of them are artificial neural networks (ANNs), which are statistical learning model motivated by biological neural networks. Fig. 2.3 demonstrates the relationship between the four models.

2.3.3.1 Multilayer perceptron

A multilayer perceptron (MLP) is a directed feedforward ANN mapping a set of input data to outputs by applying a series of operations. It is a discriminative model. As is shown in Fig. 2.4, a shlow MLP usually has an input layer, a hidden layer and an output layer, and in each layer there are a set of nodes. Nodes in neighboring layers are full connected, while nodes in the same layer do not connect with each other. Each node in the hidden and output layers is a neuron (or processing element) with a nonlinear activation function such as the sigmoid function.

The model parameters of an MLP are the connection weights between nodes and

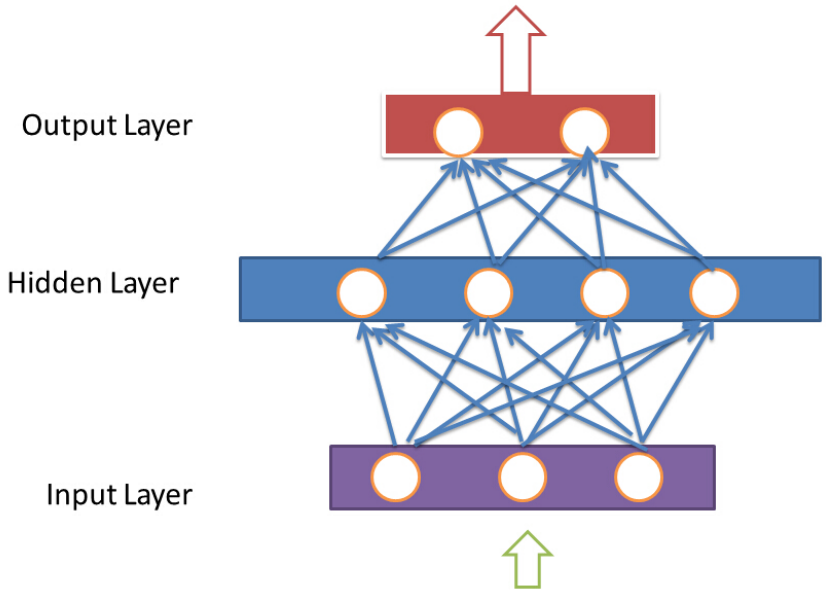

Figure 2.4: Multilayer perceptron.

learning MLP is done by adjusting the connection weights. Generally the learning objective is the minimum cross entropy (MCE) between the predictions $P(s_i|\mathbf{x})$ and the desired target d_i of each input frame $\mathbf{x}$

$$\mathcal{E} = \min \sum_{\mathbf{x}\in\mathcal{D}} \sum_{i} d_i \log P(s_i|\mathbf{x}) \tag{2.8}$$

or the least-squares error:

$$\mathcal{E} = \min \sum_{\mathbf{x}\in\mathcal{D}} \sum_{i} \frac{1}{2} ||d_i - P(s_i|\mathbf{x})||_2^2 \tag{2.9}$$

where $\mathcal{D}$ is the training set, d_i is the i-th dimension of $\mathbf{d}$, which is set to 1 if $\mathbf{x}$ is labeled as s_i and 0 otherwise.

In practice, stochastic gradient descent is usually conducted in the mini-batch mode to accelerate training. Let us denote o_i as the output of node i in the feed-forward step, δ_j as the error propagated back from node j, and w_{ij} as the connection weight between node i and j. At the backpropagation step we have $\delta_j = \frac{\partial \mathcal{E}}{\partial o_i W_{ij}}$. The weight update over a mini-batch of data is computed as the negative product of the gradient multiplied by the learning rate η:

$$\delta W_{ji} = -\eta \frac{\partial \mathcal{E}}{\partial W_{ij}} = -\eta o_i \frac{\partial \mathcal{E}}{\partial o_i W_{ij}} = -\eta o_i \delta_j \tag{2.10}$$

Training will go on for multiple epochs with reducing learning rate until the classification performance on some development data set reaches its optimum.

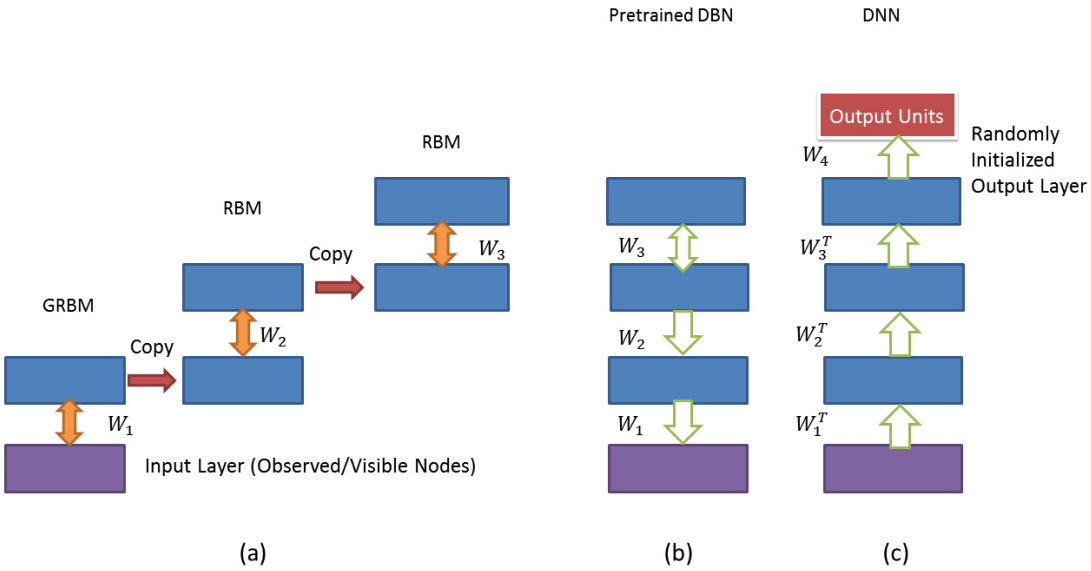

Figure 2.5: Pre-training of DBN by training RBMs, for better initialization of DNN training.

2.3.3.2 Restricted Boltzman machine

RBM is an undirected bipartite graph consisting of two disjoint groups of nodes: visible (input) nodes and hidden (output) nodes. Connections are restricted so that a visible node does not connect to other visible nodes, and a hidden node does not connect to other hidden nodes. Different from an MLP, it is a generative model that models the joint probability of the inputs and outputs. RBM can be effectively trained by minimizing the contrastive divergence [28] in an unsupervised manner. Let us denote the binary visible nodes i and binary hidden nodes j as v_i and h_j, the weight matrix between hidden nodes and visible nodes as $\mathbf{W}$, and the biases for visible and hidden nodes as a_i and b_j respectively . The energy function of a binary RBM is defined as

$$E(v,h) = -\sum_{ij} v_i W_{ij} h_j - \sum_{i \in visible} a_i v_i - \sum_{h \in hidden} b_j h_j \tag{2.11}$$

In the case of real-valued input data, the energy function of a Gaussian-Bernoulli

RBM (GRBM) is

$$E(v,h) = \sum_{i \in visible} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j \in hidden} b_j h_j - \sum_i \frac{v_i}{\sigma_i} \sum_j h_j W_{ij} \tag{2.12}$$

where the input visible node i is assumed to have Gaussian distribution with σ_i as the standard deviation and a_i as the mean. GRBM is used at the bottom of the network, and its inputs are acoustic feature frames.

The objective function of unsupervised training of a single-layer RBM is to minimize the negative log-likelihood of the joint probability distribution of all nodes

$$\min \sum_{\boldsymbol{h}} -\log P(\mathbf{v}, \boldsymbol{h}) \tag{2.13}$$

w.r.t $\mathbf{W}$, $\mathbf{a}$ and $\mathbf{b}$ where

$$P(\mathbf{v}, \boldsymbol{h}) = \frac{1}{Z} exp(-E(\mathbf{v}, \boldsymbol{h})) \tag{2.14}$$

and Z is a normalization factor on all possible $\mathbf{v}$ and $\boldsymbol{h}$: $\sum_{\mathbf{v},\boldsymbol{h}} exp(-E(\mathbf{v}, \boldsymbol{h}))$.

Since the RBM is a bipartite, it can be proved that $P(\boldsymbol{h}|\mathbf{v}) = \Pi_i P(h_i|\mathbf{v})$ and similarly $P(\mathbf{v}|\boldsymbol{h}) = \Pi_i P(v_i|\boldsymbol{h})$. Since $h \in \{0, 1\}$, we have

$$\begin{aligned} P(h_i = 1|\mathbf{v}) &= \frac{exp(-(b_i + \mathbf{v}^T \mathbf{W}_{*,i}) h_i)}{exp(-(b_i + \mathbf{v}^T \mathbf{W}_{*,i}) h_i) + exp(-(b_i + \mathbf{v}^T \mathbf{W}_{*,i})(1 - h_i))} \\ &= sigmoid(b_i + \mathbf{v}^T \mathbf{W}_{*,i}) \end{aligned}$$

Thus

$$P(\boldsymbol{h} = \mathbf{1}|\mathbf{v}) = sigmoid(\mathbf{b} + \mathbf{v}^T \mathbf{W}) \tag{2.15}$$

and similarly

$$P(\mathbf{v} = \mathbf{1}|\boldsymbol{h}) = sigmoid(\mathbf{a} + \boldsymbol{h}^T \mathbf{W}^T). \tag{2.16}$$

The form of Equation 2.15 allows us to use the weights of an RBM to initialize a feed-forward neural network with sigmoidal hidden units because the inference for RBM hidden units is equivalent to propagation in a feed-forward neural network.

RBM can be trained effectively by the simple one-step contrast divergence algorithm. [28] showed that the gradient of the connection weight w_{ij} between visible node

i and hidden node j is simply the difference between the training data and data sampled from the model. The update formula is

$$\Delta w_{ij} = \eta(< v_i h_j >_{data} - < v_i h_j >_{model}) \tag{2.17}$$

where $< . >$ means expectations of variables under the distribution specified by the subscript outside it. However, getting an unbiased sample of $< v_i h_j >_{model}$ is difficult. This is usually done by alternating Gibbs sampling for a long time. It starts from randomly picking the states of visible units. Then hidden units are updated by Equation 2.15. Next, visible units are updated by Equation 2.16. Such procedure is repeated until convergence is reached, thus a lot of computation is necessary to obtain an unbiased sample. In [28], the one-step contrast divergence algorithm approximates $< v_i h_j >_{model}$ by reconstructed data:

- Feed real-valued visible data $\mathbf{v}^{(1)}$ into the RBM to compute the probabilities that hidden variables are 1. We then obtain the values of binary hidden variables $\boldsymbol{h}^{(1)}$ by sampling from their probabilities;
- Based on Equation 2.16, the values of hidden variables in the opposite direction are used to compute the values of binary visible variables $\mathbf{v}^{(2)}$, which are the reconstructed visible data;
- Finally, the values of the hidden variables are updated again to obtain $\boldsymbol{h}^{(2)}$ according to Equation 2.15;
- $< \mathbf{v}^{(1)}\boldsymbol{h}^{(1)} >$ are used for $< v_i h_j >_{data}$, and $< \mathbf{v}^{(2)}\boldsymbol{h}^{(2)} >$ are used to approximate $< v_i h_j >_{model}$.

2.3.3.3 Deep belief network

Like RBM, a deep belief network (DBN) [28] is a generative graphical model for randomly generating visible data and modeling the joint distribution of all variables, but DBN is composed of multiple layers of hidden variables. An example of DBN is given in Fig. 2.5(b). Except the connection between the two topmost last layers is undirected (or bi-directed), other connections are directed and in the opposite direction as MLP. DBN can be regarded as composition of simple, unsupervised networks such as RBM.

DBN is usually trained by training RBMs layer by layer, and used as initialization for DNN training, which will be described later.

2.3.3.4 Deep neural network

A deep neural network (DNN) is simply a multilayer perceptron with many hidden layers, as shown in Fig. 2.5(c). Theoretically, the deep architecture is able to model highly non-linear functions and distribution of high dimensional data, but it is very difficult to train DNNs in the past. Firstly, error signals propagated back to bottom hidden layers diminish quickly, making it hard to train parameters in the bottom layers. Secondly, the computation-intensive large matrix operations in training and decoding of DNN make it hard to scale up to large vocabulary speech recognition tasks using thousands of hours of speech training data, and to be run in real time.

There was a resurgence of DNNs in recent years after Hinton et al. introduced a fast pre-training algorithm for a deep belief network. The fast advancement of graphic processing unit (GPU) parallel computing hardwares and techniques in recent years also greatly promotes the applications of DNN in various real-world machine learning tasks. With GPUs, large batch of matrix operations can be easily parallelized. DNNs have been proved to be very effective in many tasks of speech recognition [29], computer vision [30] and natural language processing [19]. More specifically, an DNN is used to replace the GMM to model the PDFs of HMM states in speech recognition, and it usually outperforms GMMs by a large margin.

Training of a DNN usually consists of the pre-training phase and fine-tuning (back-propagtion) phase.

- Pre-training of DBN:

  The aim of pre-training is to provide a better initialization point for subsequent DNN training. Hinton proposed initializing a DNN with a generative pre-trained deep belief network [28], which consists of repeated layers of RBM. As is shown in Fig. 2.5, after an RBM is trained, its connection weights will be fixed, and another new one is placed on top of it. The new one is trained by the output from the fixed ones below it. At the end, several RBMs are stacked together to form a DBN which is then converted to a DNN with the addition of an output layer that

is designed for each application. Note that in such conversion, the backward arcs between the last two hidden layers are removed, and other backward directed connections are reverted.

- Backpropagation:

  The supervised backpropagation step of DNN training is exactly the same as that of common MLP. Backpropagation is usually performed on the whole network to minimize the sum of cross-entropy (Equation 2.8) or squared error (Equation 2.9).

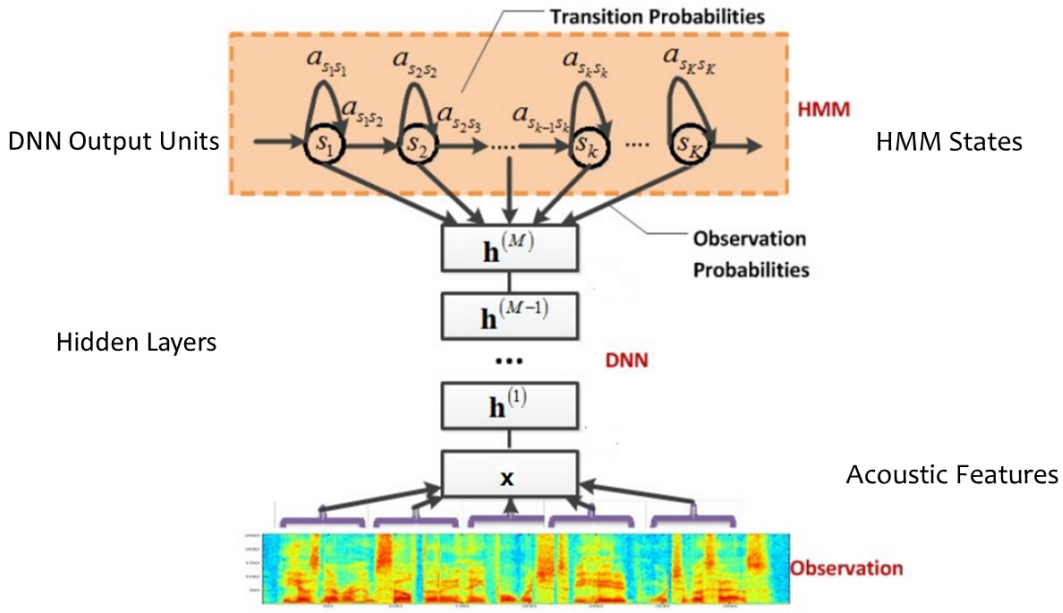

Figure 2.6: A Hybrid CD-DNN-HMM system.

In ASR, a DNN is used in hybrid a DNN-HMM to provide the state output distribution. Fig. 2.6 shows the hybrid DNN-HMM used in [31]. Unlike a GMM-HMM in which each state has a GMM as its output probability distribution, in DNN-HMM systems only one single DNN is used to model the output probability distributions of all HMM states generally.

During training, all acoustic feature frames extracted from the training set are first used for DBN pretraining. Then, an output layer consisting of units that represent phones or phonetic states are added to the DBN to form a DNN. Finally, backpropagation is performed as described above. Note that minimum cross entropy is just one of

the training criteria. As another category of training criteria, sequence training will be introduced later in Chapter 4.

During decoding, input frames are fed into the DNN to compute the posteriors of the states. The posteriors are then scaled by dividing the state priors, which are computed from the training data, to obtain the likelihoods of the frame given the states for HMM decoding:

$$P(\mathbf{x}|s_i) \propto \frac{P(s_i|\mathbf{x})}{P(s_i)} . \tag{2.18}$$

Today DNN-HMMs are deployed on most large vocabulary ASR systems. Its great success can be attributed to:

(a) DBN pretraining that provides a much better initialization point for a large network with many more parameters than a common GMM,

(b) its ability to extract information from a much longer context window of input frames, and

(c) its ability to model highly non-linear functions and distributions using many hidden layers which are regarded as high-level feature extractors.

In machine learning, there are many variants of the deep architecture. Like the great impact of DNN on ASR, deep convolutional neural network (deep-CNN) has gained great success in many computer vision tasks [32], meanwhile recurrent neural networks (RNN) are applied to language modeling [33]. Recently, deep-CNN and deep-RNN are also adopted for ASR [14, 15]. Their performance is generally comparable or even better than a DNN.

2.4 Phonetic Unit Selection

One important issue in ASR is how to choose the type of acoustic unit for modeling. In [34], Kai-Fu Lee defines "good" acoustic units with the following two criteria:

- **Consistence** means that different samples of the same unit have similar characteristics. It is important because it improves the discrimination between different modeling units, which governs the accuracy of speech recognition systems.

- **Trainability** means there are are sufficient examples for each speech unit. It is important because robust parameter estimation require a considerable amount of training data.

In the rest of this section, we will survey different phonetic units used for ASR using these two criteria.

2.4.1 Context-independent (CI) Units

- **Words**

  Words are the most natural units in speech and the word error rate (WER) is the most common measure of the performance of ASR systems. Word models have the advantages that they can capture within-word pronunciation variations. In small-vocabulary ASR tasks like digit recognition, word models usually have better results. However, constructing word models are not practical in large-vocabulary ASR tasks because not all words can be trained robustly since there are usually tens of thousands of words in a language and many words rarely occur. Therefore, sub-word units are more practical in large-vocabulary ASR.

- **Phones**

  Phones are the most popular basic units for common ASR systems for their good trainability. In general, only 40-50 phones are employed in English and therefore each of them can have sufficient data for robust training. In phone-based modeling, words are broken down into a concatenated sequence of phones according to a phonetic dictionary. Context-independent phones are also called monophones and they are modeled by individual HMMs. The use of monophones ignores the co-articulatory effects. Therefore their performance is modest compared with using context-dependent units.

- **Syllables**

  Phones are usually pronounced differently in different words. On the other hand, analysis shows that realization of syllables is more stable and consistent than phones [35]. These findings stimulate the interest to investigate the use of syllables for acoustic modeling. However, syllables suffer from poor trainability.

The main reason is that the number of distinct syllables (around 20,000 in English) is so large that it is impractical to collect enough training data for each of them. Furthermore, like phones, their boundaries are sensitive to contextual effects, which is a common defect of context-independent units.

2.4.2 Context-dependent (CD) Units

In order to model the co-articulatory effects in speech, several kinds of phone-based context-dependent units are developed.

Table 2.1: Phone transcription and triphone transcription of words.

Sentence	Phone sequence	Triphone sequence
CAT	k ae t	?-k+ae k-ae+t ae-t+?
DOG	d ao g	?-d+ao d-ao+g ao-g+?
EAT	iy t	?-iy+t iy-t+?
GREEN	g r iy n	?-g+r g-r+iy r-iy+n iy-n+?

- **Triphones**

  Triphones model context-dependency by considering the preceding and the following phones of each phone in the utterance. Table 2.1 lists the phone and triphone transcriptions of several English words. For example, triphone /k-ae+t/ represents the triphone with /ae/ as the base phone, /k/ as its previous phone and /t/ as its following phone. The symbol '?' here means the context depends on the previous or the next word in the sentence. Triphones became the most popular modeling units after Kai-Fu Lee demonstrated a successful and practical parameter sharing scheme with the use of triphones in 1990 [34]. After that, different parameter sharing schemes were further developed and the most widely-used method is clustering by a phonetic decision tree[2].

- **Word-dependent Phones**

  In word-dependent phone modeling, the same base phone in distinct words is modeled individually. Thus, it needs as many data as word models for robust training. But now since phones are labeled rather a whole word, parameter sharing or interpolation across different words become possible. The rationale be-

hind is that while an infrequent word is poorly trained with word models, the corresponding word-dependent phones can still be reasonably trained through interpolation with monophones.

- **Function-word-dependent Phones**

  Function words generally mean articles, prepositions, conjunctions and pronouns. For example, "THE", "A", "IN", "WITH", are function words in English. Such words appear more frequently than ordinary content words and have obvious pronunciation variant. [34] found that function words take only 4% of the vocabulary or about one third if weighted by frequency, but they account for nearly half of the recognition errors. [34] proposed function-word-dependent phones to enhance the modeling of function words. Function-word-dependent phones are identical to word-dependent phones, except that only a selected set of function words are represented by function-word-dependent phones, while the other words are still represented by triphones. As function words usually occur more frequently than other words, function-word-dependent phone can be estimated robustly.

Table 2.2: The characteristics of different phonetic units if they are used for large vocabulary continuous speech recognition task.

Phonetic Unit	Consistence	Trainability
Word model	good	bad
CI Phone model	bad	good
Syllable model	good	bad
CD phone model	good	through parameter sharing
Word-dependent phone model	good	through parameter sharing
Function-word-dependent phone model	good	through parameter sharing

Table 2.2 summarizes different phonetic units if they are used for speech recognition task in terms of consistence and trainability. From the above evaluation, with the same amount of training data, it seems that consistence and trainability are two conflicting goals. It is difficult to achieve the two goals at the same time by simply building independent acoustic models for each unit. However, researchers have explored ways to find a trade-off between the two requirements. In the next section, we will introduce

the popular parameter tying approach for robust context-dependent acoustic modeling in details. By clustering similar context-depedent units, and sharing some of the parameters among units in the same cluster, they can be robustly estimated.

2.5 Context-Dependent Acoustic Modeling

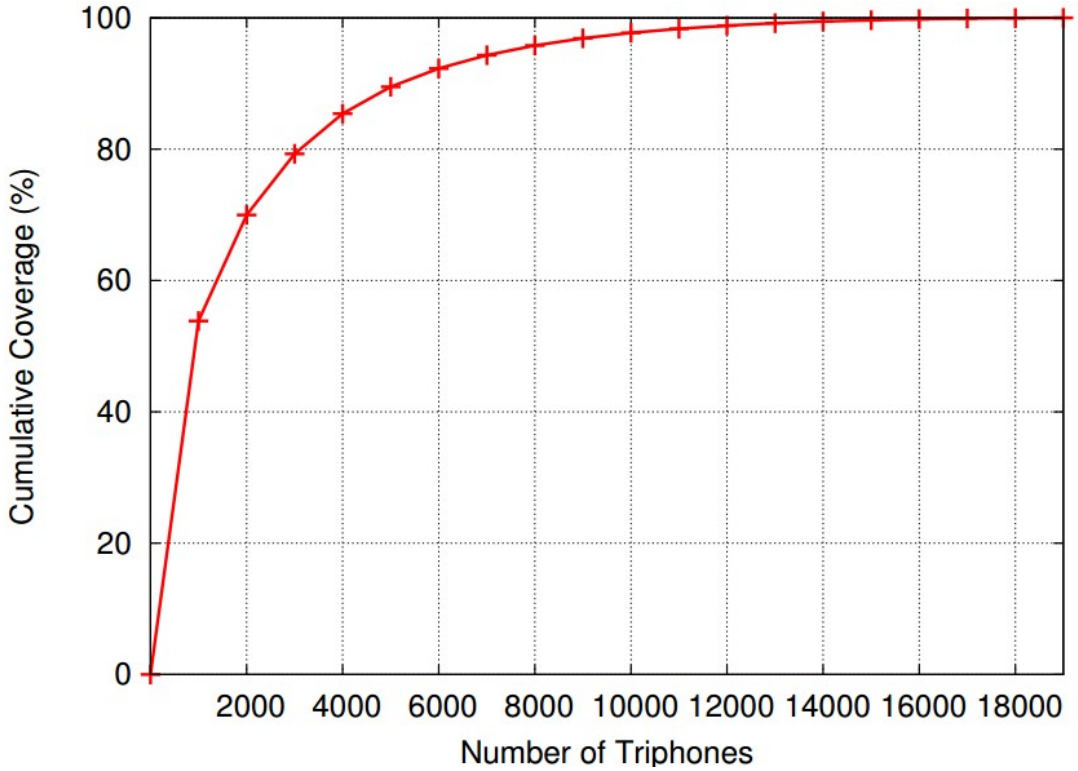

Figure 2.7: Cumulative triphones coverage in the WSJ training data set HUB2 . The triphones are sorted in descending order of their occurrence count.

Triphones are the most frequently used context-dependent units in ASR systems. During recognition, since we do not know the sequence of phones in the testing utterances, we have to consider every possible triphones. Thus, if there are N monophones, theoretically there can be as many as N^3 triphones. Typically, there are 64,000 triphones as N normally equals to 40. The number of distinct triphones is too large for having enough training data for every triphone.

What is worse, even for languages with rich resources, the data distribution over their triphones is so uneven that there are many infrequent triphones with insufficient training data. It was shown in [36] that the 80–20 Pareto rule is also true for the distribution of triphones in the 84-hour Wall Street Journal training corpora: about 80%

of all triphone occurrences in the corpus come from only 20% of the most common distinct triphones. Fig. 2.7 displays the cumulative triphones coverage in the WSJ training data set HUB2, indicating that most data samples are from a small portion of triphones. Direct estimation of the acoustic model parameters for a large set of distinct triphones suffers from the data scarcity problem and will yield poor models which hurt the overall recognition performance. Thus, the performance is worse than monophone when the triphones are trained individually.

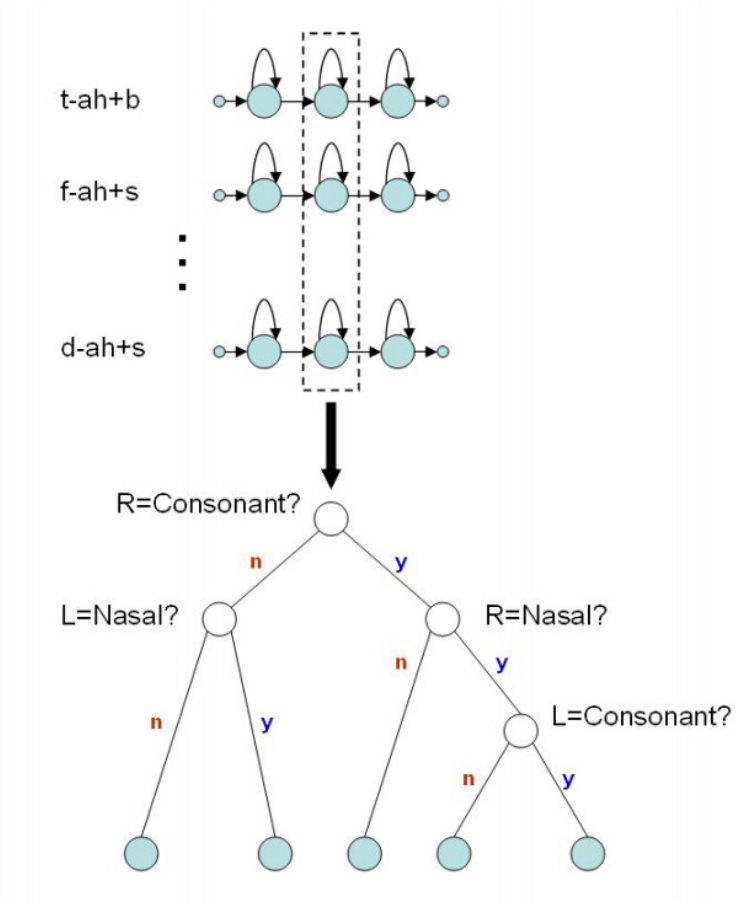

Figure 2.8: Phonetic decision tree-based state tying in the HTK toolkit [1].

Parameter tying simply shares parameters among acoustic units to address this problem. Tied subspace Gaussian distributions [37], shared distributions [38], tied states [2] and generalized triphones [34] share parameters among different kinds of acoustic units. State tying is the most popular method in modern ASR systems. It can be performed either in a data driven manner or based on phonetic knowledge. The latter way usually makes use of a phonetic decision tree, in which the degree of tying can be well controlled by setting some thresholds of state splitting and merging.

A phonetic decision tree is a binary tree in which each node asks a yes/no phonetic question. The questions are about the phonetic context of the triphones. In the tree shown in Fig. 2.8, the question at the root of the tree is asking whether the left neighboring phone of the current triphone is a consonant or not. Initially all triphone states with the same state index in the HMMs of triphones from the same base phone are placed at the root node of a tree. For example, all of the 3rd states of triphones with /ah/ as base phone are placed at the root. Depending on the answer, the pool of states is split and this continues until certain thresholds such as the minimum number of data samples from the states in the leaf node and the minimum likelihood gain of splitting nodes are reached. All states in the same stopping leaf node are then tied. For example, the tree shown in Fig. 2.8 will partition its states into subsets corresponding to the five terminal nodes. One tree is constructed for each state of each base phone. The tree topology and questions at each node are chosen to locally maximize the likelihood of the training data and ensure that sufficient data are associated with each tied state. Once all trees have been constructed, unseen triphones can be synthesized by finding the appropriate terminal tree nodes and then using the tied states associated with those nodes with the help of linguistic knowledge.

2.6 Data Scarcity Problem in ASR

Data scarcity is one of the largest obstacles in the development of human language technologies. This section introduces two common data scarcity problems in ASR which we will address later in this thesis:

- **Distinct Triphone Acoustic Modeling**

  As is described in the previous section, parameter tying successfully strikes a balance between detailed modeling and robust training, thus it is also very popular in DNN-HMM ASR systems. However, there is no discrimination among acoustic units tied together, which inevitably introduces quantization errors. The eigentriphone method together with the more recent reference model weighting (RMW) method [39] seek for robust distinct triphone modeling (see Section 5.1), but they only work for GMM-HMM system. How to perform robust distinct triphone modeling on DNN-HMM systems remains a problem.

- **Low-Resource Speech Recognition**

  In the past decades, huge efforts have been spent on the most popular languages such as English, French, German, Mandarin,... etc. As results, great success has been achieved. However, there are still many languages in the world that lack audio and language resources, and cannot benefit from the advanced human language technologies. Such languages usually have only a few hours of transcribed training data, and a good phonetic dictionary for them is also hard to obtain.

  The creation of language resources generally needs the help of native linguistics experts, and is usually costly and time-consuming; it is even more so if only non-native developers are available. Thus, an important research direction in low-resource ASR is to make the process easier and faster. For instance, a semi-automatic approach to preparing a pronunciation dictionary is to first create a small primary dictionary manually, and then extend it to a large dictionary by applying grapheme-to-phoneme conversion [40]. However, the performance of the final dictionary highly depends on the quality of the primary one. A simpler solution is to abandon the phone-based models and employ graphemes as the basic acoustic units because grapheme modeling [41, 42, 43, 44, 45] does not require a phonetic dictionary[1]. Many languages that use an alphabet writing system are suitable for grapheme-based acoustic modeling, and their grapheme set is usually selected to be the same as their set of alphabets.

  On the other hand, when out-of-domain data is available, transfer learning [46] may be applied. Notable efforts include cross-lingual ASR [47] and multi-lingual ASR [48]. A basic assumption behind these techniques is that there exist good explicit mappings between phones between different languages so that transfer learning may be applied to transfer the knowledge, or such mapping can be modeled implicitly by some shared parameters [49]. Nevertheless, explicit multi-lingual phone mapping is usually not available for many low-resourced languages.

[1] In practice, grapheme acoustic models can be trained using existing phone-based ASR softwares. During training, a "pronunciation dictionary" is simulated by simply representing each word by its graphemic transcription.

2.7 ASR System Performance Evaluation Metric

To make the recognition results of different research efforts comparable, the standard ASR performance evaluation metric is *word accuracy*. It plays an important role in the development of ASR techniques. In order to facilitate the discussions in the rest of the thesis, the definition of *word accuracy* is given in this section.

To compute the *word accuracy*, the recognized word string is first aligned to the correct word string using a string matching algorithm. After the alignment, the number of substitution errors (S), deletion errors (D) and insertion errors (I) can be calculated. Let N be the number of words in the correct word string, then the *word accuracy* can be calculated as:

$$\textit{word accuracy} = \frac{N - D - S - I}{N}. \tag{2.19}$$

2.8 Summary

The target of automatic speech recognition (ASR) is to decode the word sequence from the human speech. ASR systems are usually composed of a feature extractor, a language model and an acoustic model:

- the feature extraction component extracts acoustic features from waveform data for further processing;
- the language model assigns a prior probability to any word sequence $\mathbf{W}$;
- the acoustic model is used for modeling the likelihood of word sequence $\mathbf{W}$ given an observed acoustic feature sequence $\mathbf{X}$.

Most ASR systems use hidden Markov models (HMMs) as their acoustic models, and deep neural networks (DNNs) are utilized to model the state output probability distributions of any acoustic feature frame $\mathbf{x}$ given an HMM state s_i $P(\mathbf{x}|s_i)$. Generally, a DNN is trained in two steps: pre-training deep belief network unsupervised and supervised DNN backpropagation.

Before building ASR systems, one needs to carefully select the kind of phonetic units to model. Words and monophones are context-independent units that do not

consider the co-articulatory effects in speech. Triphones model the co-articulatory effect caused by the preceding and following phones, but direct estimation of triphones will result in overfitting because there are not sufficient training data for infrequent triphones. State tying addresses this problem by tying similar triphone states together, and is very popular in ASR systems.

However, triphone states that are tied together cannot be distinguished with each other in decoding, resulting in quantization errors. However, it is hard to model individual distinct triphones with limited training data. Moreover, speech recognition for low-resource languages suffers from the lack of training data and other language resources. Although ASR techniques have made great progress in recent years, the data scarcity problem in ASR remains to be solved.

CHAPTER 3

MULTI-TASK LEARNING DEEP NEURAL NETWORK

Before moving to the particular multi-task learning (MTL) methods, we introduce the concepts and previous works on MTL. We will then illustrate our MTL formulas and multi-task learning deep neural network (MTL-DNN), which is the fundamental machine learning model in this thesis. The basic MTL-DNN formula will be shown as well.

3.1 Multi-task Learning (MTL)

Multi-task learning (MTL) [18] or learning to learn [50] is a machine learning approach that aims at improving the generalization performance of a learning task by jointly learning multiple related tasks together. It is found that if the multiple tasks are related and if they share some internal representation, then through learning them together, they are able to transfer knowledge to one another. As a result, the common internal representation thus learned helps the models generalize better for future unseen data.

MTL has been an active research direction since two decades ago. There have been papers analyzing the behavior and generalization bound of MTL theoretically. In [51], a statistical learning theory based approach to MTL is developed and an explicit bound on the average error of MTL is derived. The bound indicates that when many related tasks are learned together, the number of training examples needed to achieve a desired performance can be greatly reduced. It further shows that the parameter hypothesis space learned from a set of related tasks can benefit novel related task in the same environment by providing a better initial choice of the hypothesis space. [52] focuses on the relatedness of tasks. On a data generation model, the notion of relatedness among multiple tasks is defined in a particular way so that a tighter generalization bound for each learning task can be derived. Precise conditions under which the MTL

approach is guaranteed to outperform the single task learning (STL) approach are also given.

On the other hand, Caruana's thesis [18] takes an experimental direction to study the effect of MTL. By a series of carefully designed experiments using backprop artificial neural networks, he tried to reveal how MTL helps learning the primary task. Caruna postulated two requirements for related tasks to be jointly learned under the MTL approach:

(a) the related tasks must share input features, and

(b) the related tasks must share hidden units to benefit each other when trained with MTL-backprop.

Moreover, training related tasks together is not guaranteed to bring performance gain. The training algorithm also plays an important role in MTL.

Assume that two tasks $\mathcal{T}_1$ and $\mathcal{T}_2$ both make use of an internal feature F (although in different manners). The training process of each task has two targets: learning the feature F, and learning its own way to use F. Caruana listed out some task relationships which would enable MTL-backprop to learn better internal representation (or more generalized model) of the related tasks:

(a) **data amplification**: For each task, MTL training increases the data sizes with the extra information received from the training signals of the related tasks. The data amplification effects are further classified into statistical data amplification, sampling data amplification and blocking data amplification.

(b) **eavesdropping**: Assume $\mathcal{T}_1$ utilizes F in a simple way, while $\mathcal{T}_2$ utilizes it in a way so complicated that it is difficult to learn the feature F by simply training $\mathcal{T}_2$ with a limited amount of data. When $\mathcal{T}_1$ and $\mathcal{T}_2$ are learned together with a shared internal representation in an artificial neural network, $\mathcal{T}_2$ can eavesdrop F that $\mathcal{T}_1$ learns in this representation since $\mathcal{T}_1$ is easier to learn.

(c) **attribute selection**: Assume that F is a function that is defined on only a few inputs, and most inputs have nothing to do with F, and thus for $\mathcal{T}_1$ and $\mathcal{T}_2$, they can be ignored. With significant noises or a limited amount of training data, $\mathcal{T}_1$

has problem filtering out those inputs that are irrelevant to learning F from the high-dimensional input space. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are learned together, both will better select the inputs that are more relevant to F because more training signals for F are now provided.

(d) **representation bias**: Consider the two tasks above, MTL training of the two tasks will force the learned parameters of F to move towards the intersection area of the two F's individually learned by the two tasks. The learned F is expected to generalize better since it is satisfactory to both tasks.

(e) **overfitting prevention**: Suppose task $\mathcal{T}_1$ is easy to overfit F if it is trained individually, due to insufficient data. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are trained together, $\mathcal{T}_2$ has chances to save $\mathcal{T}_1$ from overfitting. At some time stamp of MTL backpropagation training, when $\mathcal{T}_1$ gets trapped in a local optimum causing overfitting, $\mathcal{T}_2$ may still provide a gradient that drives the shared feature F out of the trap.

All of the above effects can be summarized as:

(a) secondary tasks bring more training signal and extra information and

(b) joint training with secondary tasks is equivalent to adding a regularizer in the original objective function of the primary task, which helps achieve better generalization.

In addition, the author suggests using the outputs of secondary tasks as inputs to another classifier to further improve the primary task.

More recent research on MTL has substantial interest on automatically learning the relationships between tasks. In [53], a local constraint defined by task relationships that are obtained in advance is added to the optimization problem. The relationships are fixed during the training procedure. Since it is preferred to learn the relationships from data automatically, a convex formulation for learning task relationships is derived in [54]. It assumes that the parameters of multiple linear regression tasks have the same matrix-variate normal distribution prior and the relationship among them is defined by the column covariance matrix. By this way, the relationship was learned to model task unrelatedness, positive task correlation and also negative task correlation. Later, the

authors extended this work by combining multi-task feature selection and relationship learning to handle high-dimensional input data. A similar work in [55] further incorporates the structure of features. Multi-task relationship learning was applied to solving real-life tasks such as learning robot inverse dynamics, examination score prediction and cancer tumor classification.

3.2 Multi-task Learning in ASR Using DNNs and Our Formulas

In ASR, MTL has been applied to improving performance robustness using recurrent neural networks [56]. It investigates training the speech enhancement task with 11 other classification tasks such as identifying the gender, the identity and the emotion of the speaker. On the classification tasks, MTL training outperforms multicondition training; on the speech enhancement task, it is also superior to spectral subtraction and noisy condition.

With the recently very successful deep neural networks (DNNs), we expect that DNNs may be used to further improve MTL performance; we call the resulting deep neural networks *MTL-DNNs*. For instance, Meltzer and Droppo investigated the training of monophone models for TIMIT phone recognition together with the learning of the phone labels, state contexts, or phone contexts [22]; significant gains were reported. However, the work did not model triphone states directly and it is not clear if it is really better to use the triphone contexts as the secondary task in learning monophone state posteriors in the MTL framework. MTL has also been employed successfully to train multi-lingual DNNs [49, 57, 58] (see Section 6.1 for more details). In these works, data from multiple languages are used in the pre-training step. Then, multiple output layers, one for each language are added on top of it. During subsequent fine-tuning, data from all training languages are fed through the common hidden layers but each language keeps its own language-specific output layer. However, when a training frame from one language is fed into the network, the units in other language's output layers will not be activated at all. Thus the output layers of other languages will not be trained by this frame. Therefore, unlike the MTL work in [18] and [22], for each input only one task is being trained, and the relatedness among the tasks are exploited

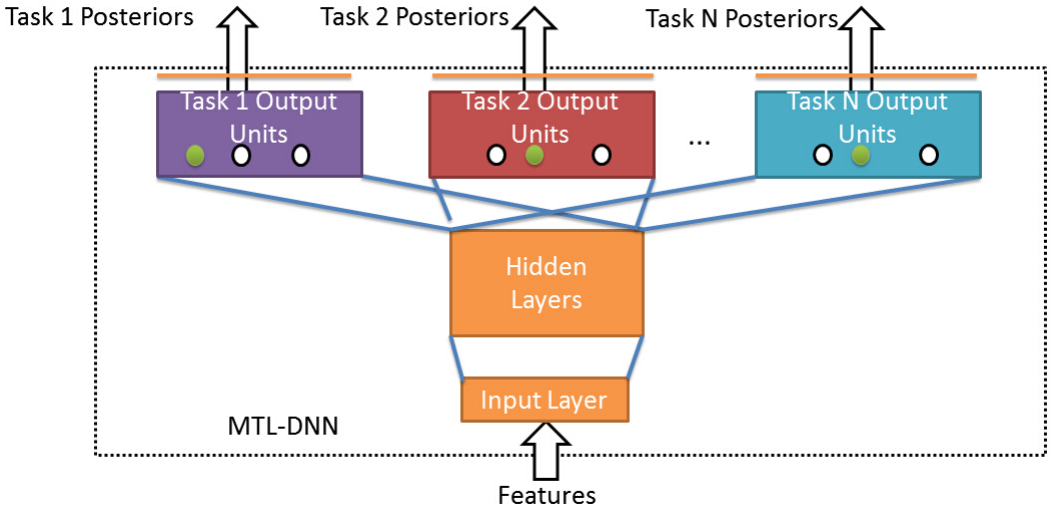

Figure 3.1: MTL-DNN used in this thesis. Outputs, labelled as green, from multiple separate tasks are turned "on" by an input vector. The bar on top of each output layer represents softmax function over the activation of nodes in that output layer.

only by enforcing common weights in the hidden layers. We will follow the notation in [57] and call these multi-lingual DNNs with shared hidden layers SHL-MDNNs.

In this thesis, we would like to apply the MTL framework to improve phone-based acoustic models for ASR using DNNs that strictly follows Caruana's two MTL requirements. That is, for each acoustic training input, one or more related secondary tasks are learned in parallel with the primary task of learning the phonetic models, and the multiple tasks share the hidden layers in a DNN.

We believe that the first requirement may give better MTL performance as it allows learning the hidden layers with constructive or destructive error gradients simultaneously from multiple tasks for each input; however, the requirement was not always enforced by some previous works (e.g., SHL-MDNN [57]). In terms of task relationships, we will make use of data amplification, representation bias, and overfitting prevention in the selection of secondary task(s), which, together with the use of early stopping in DNN fine-tuning, will result in a more generalized acoustic model.

Fig. 3.1 shows the structure of MTL-DNN we use in this proposal. Let's assume there are K tasks $\mathcal{T} \equiv \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}$ to learn under the MTL framework. The MTL model parameters are represented by $\Lambda \equiv \{\lambda_0\} \cup \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$, where λ_0 consists of model parameters that are shared by *all* tasks and λ_k consists of model parameters specific to only task $\mathcal{T}_k$. In our case, λ_0 represents the shared weights from

all hidden layers, whereas λ_k represents the weights in each task-specific output layer of $\mathcal{T}_k$. Without loss of generality, $\mathcal{T}_1$ will always be taken as the primary task, and the rest are secondary (or extra) tasks. The training objective function $\mathcal{E}$ is formulated as the weighted sum of the error functions of all the tasks as follows:

$$\mathcal{E}(\mathcal{D}, \Lambda) = \sum_{\mathbf{x} \in \mathcal{D}} \left(\sum_{k=1}^{K} \beta_k \, \mathcal{E}_k(\mathbf{x}; \lambda_0, \lambda_k) \right) \tag{3.1}$$

where $\mathcal{E}_k$ and β_k are the error function and the task weight of $\mathcal{T}_k$ with $\sum_{k=1}^{K} \beta_k = 1.0$; $\mathcal{D}$ is the whole set of training vectors from all tasks and $\mathbf{x}$ is one input vector. After training, only the model parameters (λ_0 and λ_1) of the primary task $\mathcal{T}_1$ are needed, and those of the secondary task(s) may be discarded. The key to successful application of MTL is to identify effective related learning tasks in the context of ASR.

3.3 Summary

Multi-task learning (MTL) jointly learns multiple tasks together to improve their generalization to unseen data. The tasks should share the same input features and an internal representation.

Multi-task learning deep neural network (MTL-DNN) is an ideal MTL model. Multiple tasks share the hidden layers, but have their own output layers. The hidden layers in MTL-DNN capture the common knowledge that can be transferred from one task to another. During training, for each training data sample, multiple output units will be activated; therefore, there are more than one error signals for each input. The objective function can be simply the sum of objective functions of all the tasks involved in MTL training.

CHAPTER 4

JOINT ACOUSTIC MODELING OF PHONE AND GRAPHEME: INFORMATION FROM WRITING SYSTEM

Following the guidelines in Section 1.2, we start to investigate various kinds of secondary tasks. Phones and its variants are widely used in ASR systems, but there are other candidates that are also suitable for acoustic modeling. In Method 1, we propose training phone and grapheme acoustic models together under the MTL framework. We hope to exploit extra information from the writing system of a language to benefit speech recognition.

4.1 Graphemes as Acoustic Units in ASR

Most ASR systems use phonemes or sub-phonemic units such as states in their HMM acoustic models. Given a word or word sequence, in order to build an HMM for training or decoding, a pronunciation dictionary is used to break the orthography of a word to its corresponding phoneme sequence. The production of such a dictionary usually needs participation of native speaker and linguistic expert, which is impractical for many low-resource languages in the world. For non-native developers, the process is even more expensive and time-consuming. One way of partially automating the development of a pronunciation dictionary is to first prepare a small seed dictionary manually, and then bootstrap a large dictionary by applying grapheme-to-phoneme conversion [40]. However, the quality of the primary one strongly affects the performance of the final dictionary . If the seed dictionary is not sufficiently general to cover all the implicit grapheme-to-phoneme relations in the language, the performance of the overall system will not be good. Therefore, people started to investigate other units for acoustic modeling without the need of phonetic dictionary.

Grapheme is one of the choices. It is the smallest unit used in describing the writing system of a language. [59] summarizes different categories of writing systems in

Table 4.1: Writing systems in the world.

Category of writing system	Meaning of characters	Example
Logosyllabary	Individual words or a particular syllable	Chinese
Syllabary	Particular syllables	Japanese
Abjad or Consonantory	Consonants	Arabic
Alphabet	Vowels or consonants	European languages
Abugida	A consonant accompanied by a specific vowel	South and Southeast Asia languages
Featural	Distinctive features of the segment	Korean

Table 4.2: Phone transcription and grapheme transcription of words.

Word	Phone sequence	Grapheme sequence
CAT	k ae t	c a t
DOG	d ao g	d o g
EAT	iy t	e a t
GREEN	g r iy n	g r e e n

the world, as listed in Table. 4.1. The authors hold a view that writing system actually preserves speech over time and distance. [44] studies graphemes in different types of writing systems, and concludes that graphemes in Alphabet, Syllabary and Abugida writing systems are suitable for acoustic modeling, covering a large portion of langauges in the world.

One of the advantages of grapheme-based acoustic modeling is that it does not need a phonetic dictionary. For example, the grapheme sequence of the English word *CAT* can be simply obtained by breaking it to alphabet sequence $\{$*'c' 'a' 't'*$\}$. The phone transcriptions and grapheme transcriptions of several example English words are showed in Table 4.2. Although the grapheme representation is not directly derived from its speech sound, grapheme-based acoustic modeling is a promising alternative in automatic speech recognition for languages without a good phonetic dictionary. [41] evaluates context-dependent graphemes on German ASR, and [42] further generalizes it to three other European languages: Dutch, Italian, and English. Grapheme-based acoustic modeling is also adopted for multi-lingual ASR, showing improvement for German with the multilingual acoustic models in [60]. On the issue of state tying for trigraphemes, [61] investigates different kinds of graphemic questions in decision-tree-based state tying on several languages and concluded that simple questions asking only the identity of the neighboring grapheme (named as singleton questions) work well. For low-resource languages, grapheme-based acoustic modeling was showed to be comparable to phone-based modeling in several South African languages [45].

4.2 Joint Phone and Grapheme Acoustic Modeling by MTL-DNN

Acoustic modeling of trigraphemes of the *same* language is chosen as the secondary task in the training of its triphone acoustic models using an MTL-DNN. That is, the two tasks in this MTL are:

$\mathcal{T}_1$ (primary task): posteriors of triphone senones (tied states) or monophone states

$\mathcal{T}_2$ (secondary task): posteriors of trigrapheme senones or monographeme states

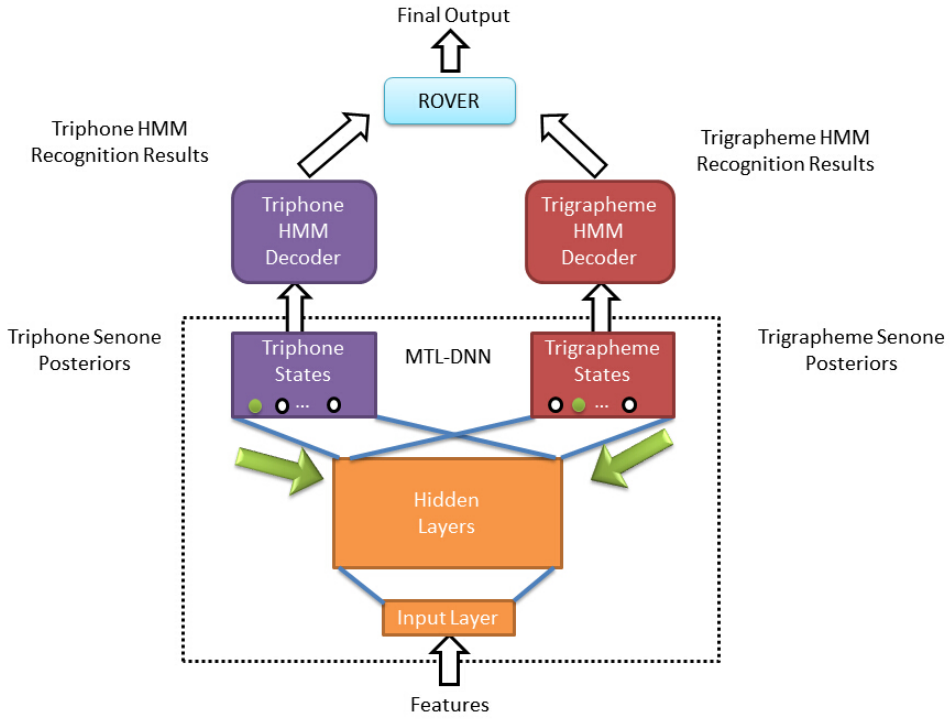

Figure 4.1: An MTL-DNN system for joint training of triphone and trigrapheme acoustic models (MTL-DNN-PG).

There are several motivations for the choice:

- although it may not be absolutely necessary, humans usually learn a language by reading, listening, and speaking. Hence, the joint learning of phones and graphemes is a real-life example of MTL, and we would like to repeat its success in ASR;
- past experiences in low-resource ASR [41]–[45] show that when the two tasks are trained individually, they give comparable recognition performance for many languages. Thus, their joint training may benefit each other; and
- grapheme-based acoustic modeling requires no additional language resources besides those already used by phone-based acoustic modeling. That is an important advantage in low-resource ASR.

Fig. 4.1 shows an overview of the proposed MTL-DNN system for the joint training of triphone and trigrapheme acoustic models; we will denote this DNN as MTL-DNN-PG. The DNN architecture is similar to the one used in common multi-lingual ASR

[49, 57]. Essentially two single-task learning DNNs (STL-DNNs), one for training triphone models and the other for training trigrapheme models are merged so that their hidden layers are shared, while each of them keeps its own output layer. The two output layers are trained to model the posterior probabilities of triphone senones (tied states) and trigrapheme senones respectively for the *same* given input acoustic frame. That is, given an input acoustic vector $\mathbf{x}$, the posterior probability of the ith triphone senone $s_i^{(1)}$ at the triphone output layer is computed using the following softmax function:

$$P(s_i^{(1)}|\mathbf{x};\lambda_0,\lambda_1) = \frac{\exp(y_i^{(1)})}{\sum_{i'=1}^{N_1}\exp(y_{i'}^{(1)})}\,,\ \forall i = 1,\ldots,N_1, \tag{4.1}$$

where $y_i^{(1)}$ is the activation of the senone $s_i^{(1)}$ and N_1 is the total number of triphone senones (in task $\mathcal{T}_1$). A similar formula may be derived for the posterior probabilities $P(s_i^{(2)}|\mathbf{x};\lambda_0,\lambda_2)$ of the N_2 trigrapheme senones (in task $\mathcal{T}_2$). For each training frame $\mathbf{x}$, the error function of task $\mathcal{T}_k$ $(k = 1, 2)$ is to minimize the following per-frame cross entropy:

$$\mathcal{E}_k(\mathbf{x};\lambda_0,\lambda_k) = -\sum_{i=1}^{N_k} d_i^{(k)} \log P(s_i^{(k)}|\mathbf{x};\lambda_0,\lambda_k)\,, \tag{4.2}$$

where $d_i^{(k)}$ is the target value of the ith senone in $\mathcal{T}_k$. Finally, the task errors are weighted and summed over all training frames as described in Eq. (3.1).

The triphone and trigrapheme senones in the MTL-DNN-PG are obtained from their corresponding tied-state GMM-HMM systems. The triphone and trigrapheme GMM-HMMs are also utilized to obtain the frame labels and senone priors by forced aligning the training data and development data. During MTL-DNN training, the target values of exactly one triphone senone output unit *and* one trigrapheme senone output unit will be set to 1.0 per training frame. During decoding, each senone posterior probability is converted back to a scaled likelihood by dividing it by its prior as follows:

$$P(\mathbf{x}|s_i^{(k)};\lambda_0,\lambda_k) \propto \frac{P(s_i^{(k)}|\mathbf{x};\lambda_0,\lambda_k)}{P(s_i^{(k)})}\,. \tag{4.3}$$

Afterward, Viterbi decoding is performed using either the triphone or the trigrapheme MTL-DNN-HMM[1].

[1] Although we start with the goal of improving a phone-based ASR system, as we will see, for some low-resource language, a grapheme-based ASR system may perform better when the amount of training data is small.

In fact, one may even combine the independent decisions from the triphone decoder and the trigrapheme decoder to get better result using, for example, ROVER [62]. Rover is a tool for combining hypothesized word outputs of multiple recognition systems and select the best scoring word sequence. In our work, for multiple word hypothesis, we simply select the word with higher confidence as the final decision.

One limitation of using grapheme acoustic modeling as the secondary task is that graphemes may not be the appropriate modeling units for all languages; graphemes are closely related to the acoustic manifestation of alphabetic languages only. Fortunately, the majority of languages in the world is alphabetic [44].

4.3 Analysis of Task Relationship

As is mentioned in Section 3.1, MTL training exploits extra information from secondary tasks. In the case of joint training of phone and grapheme acoustic models, the exploited extra information is the implicit phone-to-grapheme mapping. We believe this information is embedded in the model parameters of the learned MTL-DNN.

The hidden layers of DNN are regarded as feature extractors learned from training data. Similarly, the hidden layers in MTL-DNN are shared feature extractors for the two tasks during training. Take English as example. In English, the grapheme 'm' is always pronounced as consonant phone /m/. Since the labels for DNN training comes from forced-alignment results by acoustic models, for training acoustic frames with a grapheme label 'm', their phone labels are most likely /m/. During training, when these frames are fed into the network, the target values of phone /m/ in the phonetic output layer, and grapheme 'm' in the graphemic output layer will be set to 1.0 at the same time, while for the other frames, both target values are 0. Thus the two units are trained to have similar activations given an input acoustic frame, causing high correlation in their parameters.

While the parameters in hidden layers are shared by all output units, let us call those in output layers unit-specific parameters. Since these parameters are connection weights to the nodes in the topmost hidden layer, the number of unit-specific parameters that an output unit has is the same as the number of hidden nodes in the topmost hidden layer.

For each output unit i, concatenating these parameters together in strictly the same order, we can obtain a vector of connection wieghts $\mathbf{w}_i$. We further define the correlation between two output units i and j as the cosine angle of their parameter vectors:

$$Corr_{ij} = \frac{\mathbf{w}_i \mathbf{w}_j}{||\mathbf{w}_i||_2 ||\mathbf{w}_j||_2} \tag{4.4}$$

By computing the correlation between each pair of phone and grapheme output units, we can get a correlation matrix between the two tasks.

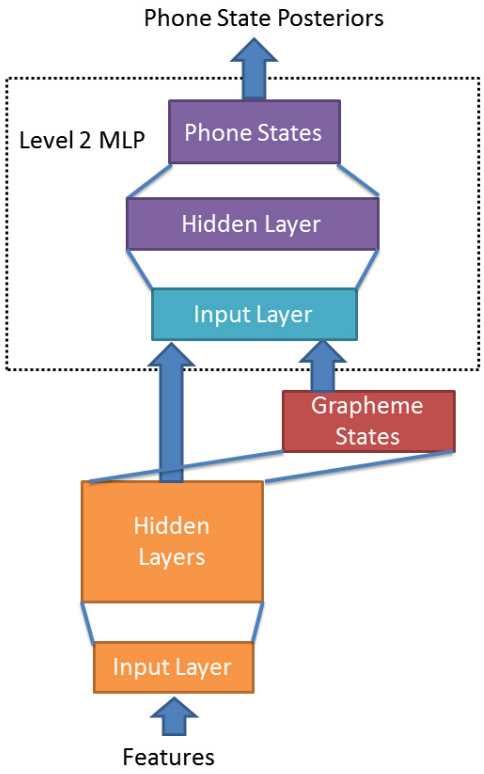

Figure 4.2: Triphone MTL-DNN2: Stacking an STL-MLP on top of the MTL-DNN system of Fig. 1.

4.4 Extension 1: MTL-DNN with an STL-MLP (Triphone MTL-DNN2)

We further investigate if the trigrapheme posteriors that are obtained as a by-product of MTL may be useful features for triphone modeling when the amount of training data

is small. In a manner similar to the use of NN tandem features in HMM training [63], we concatenate the outputs from the shared hidden layers with the trigrapheme senone posteriors from the well-trained MTL-DNN-PG and feed them to another STL multi-layer perceptron (MLP) to estimate the triphone posteriors again. The MLP has only a single hidden layer with 2048 units, and an output layer with triphone senone targets. Back-propagation is performed to train this MLP while keeping the MTL-DNN-PG unchanged. The corresponding system, which we call *triphone MTL-DNN2*, is shown in Fig. 4.2.

4.5 Extension 2: Joint Sequence Training

Recall that in Section 2.3.3 DNN is trained by minimizing the sum of cross entropies on all training frames, in which each acoustic frame is treated as independent data sample. However, speech recognition is essentially a sequential labeling problem. Neighboring acoustic frames have strong co-articulatory effects with each other. The frame-wise criterion does not capture the very long term correlation among the target classes in an utterance, either.

On the other hand, sequence-discriminative training better matches the performance metric of large vocabulary speech recognition decoding by considering sequence constraints from the HMMs, the dictionary, and the language model. The objective function is generally defined over all frames in the whole utterance, and is closer to the objective of ASR: minimum insertion, deletion and substitution errors. Since long ago, it has been an indispensable step in building state-of-the-art ASR systems that are based on hidden Markov models (HMMs) with state output probability distributions estimated using Gaussian mixture model (GMMs). Recently, sequence-discriminative training has been extended to DNN training using different training criteria, such as minimum Bayes risk (MBR) [64], minimum phone error (MPE) [5], maximum mutual information (MMI) [26] and boosted MMI (BMMI) [65]. For example, the MMI criterion is

$$F_{MMI} = \sum_u \log \left(\frac{P(\mathbf{X}^{(u)}|\mathbf{W}_p)^{\kappa_p} P(\mathbf{W}_p)}{\sum_{\mathbf{W}'_p} P(\mathbf{X}^{(u)}|\mathbf{W}'_p)^{\kappa_p} P(\mathbf{W}'_p)} \right) \tag{4.5}$$

where $\mathbf{W}_p^{(u)}$ is the true phonetic transcription of the utterance u; $\mathbf{X}^{(u)} = \{\mathbf{x}_1^{(u)}, \mathbf{x}_2^{(u)}, ..., \mathbf{x}_{T_u}^{(u)}\}$

is its acoustic observation sequence; $P(\mathbf{W}_p)$ is the probability of $\mathbf{W}_p$ given by the decoding word/phone/state lattice. κ_p is the likelihood scales used in training respectively. Consistent improvements are reported in both phone recognition [66] and large-vocabulary ASR [67, 68, 69].

Despite of different objective functions, sequence training on DNN-HMM usually starts with a DNN-HMM trained by framewise criterion, and go through the following steps:

STEP 1 : For each training utterance, perform forced alignment using the truth transcription with the current DNN-HMM to obtain the numerator lattice, which is used to compute the numerator term in Equation 4.5;

STEP 2 : For each training utterance, perform N-best recognition with the current DNN-HMM to obtain the denominator lattice, which is used to compute the denominator term in Equation 4.5;

STEP 3 : Training is done one utterance by one utterance. The posterior of training frames are computed by feeding the acoustic inputs forward through the DNN as usual, but now the gradients for backpropagation are computed from the numerator and denominator lattices by the Extended Baum-Welch algorithm [4].

The joint training of phone and grapheme acoustic models using an MTL-DNN described in the previous sections is found effective [70]. Nevertheless, the optimization criterion of minimizing the total frame-wise cross-entropies does not take into account the correlation between neighboring frames. Since sequence-discriminative training has been applied successfully to STL-DNN [66, 67], we would like to further investigate the effectiveness of joint sequence-discriminative training of both phone and grapheme acoustic models using an MTL-DNN. Moreover, since it has been shown in [67] that the various discriminative training criteria give similar performance, we simply choose the minimum phone error (MPE) criterion for the phone-based decoder, and the minimum grapheme error (MGE) criterion for the grapheme-based decoder. Hence, the joint sequence-discriminative training criterion of our MTL-DNN is to minimize the sum of phone errors and grapheme errors by optimizing the objective

as follows:

$$
\begin{aligned}
F_{mpge} &= F_{mpe} + F_{mge} \\
&= \sum_u \left(\frac{\sum_{\mathbf{W}_p} P(\mathbf{X}^{(u)}|\mathbf{W}_p)^{\kappa_p} P(\mathbf{W}_p) A(\mathbf{W}_p, \mathbf{W}_p^{(u)})}{\sum_{\mathbf{W}'_p} P(\mathbf{X}^{(u)}|\mathbf{W}'_p)^{\kappa_p} P(\mathbf{W}'_p)} \right. \\
&+ \left. \frac{\sum_{\mathbf{W}_g} P(\mathbf{X}^{(u)}|\mathbf{W}_g^{(u)})^{\kappa_g} P(\mathbf{W}_g) A(\mathbf{W}_g, \mathbf{W}_g^{(u)})}{\sum_{\mathbf{W}'_g} P(\mathbf{X}^{(u)}|\mathbf{W}'_g)^{\kappa_g} P(\mathbf{W}'_g)} \right),
\end{aligned}
$$

where $\mathbf{W}_p^{(u)}$ and $\mathbf{W}_g^{(u)}$ are the true phonetic and graphemic transcriptions of the utterance u; $\mathbf{X}^{(u)} = \{\mathbf{x}_1^{(u)}, \mathbf{x}_2^{(u)}, ..., \mathbf{x}_{T_u}^{(u)}\}$ is its acoustic observation sequence; $A(\mathbf{W}_p, \mathbf{W}_p^{(u)})$ is the phonetic transcription accuracy of the utterance defined as the number of correct phone labels in $\mathbf{W}_p^{(u)}$ minus the number of errors in the hypothesis $\mathbf{W}_p$; $P(\mathbf{W}_p)$ is the probability of W_p given by the lattice. The graphemic transcription accuracy $A(\mathbf{W}_g, \mathbf{W}_g^{(u)})$ is defined in a similar way. κ_p and κ_g are the likelihood scales used in MPE and MGE training respectively.

Taking the derivative of F_{mpge} w.r.t. $\log p(\mathbf{x}_t|s)$, we obtain, for the phone state s in phone a,

$$
\frac{\partial F_{mpge}}{\partial \log P(\mathbf{x}_t^{(u)}|s)} = \kappa_p \gamma_{p,t}^{den(u)}(s) \left(\bar{A}_p^{(u)}(s(t) \in \mathbf{S}_a) - \bar{A}_p^{(u)}(*) \right)
$$

where $\mathbf{S}_a$ is the set of states of phone a; $\bar{A}_p^{(u)}(*)$ is the average accuracy of all the paths in the lattice of utterance u; $\bar{A}_p^{(u)}(s(t) \in \mathbf{S}_a)$ is the average accuracy of those paths going through phone a at time t in the phone lattice; $\gamma_{p,t}^{den(u)}(s)$ is the posterior probability that at time t the utterance u reaches state s, and is calculated by the extended Baum-Welch algorithm using the phone denominator lattice. Similarly,

$$
\frac{\partial F_{mpge}}{\partial \log P(\mathbf{x}_t^{(u)}|s)} = \kappa_g \gamma_{g,t}^{den(u)}(s) \left(\bar{A}_g^{(u)}(s(t) \in \mathbf{S}_b) - \bar{A}_g^{(u)}(*) \right)
$$

for grapheme state s in grapheme b. Note that the phone lattice and grapheme lattice of the same utterance are disjoint.

An overview of the sequence training procedure is shown in Fig. 4.3. Firstly, an MTL-DNN-PG is trained by minimizing the total frame-wise cross-entropies. Then the well-trained MTL-DNN-PG is used to produce both the phone and the grapheme

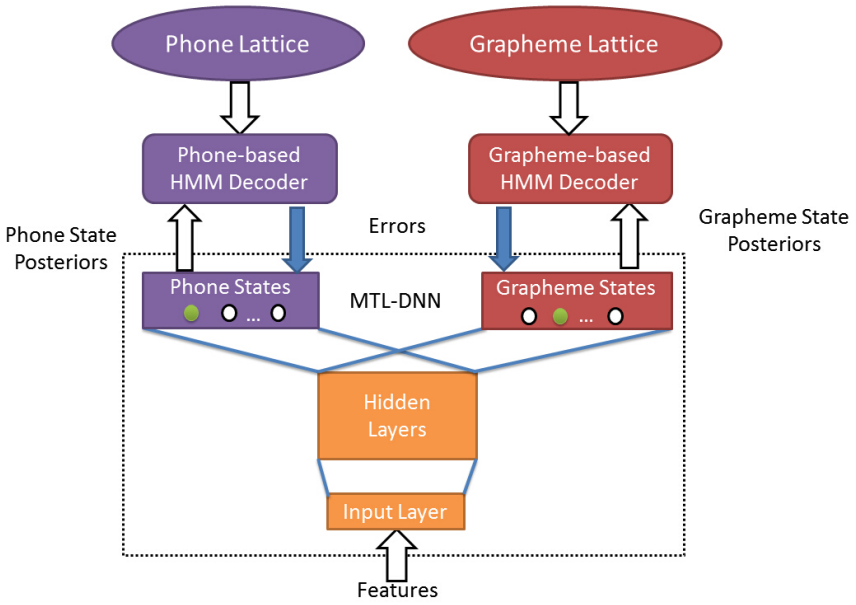

Figure 4.3: Joint sequence training of phone and grapheme MTL-DNNs.

state posteriors of each training utterance. The phone posteriors are used by the phone-based decoder to generate the phone denominator and numerator lattices for the utterance, while the grapheme state posteriors are used by the grapheme-based decoder to generate the grapheme denominator and numerator lattices separately. Finally, the following procedure is repeated for each utterance u in the data set:

STEP 1 : Acoustic features of the whole utterance are again fed into the MTL-DNN-PG to produce the posteriors of the phone and grapheme states.

STEP 2 : The two phone-based and grapheme-based decoders take in the corresponding state posteriors and compute the respective MPE and MGE statistics and the required gradients using the extended Baum-Welch algorithm.

STEP 3 : The weights of the MTL-DNN-PG are updated by back-propagating the combined MPE and MGE errors from the two decoders through the hidden layers to the bottom layer.

4.6 Experiment Evaluation

The proposed joint phone and grapheme acoustic model was evaluated on three tasks.

- TIMIT English phone recognition [71] is a simple and popular English phone

recognition benchmark to examine new ideas in ASR quickly. Grapheme is regarded to be inappropriate for English acoustic modeling, and it would be interesting to see whether it helps the primary phone modeling task. Moreover, since we understand English, analyzing the observations and experimental results will be easier.

- Wall Street Journal reading speech recognition is a larger English corpora. We would like to know if the improvement from MTL still holds when there are enough training data or not. Moreover, different from TIMIT task, WSJ is English word recognition task.

- As theoretically MTL is more effective when data are limited, we also evaluate the proposed method on true low-resource languages.

4.6.1 TIMIT Phone Recognition

English is notorious for the complicated relationship between its writing and pronunciation. In fact, grapheme-based acoustic models perform much worse than phone-based acoustic models in English [44]. Even though, we would like to start from TIMIT to evaluate our proposed joint training method using MTL-DNN. Because of our better understanding of the English language, this evaluation is also designed to verify our claim that the proposed MTL-DNN-PG method exploits extra information in the acoustic data — which is the implicit phone-to-grapheme mappings — to learn a more generalized acoustic model.

4.6.1.1 The TIMIT Corpus

Table 4.3: Information of TIMIT data sets.

Data Set	#Speakers	#Utterances	#Hours
training	462	3,696	3.14
core test	24	192	0.16
development	24	192	0.16

The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT - Texas Instruments (TI) and Massachusetts Institute of Technology (MIT)) [71] contains speech

of American English speakers of different sexes and dialects. Both phonemically and lexically transcriptions are provided. The speech data were recorded using a Sennheiser close-talking microphone at 16 kHz rate with 16 bit sample resolution, from 8 major dialect regions of the United States in 1980s. The collection process of the speech data is prompted carefully to be phonetically-balanced. Even after decades, it is still a standard experiment database for both the speech recognition and speaker recognition communities. One reason is that each utterance is phonetically hand labelled and provided with codes for speaker number, gender and dialect region. Moreover, it is small and thus convenient to quickly test new approaches.

The standard NIST training set consisting of 3,696 utterances from 462 speakers was used for training, whereas the standard core test set consisting of 192 utterances spoken by 24 speakers was used for evaluation. The development set is part of the complete test set, consisting of 192 utterances spoken by 24 speakers. Speakers in the training, development, and test sets do not overlap. Details of the data sets are listed in Table. 4.3.

We followed the standard experimentation on TIMIT and collapsed the original 61 phonetic labels in the corpus into a set of 48 phones for acoustic modeling; the latter were further collapsed into the standard set of 39 phones for error reporting. Moreover, the glottal stop [q] was ignored.

HMM-based recognition systems were built using the proposed two MTL-DNN training methods, and they are compared with two kinds of baseline systems: GMM-HMMs and STL-DNN-HMMs.

4.6.1.2 GMM-HMM Baselines

Acoustic models of all phone-based and grapheme-based baseline systems were strictly left-to-right 3-state continuous-density hidden Markov models (HMMs). HMM state emission probabilities were modeled by Gaussian mixture models (GMMs) with at most 16 components. The GMM-HMMs were trained using maximum-likelihood estimation with 39-dimensional PLP acoustic feature vectors extracted at every 10ms over a window of 25ms from the training utterances. Moreover, states were tied using phonetic decision trees and the optimal number of tied states (senones) were determined using the development data set.

In the phone-based system, there were altogether 15,546 cross-word triphone HMMs based on 48 base phones and 587 senones. Phone recognition was performed with a phone bigram LM that was trained only from the TIMIT training transcriptions, and it has a perplexity of 16.44 on the core test set. The grapheme-based system made use of the 26 English alphabets plus the silence symbol as the graphemic labels. Its GMM-HMMs had altogether 760 senones. A grapheme bigram LM was estimated from the training transcriptions, and it has a perplexity of 22.79 on the core test set — which is very high given that there are only 26 letters to recognize!

4.6.1.3 DNN Baseline System

It is a shared view in [72] that filter-bank acoustic features perform better than PLP and MFCC features in DNN-HMM. Therefore, the acoustic features using in our DNN-based TIMIT experiments consist of the first 40 filter-bank coefficients, including c0, and their first- and second-order derivatives. Thus the total dimension of feature vector is 123.

Single-task learning (STL) DNNs were trained to classify the central frame of each 15-frame acoustic context window. Feature vectors in the window were concatenated. All DNNs in our experiments are multi-layer perceptrons (MLPs) with 4 hidden layers and 2048 nodes per layer. The weights of the hidden layers were initialized by unsupervised pre-training a deep belief network (DBN) of the same architecture [28]. The DBN was built by stacking layers of restricted Boltzmann machines (RBMs) on top of one another, and the RBMs were trained one layer at a time. During pre-training, the mini-batch size was kept at 128 (input vectors), and a momentum of 0.5 was employed at the beginning which was then grown to 0.9 after 5 iterations. For Gaussian-Bernoulli RBMs, training kept going for 220 epochs with a learning rate of 0.002, while Bernoulli-Bernoulli RBMs were trained for 100 iterations with a learning rate of 0.02. After pre-training, a softmax layer was added on top of the DBN to obtain the final DNN. The softmax output layers consisted of monophone and/or monographeme states as it is usually found that the use of context-dependent monophones does not give performance gain in DNN-HMM systems for TIMIT. The DNN is now a typical feed-forward MLP and was trained by standard stochastic gradient descent. The targets were derived from the senones of the respective GMM-HMM baseline mod-

els. The whole network was fine-tuned with a learning rate starting at 0.02 which was subsequently halved when performance gain on the validation set was less than 0.5%. Training continued for at least 10 iterations and was stopped when the classification error rate on the development set started to increase. It should be noted that the same DBN can be used to initialize both the triphone STL-DNN and the trigrapheme STL-DNN; they only differ in their output softmax layer.

Standard Viterbi algorithm was used for decoding in all experiments using a bigram language model (LM). All system parameters such as the grammar factor and insertion penalty were tuned using the development data.

4.6.1.4 MTL-DNN Training for Joint Phone & Grapheme Modeling

A single MTL-DNN, labelled as *MTL-DNN-PG*, was trained to estimate the posterior probabilities of both triphone and trigrapheme senones of the language. The construction of the MTL-DNN-PG is very similar to that of an STL-DNN. Firstly, the weights in its hidden layers were initialized by the weights of the same DBN of the corresponding STL-DNNs. But now the output layer in the MTL-DNN-PG consists of two separate softmax layers: one for the primary task and one for the secondary task. For each training sample, two error signals — one from each task's softmax layer — were propagated back to the hidden layers. Thus, the learning rate of the hidden layers was set to half of the original one, while that of the output layer remains the same. Otherwise, the training procedure was the same as that of STL-DNN. In addition, the task weights were set to 0.5 for both tasks as other values did not make much difference in our preliminary experiments.

4.6.1.5 Sequence Training of DNNs

STL-DNN or MTL-DNN trained by minimizing the total frame-wise cross-entropies was employed to generate the numerator and denominator lattices for its own sequence training. The denominator lattice were obtained by performing 30-best recognition using the HTK toolkit. Afterwards, sequence training was performed on top of the well-trained STL-DNN or MTL-DNN by following the procedure described in Section 3.4. It was empirically found that sequence training of STL-DNN might well be started

with a small global learning rate of 1e-5, but sequence training of MTL-DNN required a larger learning rate of 1e-4 to start. This may indicate that the parameter update of joint sequence training of MTL-DNN is more stable so that a larger learning rate may be used. Training continued for at least 5 iterations with learning rate halving, and stopped if no further improvement was observed. In joint sequence training, the likelihood scales and insertion penalties of both tasks were tuned to obtain the least phone error rate on the development set.

During decoding, the insertion penalty was fixed to 0 and the grammar factor was fixed to 1 for all DNN systems.

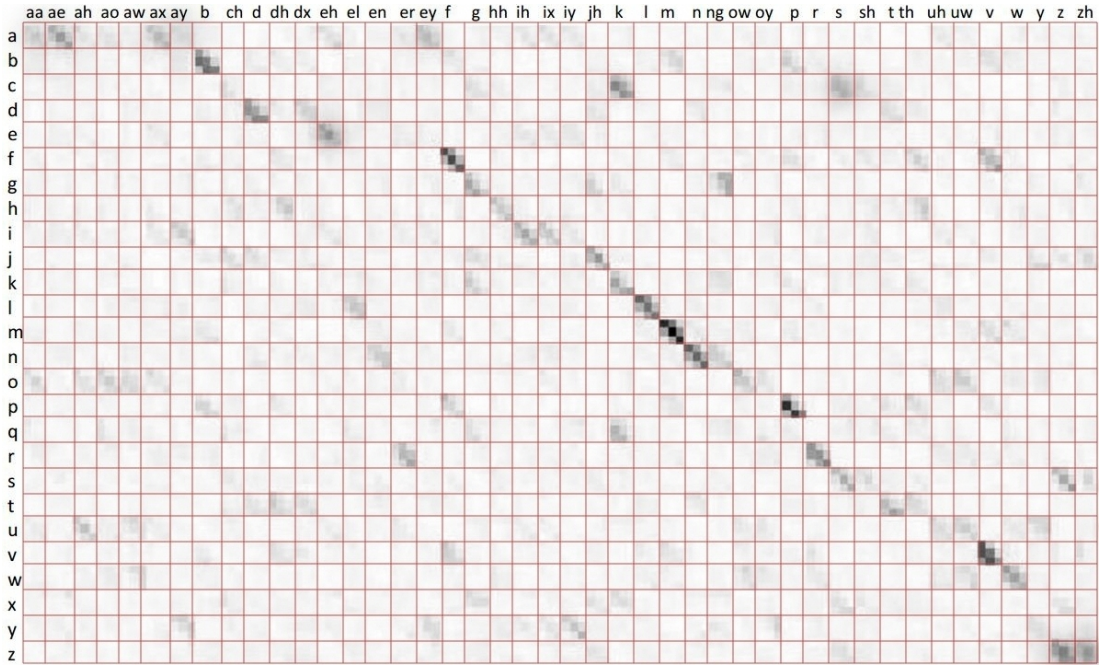

Figure 4.4: The relationship matrix between phone weight vectors (abscissa) and grapheme weight vectors (ordinate) in the MTL-DNN-PG trained on TIMIT.

4.6.1.6 Results and Discussions

Results on the core test set are summarized in Table 4.4.

- We may see that English grapheme recognition is far more difficult than English phone recognition with more than 10% higher error rate. This is expected in English when the estimated grapheme bigram has a perplexity of 22.79; that means the LM does not help much in TIMIT grapheme recognition.

Table 4.4: TIMIT: Recognition performance in terms of phone error rate (PER) and grapheme error rate (GER).

State Model	PER (%)	GER (%)
GMM	28.20	42.64
STL-DNN	21.88	38.11
STL-DNN (MPE/MGE)	21.37	37.14
MTL-DNN-PG	21.29	36.70
MTL-DNN-PG (MPGE)	**20.78**	**36.02**
Baseline DNN in [73]	22.10	-
Baseline DNN in [22]	21.63	-
Best result by LSTM [15]	17.7	-

- The STL-DNN-HMM system again outperforms the GMM-HMM system by a large margin — 22.4% relative in phone recognition[2] and 10.6% relative in grapheme recognition.

- Using grapheme acoustic modeling as a secondary task in MTL-DNN-PG training again helps improve the English phone models and lowers the PER by 2.70% relative. The PER reduction obtained in TIMIT is similar to the WER reduction obtained in Sesotho and siSwati. Thus, we conclude that grapheme acoustic modeling can be a good secondary MTL task for training phone models even for languages in which the relationship between graphemes and phones is not strong.

- To visualize the relationship between the trained English monophone posteriors and grapheme posteriors, we compute the cosines of the angles between any two weight vectors in the output layers — one from the monophone output layer and one from the monographeme layer as Equation 4.4. The results are plotted as a relationship matrix in Fig. 4.4 in gray scale; a darker cell indicates a stronger relationship between the corresponding phoneme and grapheme. The relationships described in the matrix generally agree well with what we expect. For example, according to Fig. 4.4, the letter 'c' is mostly related to the phonemes [k] and [s], while the letter 'f' is mostly related to the phonemes [f], [v] and [th], and so

[2]Our DNN baseline result is comparable with others. For example, one Microsoft group recently reported a PER of 21.63% [22] though a stronger baseline of 20.7% was reported by Hinton's group in [29].

forth. The figure provides some evidence that the MTL-DNN-PG encodes the grapheme-to-phoneme mappings in English.

- Both STL-DNNs are further improved by sequence-discriminative training. MPE training reduces the PER by 0.51% absolute, which is close to the results of MMI training in [66].

- Joint sequence-discriminative training of MTL-DNN-PG gives the best phoneme recognition performance. The absolute gain is 1.10% (or relatively 5.0%) when compared to the STL-DNN baseline, and 0.51% (or relatively 2.4%) when compared to the MTL-DNN-PG trained on minimizing the frame-wise cross-entropy.

4.6.2 WSJ0 Reading Speech Recognition

MTL is known to be more effective when there is limited amount of data, but there is no reason why it may not be applied to a larger ASR task with adequate training data. Here we further evaluate MTL-DNN-PG on the Wall Street Journal speech recognition task (WSJ0) with a larger training set than TIMIT.

4.6.2.1 Speech Corpus and Experimental Setup

Table 4.5: Information of WSJ data sets.

Data Set	#Speakers	#Utterances	#Hours
SI84	83	7,138	15
si_dt_05	10	410	0.8
Nov'92	8	330	0.7

The ARPA WSJ corpus contains samples of continuously read texts from the Wall Street Journal publications and provides training and test materials for speaker-independent continuous ASR in American English. Speech data used in our experiments are recorded in a clean environment using close microphones at 16 kHz rate with 16 bit sample resolution in 1993. It has different vocabulary size setting ranging from 5K to 20K, depending on different subtasks. Standard bigram and trigram language models are provided by D. Paul from MIT Lincoln Lab.

The standard WSJ0 [74] SI-84 training set with 15 hours of speech was used for acoustic modeling. Evaluation was performed on the standard Nov92 5K vocabulary non-verbalized test set, and the si_dt_05 data set was used as the development set for tuning system parameters. All the data are reading news speech. Evaluation was performed with both bigram and trigram language models trained from the transcription of the training data by the SRILM toolkit [75].

4.6.2.2 Acoustic Modeling

Steps and configurations of feature extraction, GMM-HMM training, and DNN-HMM training are exactly the same as those in the TIMIT task. Different from TIMIT experiments, we use senone (tied-state) as DNN output, and extract MFCC features from the data. Triphone GMM-HMMs with 1254 tied-states and 32 Gaussian mixtures per state was found to be optimal, while the best number of tied-states for trigrapheme acoustic modeling is 1489.

Table 4.6: WSJ0: WERs (%) of various systems. Figures in [ ] are WER reductions (%) over the phonetic GMM-HMM baseline.

State Model	Units	Bigram	Trigram
GMM	triphones	6.7	4.2
	trigraphemes	12.9	9.6
STL-DNN	triphones	5.1 [23.9]	3.2 [23.8]
	trigraphemes	8.6	6.5
MTL-DNN-PG	triphones only	**4.5** [32.8]	**2.7** [35.7]
	trigraphemes only	7.7	5.8
Best result in Kaldi [76]	triphone	-	3.5

4.6.2.3 Results and Discussions

Table 4.6 lists the results on the Nov92 test set. The results clearly showed that when other experimental settings are the same:

- Trigram LM outperforms bigram LM by $\sim$ 35% overall;
- STL-DNN outperforms GMM by $\sim$ 24% ;

- Phone-based acoustic modeling outperforms grapheme-based acoustic modeling by more than 40%!

However, similar to what is observed on the TIMIT phone recognition task, even though trigrapheme acoustic modeling is far inferior to triphone modeling for English, training the two tasks together using our MTL-DNN-PG method benefits both of them, reducing the WER of triphone STL-DNN by 15.6% (11.8%) with trigram (bigram) language model. Meanwhile the grapheme-based modeling tasks are significantly improved. It is encouraging that the proposed method works on larger English task, especially under the condition that the baseline system already has a low word error rate.

4.6.3 Lwazi Low-resource Speech Recognition

MTL techniques are believed to be more prominent on smaller training set, and previous study on grapheme-based modeling show it is comparable to or even better than phone-based modeling on many low-resource languages. Therefore, we lastly move to real low-resource speech recognition tasks, which have only a few hours of training data and poor phonetic dictionary, and expect the proposed MTL method will relieve the data scarcity problem.

4.6.3.1 The Lwazi Speech Corpus

The Lwazi project was set up to develop a telephone-based speech-driven information system in South Africa. In the project, the Lwazi ASR corpus [77] was collected over a telephone channel from approximately 200 native speakers for each of the 11 official languages in South Africa. Each speaker produced approximately 30 utterances, in which 16 of them are phonetically balanced read speech and the remainders are elicited short words such as answers to open questions, answers to yes/no questions, spelt words, dates, and numbers. A 5,000-word pronunciation dictionary was also created for each language, which covers only the most common words in the language. Thus, for the phone-based experiments, the *DictionaryMaker* software [78] was used to generate dictionary entries for the uncovered words in the corpus. DictionaryMaker took each of the latter as a seed dictionary and extracted a set of grapheme-to-phoneme con-

version rules for the language to generate the pronunciations of the uncovered words. These automatically generated pronunciations were directly used without any modifications.

Table 4.7: Number of phonemes and graphemes of 3 South African languages and the test-set perplexities of their LMs.

Language	#Phonemes	#Graphemes	LM Perplexity
Afrikaans	37	31	11.18
Sesotho	41	25	19.69
siSwati	41	25	10.94
Universal set	67	30	-

Table 4.8: Details of various Lwazi data sets. OOV means "out-of-vocabulary" and "-S" means "small training set".

Data Set	#Spkr	#Utt	Dur(hr)	Vocab	OOV
Afrikaans:					
Train-S	160	1,195	0.82	1,159	—
Train	160	4,784	3.37	1,513	—
Dev	20	600	—	870	0.89%
Eval	20	599	—	876	0.97%
Sesotho:					
Train-S	162	1,206	1.43	1,513	—
Train	162	4,826	5.70	2,360	—
Dev	20	600	—	1,096	1.86%
Eval	20	601	—	1,089	2.29%
siSwati:					
Train-S	156	580	1.02	1,833	—
Train	156	4,643	8.38	4,645	—
Dev	20	599	—	1,889	6.14%
Eval	20	596	—	1,851	4.53%

Three languages were selected from the corpus in our evaluations. They are Afrikaans, Sesotho, and siSwati. Afrikaans is a Low Franconian, West Germanic language, originated from Dutch; Sesotho is a Southern Bantu language, closely related to other languages in the Sotho-Tswana language group; SiSwati is also a Southern Bantu language, but is more closely related to the Nguni language group. Thus, the three chosen languages come from different language families. The numbers of phonemes and

graphemes in the three languages and the size of the corresponding universal phoneme and grapheme sets are shown in Table 4.7. Since the corpus does not define an official training, development, and test set for each language, we followed the partitions used in [45]. In addition, in order to evaluate the efficacy of MTL in the scenarios where acoustic data is scarce, smaller data sets consisting of approximately one hour of speech were further created by randomly sampling from the full training set of each language. Care had been taken to ensure that there are roughly the same number of utterances for each speaker. Details of the various data sets are listed in Table 4.8. An LM was trained for each language using only the transcriptions in the training set of its language. The test-set perplexities of these LMs are given in Table 4.7.

4.6.3.2 DNN Systems

The experimental setup and procedure used to build the various models were very similar to the ones in TIMIT experiments. Since the data in the corpus are recorded in noisy channel, speaker-based cepstral mean subtraction and variance normalization were applied to the extracted PLP features before they were used.

Table 4.9: Lwazi: WERs (%) of **MONO-LINGUAL** systems trained on the full training sets. Figures in ( ) are #senones and figures in [ ] are WER reductions (%) over the phonetic GMM-HMM baseline.

State Model	Units	Afrikaans	Sesotho	siSwati
GMM	triphones	9.3 (641)	24.4 (741)	20.2 (339)
	trigraphemes	10.6 (728)	24.3 (543)	20.0 (931)
STL-DNN	triphones	7.2 [22.6]	20.1 [17.6]	18.0 [10.9]
	trigraphemes	8.0	20.4	18.2
MTL-DNN-PG	triphones only	**6.4** [31.2]	**19.5** [20.1]	**17.5** [13.4]
	trigraphemes only	7.6	19.8	18.0
ROVER: triphone STL-DNN + trigrapheme STL-DNN		6.7 [28.0]	19.7 [19.3]	**17.4** [13.9]
ROVER: MTL-DNN triphones + same MTL-DNN trigraphemes		**6.2** [33.3]	**19.3** [20.9]	**17.0** [15.8]

Table 4.10: Lwazi: WERs (%) of **MONO-LINGUAL** systems trained on ∼1-hour small training sets. Figures in ( ) are #senones and figures in [ ] are WER reductions (%) over the phonetic GMM-HMM baseline.

State Model	Units	Afrikaans	Sesotho	siSwati
GMM	triphones	12.5 (514)	30.0 (722)	27.1 (271)
	trigraphemes	14.5 (210)	27.7 (324)	24.6 (243)
STL-DNN	triphones	9.5 [24.0]	23.1 [23.0]	21.4
	trigraphemes	11.8	23.5	19.8 [26.9]
MTL-DNN-PG	triphones only	**8.9** [28.8]	**22.1** [26.3]	20.6
	trigraphemes only	11.3	23.1	**18.9** [30.3]
triphone MTL-DNN2		8.7 [30.4]	21.9 [27.0]	18.8 [30.6]
ROVER: triphone STL-DNN + trigrapheme STL-DNN		9.2 [26.4]	22.4 [25.3]	19.3 [28.8]
ROVER: MTL-DNN triphones + same MTL-DNN trigraphemes		**8.7** [30.4]	**21.8** [27.3]	**18.4** [32.1]

4.6.3.3 Results and Discussions

The evaluation was first performed using the full training data set of each language, and then repeated with the reduced training sets to investigate the effect of limited amount of training data on MTL. The recognition performances of the MTL-DNN-PGs are compared with the corresponding GMM-HMM baselines, STL-DNN baselines, and the ROVER integration (using maximum confidence) of the triphone and trigrapheme STL-DNNs, as well as the ROVER integration of the triphone models and trigrapheme models derived from the MTL-DNN-PGs, and they are listed in Table 4.9 and Table 4.10. We have the following observations:

- For all the three languages, when the full training data sets were used for acoustic modeling, both triphone and trigrapheme GMM-HMMs give similar recognition performance. Similar findings were reported in [42] and [43] though the latter used larger amounts of training data (8–80 hours) than what are available in the Lwazi corpus (3–8 hours). Among the three languages, the GMM-HMMs perform the best in Africaans and the worst in Sesotho even though the amount of training data is the least in Africaans and the highest in siSwati. The results may be partly explained by the highest LM perplexity in Sesotho. Moreover, it probably means that the acoustic manifestations of the phones and graphemes in

Africaans are less confusable.

- When the training data sets were reduced to about an hour, the recognition performance in all three languages drops as expected. However, trigrapheme models start to outperform the triphone models in siSwati and Sesotho. One reason may be that there are much fewer graphemes than phonemes in the two languages: the ratio is 1:1.6 in these two languages but is 1:1.2 in Africaans. Thus, the trigrapheme models were better trained than the triphone models with the smaller amount of data. In fact, the better performance disappears when the full training set was used. The finding again supports the use of graphemic acoustic models in low-resource ASR.

Table 4.11: Number of model parameters when the models were estimated using the reduced data sets (in millions). Models in last 5 rows will be described in next chapter.

State Model	Units	Afrikaans	Sesotho	siSwati
GMM	triphones	0.650	0.913	0.343
	trigraphemes	0.265	0.410	0.307
STL-DNN	triphones	14.8	15.3	14.3
	trigraphemes	14.2	14.4	14.3
MTL-DNN-PG		15.3	15.9	14.8
SHL-MDNN	triphones	16.9		
	trigraphemes	15.4		
ML-MTL-DNN-UPS		17.2		
ML-MTL-DNN-UGS		15.6		
ML-MTL-DNN-UPS-UGS		19.0		

- All phone-based and grapheme-based STL-DNN-HMMs outperform their GMM-HMM counterparts by 9–25% relative in the full training sets, and 15–24% relative in the reduced training sets. The amount of performance gains are typical in large-vocabulary ASR (e.g., [31]) and here we show that such gains can also be obtained in low-resource ASR. This is surprising given that the number of model parameters in STL-DNNs is generally much greater than that in GMMs. Table 4.11 shows the number of model parameters[3] in the various kinds of state models estimated using the reduced training data sets in the three languages. It

[3]The figures do not include HMM transition probabilities but only parameters describing HMM state probability distributions.

can be seen that the STL-DNNs are bigger than the GMMs by more than an order of magnitude. We attribute the robust estimation of the large number of DNN parameters to the effective initialization of the DNN weights by the corresponding pre-trained DBN and/or the effective discriminative fine-tuning of the parameters by back-propagation without overfitting them.

- After MTL was applied to jointly training the triphone and trigrapheme posteriors in a single MTL-DNN, compared with the corresponding STL-DNN, word error rates (WERs) were further reduced by 3–9% absolute in the full set and 3–5% absolute in the reduced set. Consistent performance gain is observed for both the larger and smaller training sets, and in both the primary and secondary tasks. The results show that MTL benefits learning of not only the primary task but also the secondary task, and it is still effective with even an hour of training speech. Furthermore, the gains are obtained with no additional language resources.

- The triphone models derived from the MTL-DNN-PGs even outperform the ROVER integration of the corresponding triphone and trigrapheme STL-DNNs (except for the case of using the reduced set in siSwati where the trigrapheme models derived from the MTL-DNN-PG is better). This shows that knowledge transfer between multiple learning tasks can be done more effectively by MTL than ROVER integration. Nevertheless, ROVER may still take advantage of any complementary residual errors made by the triphone and trigrapheme models derived separately from the MTL-DNN-PGs and gives the best recognition performance by integrating them. At the end, the best results reduce the WERs of the GMM-HMM baselines by 16–33% relative in the full training set and 27–32% relative in the reduced training set.

- Triphone MTL-DNN2 trained with the reduced small training set , which is a simple extension of the basic MTL-DNN with the addition of another STL-MLP, gives a performance that is almost as good as the ROVER integration of the triphone and trigrapheme MTL-DNN-PGs, even on siSwati where triphone DNNs are inferior to the corresponding trigrapheme DNNs.

To see the generalization effect of MTL-DNN-PG training, we look at the frame classification errors over both the reduced training and development data sets after

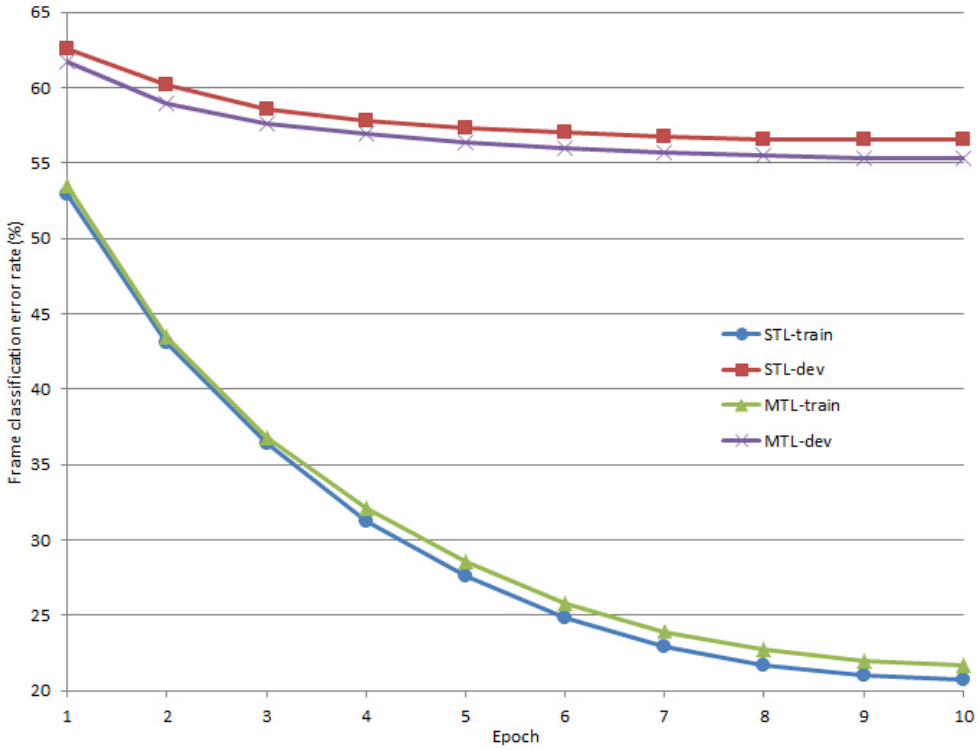

Figure 4.5: Frame classification error rates of STL- and MTL-DNN on the Lwazi training and development sets of Sesotho during back-propagation.

each back-propagation epoch during both STL-DNN training and MTL-DNN training. The results for Sesotho are plotted in Fig. 4.5; similar behaviors are also found for Afrikaans and siSwati. The plots clearly show that although MTL-DNN-PG training converges to a worse local optimum than STL-DNN training for the training data, it performs better on the unseen development set. Thus, we may conclude that the extra grapheme acoustic modeling task really provides a representation bias to a better local optimum which generalizes better for unseen data.

4.7 Summary

This chapter illustrates the first MTL method we propose for ASR — joint training of phone and grapheme acoustic models using MTL-DNN. We believe that MTL-DNN learns the implicit phone-to-grapheme mapping, and we analyze the model to verify our assumption in the experiments. Two extensions are further proposed to improve the performance. One of them is to use outputs of the secondary task in the MTL-DNN as the inputs to another MLP to train the primary task, following the suggestion in [18].

The other one is to adopt sequence training criteria of STL-DNN to MTL-DNN.

Consistent experimental results on TIMIT English phone recognition, WSJ0 reading English speech recognition and recognition of three low-resource South African languages demonstrate that, by sharing the hidden layers, both tasks get improved compared to STL training. Analysis of task relationship embedded in the parameters of the MTL-DNN confirms our claim that MTL-DNN learns the implicit phone-to-grapheme mapping.

CHAPTER 5

DISTINCT TRIPHONE ACOUSTIC MODELING: MORE CONTEXTUAL INFORMATION

The second method presented in this thesis aims at solving a common data scarcity problem in ASR — robustly modeling of a huge number of distinct context-dependent acoustic units.

Most modern DNN-based ASR systems employ tied-states as the acoustic units. Although state tying reduces the amount of data needed to estimate triphones by sharing their parameters, triphones states tied together are indistinguishable during decoding. With fixed amount of training data, detailed acoustic modeling and robust estimation look contradictory: the more detailed granularity of acoustic units we use, the fewer training data that an individual acoustic unit has, causing overfitting easily. Is it possible to fulfill both requirements? We propose the concept of robust distinct triphone modeling to model context variability as detailed as possible without sacrificing generalization to unseen data. Starting from a common well-trained tied-state system, we try to boost the number of acoustic units and model parameters, while sharing as few parameters of distinct triphone states with tied-states as possible.

5.1 Previous Works using GMM-HMM

Despite of the success of state tying, researchers keep exploring better methods for context-dependent acoustic modeling. There have been approaches addressing the quantization error brought by state tying for GMM-HMM, such as:

- **Triphone-By-Decomposition** [79]

  The basic idea of the triphone-by-decomposition approach is to estimate parameters of triphones from models with less contextual information. For example, in model interpolation [80], the state distribution of a triphone is represented as

a linear interpolation of its ML esitmate and the state distributions from its corresponding left and right context-dependent models, and optionally the context-independent model. In quasi-triphones [81], the first state of a three-state triphone only depends on the triphone's left context, and the last state only depends on the right context, while the middle state is assumed to be totally context-independent. Back-off acoustic modeling [82] is a more recent method borrowing the idea from back-off language modeling to acoustic modeling. It combines the score of a triphone with those from triphones estimated under broad phonetic class contexts.

- **Basis Approach** [83]

  In basis approach, usually one or more bases are derived and then model parameters are represented as a combination of the basis functions or vectors so that the number of parameters to estimate is greatly reduced. Semi-continuous hidden Markov model (SCHMM) [84] and subspace Gaussian mixture model (SGMM) [85] represent state probabilities using a pool of Gaussians, while the bases in Bayesian sensing HMM [86] are sets of state-dependent basis vectors. The eigentriphone method [87] follows the eigenface [88] and eigenvoice [89] concepts in computer vision and speaker adaptation. For each cluster of untied triphone states, a set of Eigentriphones is first derived as the basis by weighted PCA. Then, each triphone state in the cluster is projected into the space spanned by it using regularized maximum likelihood decomposition (MLED). By this way, each triphone state has a set of unique parameters which enhances discrimination of the acoustic models so that the modeling power of tied-state systems is improved.

Table 5.1: Relationship between the four methods for distinct triphone modeling and speaker adaptation.

Basis	Distinct Triphone Modeling	Speaker adaptation
Eigenvectors	Eigentriphone	Eigenvoice [89]
Reference vectors	Reference model weighting	Reference speaker weighting [90]

In this section, we first illustrate our previous reference model weighting (RMW) method for GMM-HMM before we apply the same idea to DNN-HMM. RMW is

motivated by reference speaker weighting (RSW) in the speaker adaptation problem. Its relationship with eigentriphone method, is analogical to the relationship between eigenvoice and RSW in speaker adaptation. All of the four methods lsited in Table 5.1 aim at robustly estimating GMM parameters given a small amount of training or adaptation data, by projecting the GMM parameter supervectors to a lower dimensional space spanned by a set of basis, so that the number of parameters to estimate is reduced a lot. Similar to eigentriphone, RMW seeks robust distinct triphone modeling over tied-state systems, but the process of deriving the bases and projecting triphone states into the lower dimensional space is much more straightforward.

Reference model weighting can be applied over any group of context-dependent acoustic units such as triphones from the same base phone. In [87], the cluster-based eigentriphone modeling (ETM) is found to outperform the model-based and state-based ETM methods. Therefore we follow the application of cluster-based ETM method and apply RMW over clusters of states. The leaf nodes in the same state-tying tree are chosen as state clusters. For a given task, a development set of speech data will be used to determine the optimal choice of nodes empirically using the method described in Section 2.5. Although the nodes selected for conventional tied-state HMMs and cluster-based RMW come from the same phonetic decision tree, they need not be the same for the two procedures.

Based on the state clustering results, a conventional tied-state triphone HMM system is first trained as usual. Each triphone model is a 3-state left-to-right HMM, and each state is an M-component GMM. Then the selected nodes are regarded as state clusters, and for each state cluster q we repeat the following procedure to find the reference states and to project the member states as a linear combination of the reference states.

STEP 1 : Clone the tied-state GMM to all the member states which have at least 3 training samples in the state cluster q. From now on the parameters of the triphone states are not tied any more.

STEP 2 : Re-estimate only the Gaussian means of the cloned triphone states in STEP 1. At the same time, collect the zeroth- and first-order statistics on the training data of each Gaussian component m of state j in state cluster q — that is, its soft occupa-

tion count, $\sum_t \gamma_{qjm}(t)$, and its mean vector, $\sum_t \gamma_{qjm}(t)\mathbf{x}_t$, where $\mathbf{x}_t$ is the acoustic vector at frame t. Furthermore, the soft occupation count for each state j may be computed by summing up the occupation counts of all its mixture components as $\sum_t \sum_m \gamma_{qjm}(t)$. We will call the resulting system the *untied triphone state (UTS) HMM system*.

STEP 3 : Based on a threshold θ on the occupation count, split the member states of the cluster Ω_q into two groups: the frequent state set Ω_q^F and the infrequent state set Ω_q^I.

STEP 4 : Stack up the M Gaussian means $\{\boldsymbol{\mu}_{qjm}, m = 1, \ldots, M\}$ of state j in the frequent state set Ω_q^F according to their order in the original tied-state GMM onto a Gaussian mean supervector $\mathbf{v}_{qj} \equiv [\boldsymbol{\mu}'_{qj1}, \boldsymbol{\mu}'_{qj2}, \cdots, \boldsymbol{\mu}'_{qjM}]'$. In addition, a Gaussian mean supervector $\mathbf{v}_{q0}$ is constructed similarly for the tied state which will be indexed by $j = 0$.

STEP 5 : Form the set of reference models, or more specifically, the reference state supervectors, Ω_q^R, using the mean supervectors from the tied state and the frequent states. That is, $\Omega_q^R \equiv \{\mathbf{v}_{qj} : j = 0 \cup j \in \Omega_q^F\}$.

STEP 6 : Take the set of reference state supervectors Ω_q^R as a basis, and assume that all infrequent state supervectors of cluster q lie in the vector space spanned by the basis. Let $\mathbf{B}_q = [\mathbf{v}_{q0}\, \mathbf{v}_{qj_1} \cdots \mathbf{v}_{qj_{K_q}}]$ be the matrix of the basis vectors, where $j_k \in \Omega_q^F$ and $K_q = |\Omega_q^F|$ is the number of the reference models in cluster q. The Gaussian mean supervector $\mathbf{v}_{qi}$ of each infrequent state $i \in \Omega_q^I$ is modeled as

$$\mathbf{v}_{qi} = \sum_{j \in \Omega_q^R} w_{qij} \mathbf{v}_{qj} = \mathbf{B}_q \mathbf{w}_{qi} \tag{5.1}$$

where $\mathbf{w}_{qi} = [1\ w_{qij_1} \cdots w_{qij_{K_q}}]'$ is the (interpolation) weight vector of the infrequent state i. Note that the weight for the tied-state mean supervector $\mathbf{v}_{q0}$ is fixed to 1; $\mathbf{v}_{q0}$ is treated as a bias for the estimation of $\mathbf{v}_{qi}$.

STEP 7 : Estimate the weight vector $\mathbf{w}_{qi}$ by maximizing the following log-likelihood

$L(\mathbf{w}_{qi})$ of its training data after removing all the irrelevant terms:

$$-\sum_{t,m}\gamma_{qim}(t)(\mathbf{x}_t-\boldsymbol{\mu}_{qim})'\mathbf{C}_{qm}^{-1}(\mathbf{x}_t-\boldsymbol{\mu}_{qim}) \tag{5.2}$$

where $\mathbf{C}_{qm}$ is the covariance matrix of the mth Gaussian component of the original tied state that corresponds to state cluster q.

STEP 8 : Substitute Eqn. (5.1) to Eqn. (5.2) and take its first order derivative. Setting the derivative to zero, we have

$$\begin{aligned}
&\sum_{t,m}\gamma_{qim}(t)\mathbf{B}'_{qm}\mathbf{C}_{qm}^{-1}(\mathbf{x}_t-\mathbf{B}_{qm}\mathbf{w}_{qi})=0\\
\Rightarrow\quad &\mathbf{w}_{qi}=\left[\sum_m\left(\sum_t\gamma_{qim}(t)\right)\mathbf{B}'_{qm}\mathbf{C}_{qm}^{-1}\mathbf{B}_{qm}\right]^{-1}\\
&\qquad\left[\sum_m\left(\sum_t\gamma_{qim}(t)\mathbf{x}_t\right)\mathbf{B}'_{qm}\mathbf{C}_{qm}^{-1}\right]
\end{aligned} \tag{5.3}$$

where $\mathbf{B}_{qm}$ is the sub-matrix of $\mathbf{B}_q$ when only the rows corresponding to the mth Gaussian component of the reference mean supervectors are considered.

During the estimation process, only the Gaussian means of the infrequent states are re-estimated. The other HMM parameters such as the Gaussian covariances, transition probabilities, and mixture weights are not updated; that is, they are the same as the baseline tied-state HMMs. Meanwhile, unseen triphones are still tied.

In STEP 3 of the basic procedure, one has to classify a state as frequent or infrequent based on a fixed threshold θ on its occupation count. Such hard decision does not take into account the wide distribution of occupation counts among the states. In addition, it is more logical to put more weight to reference models/states that are better trained with more data.

Hence the RMW procedure is further enhanced by

- using *all* states as reference states, and the mean vectors of *all* of them are re-estimated using Eqn. (5.1). Thus, the hard binary decision of frequent or infrequent states is avoided.

- penalizing the likelihood function with the addition of a regularization term that varies according to the occupation counts of the states: greater penalty for states with small counts and smaller penalty for states with large counts. The regularization term is necessary, otherwise the re-estimated model will degenerate to the untied-state HMM due to the maximum likelihood principle.

The following penalized log likelihood function was tried:

$$\hat{L}(\mathbf{w}_{qi}) = L(\mathbf{w}_{qi}) - \sum_{k \in \Omega_q} \frac{\beta}{2 \sum_{t,m} \gamma_{qim}(t)} ||w_{qik}||^2 \tag{5.4}$$

where β is the regularization parameter. The closed-form solution is given by

$$\begin{aligned} \mathbf{w}_{qi} &= \left[\sum_m \left(\sum_t \gamma_{qim}(t) \right) \mathbf{B}'_{qm} \mathbf{C}^{-1}_{qm} \mathbf{B}_{qm} + \mathbf{R} \right]^{-1} \\ & \quad \left[\sum_m \left(\sum_t \gamma_{qim}(t) \mathbf{x}_t \right) \mathbf{B}'_{qm} \mathbf{C}^{-1}_{qm} \right] \end{aligned} \tag{5.5}$$

where $\mathbf{R}$ is a diagonal matrix, and

$$\mathbf{R} = \frac{\beta}{\sum_{t,m} \gamma_{qim}(t)} \cdot \mathbf{I}_{|\Omega_q| \times |\Omega_q|} \, .$$

5.2 Distinct Triphone Acoustic Modeling by MTL-DNN

For DNN-HMM, we investigate distinct triphone acoustic modeling under the multi-task learning (MTL) framework. Phonetic units of different granularities are jointly trained within a single acoustic model simultaneously. We show that reference model weighting on DNN is equivalent to inserting an extra linear layer, and the outputs are further combined in this layer to achieve best performance. The proposed methods can be readily applied on top of any already trained tied-state GMM-HMM or DNN-HMM ASR systems for further improvement, thus benefit from synthesizing unseen triphones by the phonetic regression tree.

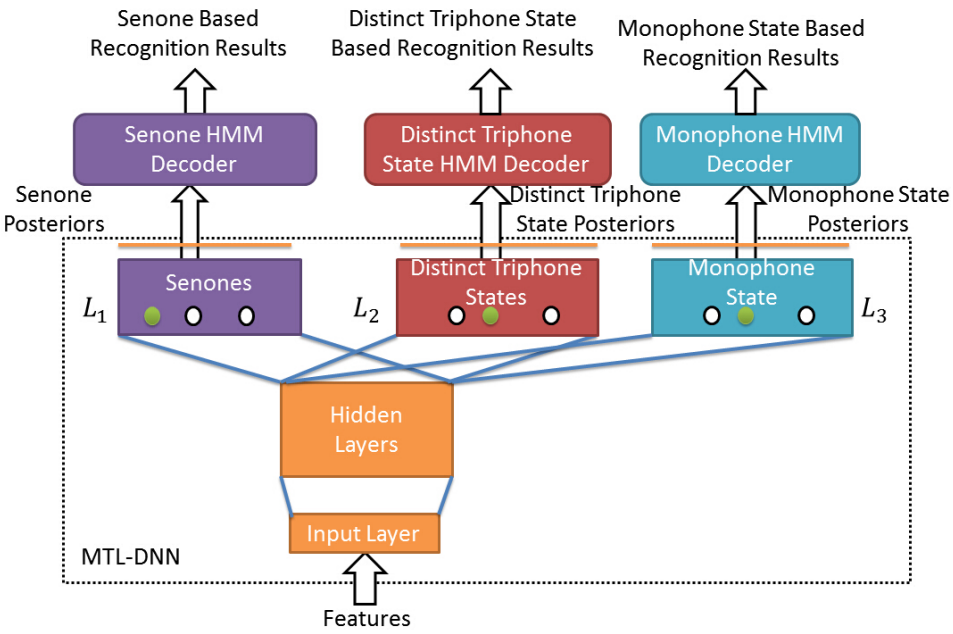

Figure 5.1: The MTL-DNN-DTM used for the joint training of monophone states, senones (or tied states) and distinct triphone states (DTS). The horizon bars represent the softmax activation.

5.2.1 Joint Training of Different Kinds of Acoustic Units

Due to the large number of distinct triphone states (DTS) in common tasks (when compared with the number of tied states or senones), standard single-task learning (STL) of DNNs does not work well, resulting in overfit models with poor recognition performance on unseen test data. Regularization is required, and in this thesis, we investigate the use of highly related task(s) as the regularizer(s) under the framework of multi-task learning of DNN. The resulting networks will be called MTL-DNNs. Our approach again strictly follows Caruana's two MTL requirements as the two MTL methods in Chapter 3 and 4: for each training frame, multiple output units are activated and multiple tasks are learned in paralle; the tasks share the same acoustic observations and all hidden layers in a DNN.

Section 1.2 suggests a heuristic guideline to select the extra task(s) for a primary task in MTL: select task(s) that may exploit extra information from the training data of the primary task. Here, senone modeling is taken as the primary task $\mathcal{T}_1$, and the following two extra tasks are selected for its MTL training: DTS modeling task $\mathcal{T}_2$, and monophone states modeling task $\mathcal{T}_3$. The three tasks are obviously related as they all represent acoustic modeling at different phonetic resolutions. The extra information

is the implicit membership of the DTS's in the senones and monophones, and more contextual information. We call the new model for distinct triphone modeling as multi-task learning deep neural network for distinct triphone modeling (MTL-DNN-DTM).

Fig. 5.1 shows the MTL-DNN-DTM used for training the three tasks together. Basically, three single-task learning DNNs (STL-DNNs) which estimate the posterior probabilities of monophone states, senones, and distinct triphone states (DTS) are merged together so that they their inputs and all hidden layers, while each of them keeps its own output layer (L_1, L_2 and L_3). The MTL objective function is to minimize the sum of cross entropies of the three tasks over the training set $\mathcal{D}$:

$$\mathcal{E}(\mathcal{D}, \Lambda) = -\sum_{\mathbf{x} \in \mathcal{D}} \left(\sum_{j=1}^{3} \sum_{i=1}^{N_j} d_i^{(j)} \log P(s_i^{(j)} | \mathbf{x}; \lambda_0, \lambda_j) \right) \tag{5.6}$$

where $d_i^{(j)}$ is the target value of the ith state in $\mathcal{T}_j$, λ_0 consists of model parameters that are shared by *all* tasks (i.e., the hidden layers) and λ_j consists of model parameters specific to only task $\mathcal{T}_j$ (i.e., the output layers). During training, for each input acoustic vector, 3 output units, one for each task, will be activated. Thus, the shared hidden layers help the tasks regularize each other to achieve better generalization.

Table 5.2: Forced alignment results and the mapping to monophone state, tied-state and distinct triphone state labels.

Frame range	Triphone[state index]	Monophone state	Tied-state	Distinct triphone state
...	...	...	...	...
34-37	v-ih+n[3]	ih_3	ST_ih_3_8	v-ih+n_3
38-39	v-ih+n[4]	ih_4	ST_ih_4_5	v-ih+n_4
40-42	ih-n+eh[2]	n_2	ST_n_2_6	ih-n+eh_2
43-46	ih-n+eh[3]	n_3	ST_n_3_1	ih-n+eh_3
...	...	...	...	...

In practice, starting from a conventional GMM-HMM, we go through the following steps to construct and train the proposed MTL-DNN.

STEP 1 : A conventional senones (tied-states) GMM-HMM system with optimal number of senones is first built.

STEP 2 : The senone GMM-HMM system is used to forced-align the training and development data to get the frame labels for DNN training. Table 5.2 demonstrates part of the forced alignment result of an utterance, and the mapping to monophone state, tied-state and DTS labels. Each frame is firstly assigned a label of `triphone[state index]`. Then, this label is converted to the 3 kinds of labels according to the mapping. Thus, each frame has 3 labels totally: `monophone state`, `tied-state` and `DTS`.

STEP 3 : Pre-train a DBN by contrast divergence [28].

STEP 4 : Add an output softmax layer of senones, and train the senones posteriors by standard DNN back-propagation as common tied-state DNN-HMM.

STEP 5 : Add an output softmax layer of monophone states and initialize it with random weights.

STEP 6 : Similar to the RMW procedure running on GMM-HMM systems, treat each senone as a state cluster and untie it to get all its member DTS's. Add an output softmax layer consisting of all those DTS's that have at least 10 training samples[1]. The output weight vector of a DTS is initialized by the well-trained weight vector of its corresponding senone.

STEP 7 : Train the resulting MTL-DNN by standard back-propagation like an STL-DNN except that the learning rate of the hidden layers is set to $1/3$ of that for STL-DNN, since 3 error signals are now propagated back to the hidden layers. The learning rate of the output layers remains the same.

STEP 8 : During decoding, posteriors for the three kinds of output units are generated separately and fed into their corresponding HMM decoders.

5.2.2 Transformation of DTS Activations by RMW

When reference model weighting (RMW) is applied to distinct triphone modeling using GMM-HMMs, each triphone state is represented as a supervector created by stacking up the mean vectors of its Gaussian mixtures. Then for each state cluster (which is

[1]For unseen triphone states and those DTS's with fewer than 10 samples, they will be still represented by the appropriate senones.

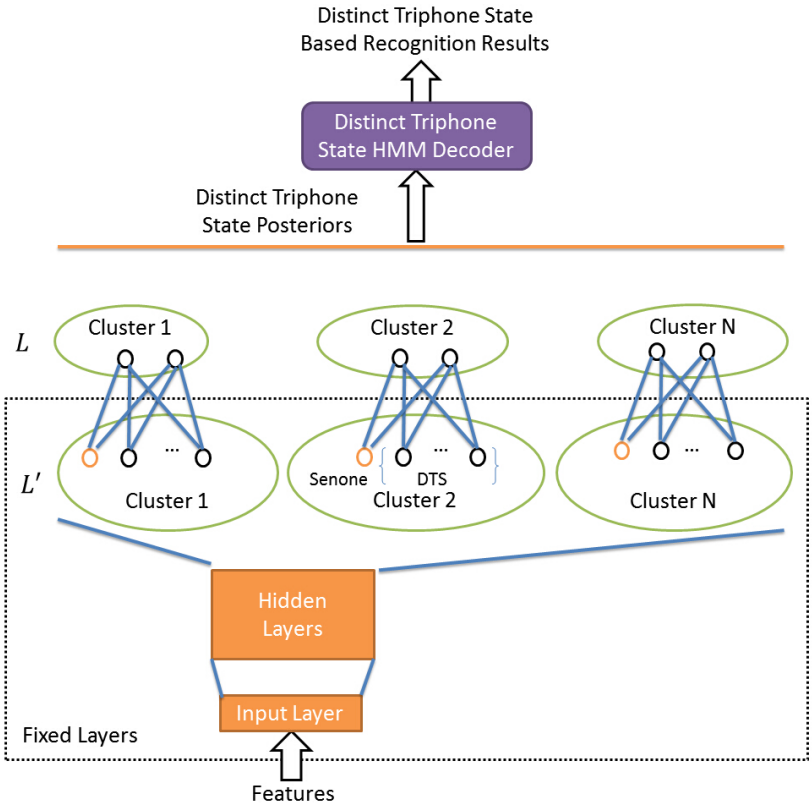

Figure 5.2: Transforming the activation of each DTS using the RMW technique.

equivalent to a tied state in practice), all or a subset of its member DTS supervectors are collected to form a basis and all DTS's of the cluster can then be expressed as a linear combination of the basis vectors. Since the number of such basis vectors is much smaller than the dimension of the state supervectors, the combination weights may be estimated robustly even with very small amount of training samples.

In the DTS DNN, each output node represents a DTS. Let's assume that there are N_k DTS's $\{s_i^{(k)} : i = 1, 2, \ldots, N_k\}$ in the state cluster represented by the kth senone; there are H hidden units in the last hidden layer, and their outputs (including the bias) are represented by $\mathbf{y} = [1, y_1, y_2, ..., y_H]'$. Let's also denote the DNN weight vector connecting the last hidden layer to $s_i^{(k)}$ as $\mathbf{w}_i^{(k)} \in \mathbb{R}^{H+1}$. Thus, its activation is given by

$$a_i^{(k)} = \mathbf{y}'\mathbf{w}_i^{(k)} \ . \tag{5.7}$$

When applying RMW for robust training of DTS DNNs, the set of DTS DNN weight vectors of the kth state cluster, $\{\mathbf{w}_0^{(k)}\} \cup \{\mathbf{w}_i^{(k)} : i = 1, 2, \ldots, N_k\}$, where $\mathbf{w}_0^{(k)}$ is the DNN weight vector of the kth senone, is treated as a basis. Then the DNN weight

vector of each member DTS is re-modeled as a linear combination of the basis vectors as follows:

$$\hat{\mathbf{w}}_i^{(k)} = \mathbf{W}^{(k)}\mathbf{u}_i^{(k)} \tag{5.8}$$

and the new activation $\hat{a}_i^{(k)}$ of the $s_i^{(k)}$ unit is

$$\hat{a}_i^{(k)} = \mathbf{y}'\mathbf{W}^{(k)}\mathbf{u}_i^{(k)} \tag{5.9}$$

where $\mathbf{W}^{(k)} = [\mathbf{w}_0^{(k)}\mathbf{w}_1^{(k)}\ldots]$ and $\mathbf{u}_i^{(k)}$ is the combination weights for the DTS unit $s_i^{(k)}$. Therefore, if we represent the activations to all $\{s_i^{(k)}\}$ by the vector $\mathbf{a}^{(k)}$, and all their combination weight vectors by $\mathbf{U}^{(k)}$, then the new activation vector after RMW is given by

$$\hat{\mathbf{a}}^{(k)} = (\mathbf{W}^{(k)}\mathbf{U}^{(k)})'\mathbf{y} = \mathbf{U}^{(k)'}\mathbf{W}^{(k)'}\mathbf{y} = \mathbf{U}^{(k)'}\begin{bmatrix} a_0^{(k)} \\ \mathbf{a}^{(k)} \end{bmatrix} \tag{5.10}$$

where $a_0^{(k)}$ is the activation of senone.

Eq. (5.10) demonstrates that the application of RMW on the connection weight vectors between the last hidden layer and the DTS output layer is equivalent to applying a linear transform $\mathbf{U}^{(k)} \in \mathbb{R}^{(N_k+1)\times N_k}$ on the activations to DTS output units. The proposed RMW method can be implemented by adding an additional RMW layer between the last hidden layer and the DTS output layer of the DNN as shown in Fig. 5.2.

In practice, this procedure is done as follows:

STEP 1 : Since the senone units and DTS units are in different output layers in the MTL-DNN, we need to firstly merge the two output layers, and remove the softmax operations on them;

STEP 2 : Group the DTS from the same state cluster together with the senone output unit of that cluster, as is showed by the layer L' in Fig. 5.2;

STEP 3 : On top of it, add a block diagonal linear transformation, in which each block is the combination weight matrix for a state cluster.

STEP 4 : A softmax operation is added over all DTS's finally.

5.2.3 Estimation of the Combination Weights

To estimate the additional RMW combination weights for each state cluster, all the network weights between layers below L' of a well-trained DNN are fixed. Each RMW combination weight vector is initialized as $\mathbf{u}_i^{(k)} = [1, 0, 0, \ldots]$ where the unity value corresponds to the combination weight to the senone vector which will be fixed during the estimation. Since there are many parameters in the additional layer, L2 regularization was tried. In some preliminary TIMIT experiments, we empirically found that similar results could be achieved by simply re-estimating each DTS weight vector as a linear combination of its original vector and its senone weight vector as below:

$$\hat{\mathbf{w}}_i^{(k)} = \mathbf{w}_0^{(k)} + \alpha \mathbf{w}_i^{(k)} \tag{5.11}$$

where α is a global combination weight for all DTS's which is tuned using a development set. Therefore we employed this simple method in the following experiments.

5.3 Experiments

Following the experiment settings in previous Chapter, the proposed MTL-DNN-DTM as described above was tested on Three English speech recognition tasks. They are the TIMIT phone recognition task, the WSJ0 word recognition task and SWitchBoard (SWB) large vocabulary telephony speech recognition task. The performance of MTL-DNN-DTMs on TIMIT and WSJ0 task are compared with their respective STL-DNN baselines which were prepared as described in Chapter 4. We further examined MTL-DNN-DTM on the SWB large vocabulary speech recognition task, which is more difficult and complex than TIMIT and WSJ0.

5.3.1 TIMIT Phone Recognition

5.3.1.1 Acoustic Modeling

The same single-task learning DNN (STL-DNN) modeling monophone states in Chapter 4 is used as baseline system. Beside it, we further trained a senone STL-DNN. Then triphone states with at least 10 training samples were untied from their senones. At the end, a set of 9823 distinct triphone states (DTS) were obtained which is $\sim$17

times of the optimal number of senones. The network weights of the DTS STL-DNN were initialized from the senone STL-DNN and trained as describe in previous section. MTL-DNNs were then built by jointly training at least two of the three kinds of acoustic units (monophone states, senones, and DTS's) together. During testing, outputs of different acoustic units were computed independently and fed into corresponding decoders.

Finally, the DTS activations were further transformed using the RMW technique as described by Eq. (5.11). The optimal value of the global parameter α was determined using the development data, and 0.1 was found to give the best results. For comparision, STL-DNNs modeling different acoustic units are also trained.

Table 5.3: Phone recognition errors (%) of different DNN systems. M, S and D represent systems using monophone states, senones and distinct triphone states as output units respectively.

Models	M output	S output	D output
STL-DNN (M)	21.88	—	—
STL-DNN (S)	—	21.11	—
STL-DNN (D)	—	—	24.70
MTL-DNN (MS)	21.64	20.82	—
MTL-DNN-DTM (SD)	—	20.40	23.28
MTL-DNN-DTM (MSD)	21.58	19.99	22.26
+ RMW activations	—	—	19.70

5.3.1.2 Experiment Results and Discussion

Table 5.3 summarizes the recognition performance of various DNN systems. We have the following observations:

- Among three STL-DNNs, senone STL-DNN has the best performance, as it successfully trades off detailed modeling between monophone states and DTS's.
- Joint training of different kinds of acoustic units using MTL-DNN-DTMs between both the primary task as well as the extra task(s). DTS's were more robustly trained due to the regularization effect from the senones and monophone states which were well trained and thus might provide a representation bias towards a better local optimum for DTS modeling. On the other hand, senones

were also better trained due to the extra contextual information provided by the DTS's.

- Although after MTL training, DTS's still perform the worst, the proposed RMW-based transformation of the DTS activations could successfully re-estimate their parameter robustly. Compared with the senone DNN baseline, the final DTS DNN system reduces the phone recognition errors by 6.7%.

Table 5.4: WSJ0: Word error rates (%) of various DNN-HMM systems using bigram/trigram language models.

Models	M output	S output	D output
STL-DNN (M)	8.07/—	—	—
STL-DNN (S)	—	5.10/3.21	—
STL-DNN (D)	—	—	5.17/3.46
MTL-DNN-DTM (MSD)	7.22/—	4.75/2.80	4.48/2.76
+ RMW activations	—	—	4.33/2.64

5.3.2 WSJ0 Reading Speech Recognition

5.3.2.1 Acoustic Modeling

For WSJ0 experiments, we also employ the same baseline STL-DNN system we used in Chapter 4. Then, we untied the state clusters (tied-states) to obtain the DTS's. There are 22407 DTS's that have at least 10 training samples since the training set of WSJ0 task is much larger than that of TIMIT. This number is about 18 times the number of senones. Moreover, during recognition, the posterior probabilities of the output units were first scaled by the their statistical priors counted from the training data before they were fed into their corresponding decoders for Viterbi decoding.

5.3.2.2 Results and Discussion

Experiment results of the various DNN-HMM systems using bigram and trigram language models are presented in Table 5.4. [2]

[2]Since the HTK toolkits used in our experiments don't support decoding with context-independent acoustic model and trigram language model, the recognition results based on monophone state outputs and trigram language model are not provided.

- Different from the experiment results of the TIMIT task, the performance of DTS DNNs are comparable with senone DNNs and are much better than the monophone state DNN. An analysis on the occurrences of the DTS's over the test sets show that there are fewer infrequent DTS's in the WSJ0 test set than that in the TIMIT test set.

- MTL training consistently reduces the word error rate (WER) of all tasks simultaneously.

- RMW transformation of the DTS activations yields the best performance for DTS DNNs, resulting in a relative WER reduction of 15.1% (17.8%) over the senone STL-DNN baseline using a bigram (trigram) language model.

5.3.3 SWB Telephony Speech Recognition

SWitchBoard (SWB) [91] is a large vocabulary telephony conversational speech recognition data set. In the past decade, various of ASR techniques have been proved effective on LVCSR tasks. We aim to see how the proposed distinct triphone acoustic modeling methods work with state of art ASR techniques for building LVCSR systems.

5.3.3.1 Speech Corpus and Experimental Setup

The SWitchBoard-1 release 2 data set contains around 290 hours of US English conversational speech on telephone line by 500 speakers. The 4870 conversation sides are further spliced into 259,890 utterances for acoustic modeling.

Recognition results are reported on the standard Hub5 2000 evaluation set. It consists of 1,831 SWitchBoard-2 telephony conversation sides and 2,628 CallHome telephony conversation sides. There are totally 2 hours of conversational speech.

5.3.3.2 GMM-HMM Baseline

We followed the Kaldi recipe [3] setup to build the GMM-HMM baseline systems. The first 5 hours of speech in the data set are used as development data, while the rest 286 hours of speech data are used for model estimation.

[3] kaldi/egs/swbd/s5b/run.sh

A trigram language model was trained on all the transcription text of the training set defined above using the SRILM toolkit [92]. It has perplexity of 89.6. Scoring is done by the NIST Scoring Toolkit (SCTK). Training and decoding were performed with a dictionary with 30k pronunciation entries.

Firstly, conventional MFCC features were extracted, with conversation side-based cepstral mean and variance normalization. They were enhanced by concatenating the delta and delta delta coefficients to form 39-dimensional feature vectors. A tied-state context dependent GMM-HMM was first trained using only one third of the training data.

Then, seven consecutive feature vectors, each consisting of 13 static MFCC coefficients, were concatenated, and then were reduced to 40-dimensional feature vectors using linear discriminant analysis (LDA) [93] with the frame labels obtained by the model trained in the previous step. After that, maximum likelihood linear transform (MLLT) [94] is applied on the LDA feature vectors. Another tied-state GMM-HMM was estimated on all transformed training data.

Finally, speaker adaptive training (SAT) [95] were applied. The resulted model has 8704 tied states and 200k Gaussians in total. It is used as the maximum likelihood GMM-HMM baseline in our experiments.

5.3.3.3 Training DNN-HMMs

We started from the ML GMM-HMM tied-state system to train the senone DNN-HMM baseline system using the Kaldi recipe. First of all, The ML GMM-HMM tied-state system is employed to perform forced-alignment on all utterances in the SWB-1 data set. Then, MLLR transforms are estimated per speaker. The features are transformed and stored for DNN training.

The DBN pre-training process is similar to that in the TIMIT and WSJ0 experiments. Now with more training data, we pre-trained a deeper network with 6 hidden layers. The number of training epoch is only 1 for Bernoulli RBM and 2 for Gaussian RBM. Moreover, the global learning rate in the recipe is smaller than those in the TIMIT and WSJ0 experiments. Afterwards, 8704 output units representing senones in the ML GMM-HMM tied-state system were added on top of the pre-trained DBN to

form a DNN. Common STL-DNN was then trained by minimum cross entropy criterion.

We untied the senones to 60,000 DTS's by the Kaldi toolkit. MTL training was applied to estimate senone DNN and DTS DNN simultaneously. The learning rate for the hidden layers was again set to half of that for the output layers. Lastly, DTS activations are transformed by RMW described in Equation 5.11.

5.3.3.4 Results and Discussion

Table 5.5: Recognition word error rate (%) of various DNN-HMM systems on the Hub5 2000 evaluation set using trigram language model.

Model Description	SWB	CHE	Total
ML GMM	22.2	36.8	29.5
Senone STL-DNN	14.6	26.9	20.8
DTS STL-DNN	14.3	26.6	20.5
MTL-DNN-DTM (senone output)	14.3	26.5	20.4
MTL-DNN-DTM (DTS output)	13.9	26.0	19.9
MTL-DNN-DTM + RMW activations	13.7	25.8	19.8

Table 5.5 presents the word error rates of STL- and MTL-DNNs on the Hub5 2000 evaluation set. The detailed WERs on SWB-2 and CallHome subset are also listed.

- Different from what is observed on the TIMIT and WSJ0 tasks, STL-DNN using DTS as output units outperforms common tied-state STL-DNN. This is not surprising since the SWB data set has many more training data.
- The senone and DTS outputs of MTL-DNN outperform their counterpart STL-DNN. It indicates that MTL training does benefit both tasks.
- After applying RMW activation transformation, we are able to further reduce the WER. The overall relative WER reduction is 4.8% compared to the senone STL-DNN or 3.4 % compared to the DTS STL-DNN baseline.

5.4 Summary

State tying has been a canonical technique for context-dependent acoustic modeling. Triphone states are tied together to share training data, and unseen triphones can be synthesized according to phonetic state-tying decision tree. However, triphone states that are tied together lose their unique characters, since all of their model parameters are exactly the same. Our previous work on distinct triphone GMM-HMM represents the parameters of a state as the weighted combination of all states in the same state cluster by a regularized maximum likelihood formula. By this way, the number of parameters to estimate is greatly reduced.

In this chapter, we follow the same idea, and aim at modeling distinct triphone states in DNN-HMM systems. First, distinct triphone states are trained with senones in an MTL-DNN. The DTS modeling task embeds more contextual information into the hidden layers, while the senone modeling task in return prevents the training of the complex DTS modeling task from overfitting. Then, we add a block diagonal linear transform on top of the MTL-DNN, which is equivalent to representing the DTS's as a weighted combination of the senone and the member DTS's in the same cluster. Significant improvement is observed on TIMIT phone recognition, WSJ0 reading speech recognition and SWB telephony speech recognition.

Currently, the linear combination step is working in a very simple way as Equation 5.11. In our experiments, even if the transformation matrix is constrained to be block diagonal, overfitting happens easily when we train these weights. Our method differs from the ensemble learning method in [73] where model integration is performed *after* the softmax function. The key difference is that [73] simply uses a least mean-squares objective, and the solution can be obtained in one step, while we still need to train this combination layer by gradient decent. Whether the methods in [73] help to improve the MTL-DNN-DTM remains to be explored.

CHAPTER 6

MULTI-LINGUAL ACOUSTIC MODELING OF LANGUAGE-SPECIFIC TRIPHONES AND UNIVERSAL PHONES: BENEFIT FROM LINGUISTICS

In previous chapters, it is assumed that we are given only resources of a single language. When resources from other languages are available, cross-lingual or multi-lingual ASR techniques may be used to improve recognition accuracy [47]–[23]. Some of previous works share the hidden layers of DNN of different languages, while they have independent output layers. In our second method, we propose to improve multi-lingual acoustic modeling by further exploiting the relationship among the phones from multiple languages via a universal phone set derived by linguistic knowledge in the MTL framework without directly defining the mappings between them. The MTL-DNN is expected to learn the multi-lingual phone mappings and benefits all language-specific acoustic modeling tasks.

6.1 Multi-lingual ASR

Despite of the huge amount of data we can attain for popular languages, for many languages in the world it is still difficult to get enough data to train decent large vocabulary speech recognition systems, especially for low-resource languages with only a few hours of transcribed speech data. In such situation, multi-lingual acoustic modeling is an effective way to share data and transfer knowledge among multiple languages.

For GMM-HMM acoustic models, [96] directly maps phones from multiple languages together according to the universal phone set which is a multi-lingual phone inventory and does detailed studies on the issues of multilingual ASR. Although multi-lingual training benefits from more training data compared to language-specific modeling, experiment results reveal that such direct mapping of phones across languages can not improve ASR accuracy over language-specific modeling, due to the loss of

language-specific phone identity. Moreover, context-dependent units like senones in the multi-lingual model are trained to cover all languages and thus are less precise to a specific language compared with senones in the language-specific model.

On the other hand, multi-lingual subspace Gaussian mixture model (SGMM) [97] was proved to be more dedicate and effective. SGMM establishes all phonetic states on a common Gaussian mixture model structure, in which the means and mixture weights lie in a subspace of the total parameter space. Such model structure allows more compact and automatic representation of the model parameters, and is more robust given small amount of training data compared to standard GMM. In a multi-lingual setting, the common subspace GMM structure is shared across multiple languages, while each language-specific phone still keeps its identity by holding distinct substate mixture weights [97].

For ANN-based acoustic modeling, [98] and [99] estimate shallow neural networks with bottleneck features using multi-lingual data, and then build language-specific systems on these bottleneck features by a tandem approach. Improvement is observed on eight European languages. In the era of DNN, [100] use multi-lingual data for DBN pre-training, but the later DNN fine-tuning is still performed on language-specified data. The method was then improved by switching output layers of different languages during training, but special attention still needs to be paid to the training order of languages.

Almost at the same time, in 2013, researchers from Microsoft and Google proposed a more flexible model for multi-lingual acoustic modeling using DNN [57, 49]. Fig. 6.1 displays its structure. Multiple languages share the hidden layers, but they keep their own output layers. The hidden layers are trained by data from multiple languages during both pre-training and fine-tuning steps, while the output layers are trained by only language-specific data. During training, multi-lingual data are shuffled and mixed, to get rid of the issue on the training order of different languages. For a training frame, only the output units in the output layer of the language it belongs to are activated, while other output layers are not affected by its training signal. However, these methods are purely data-driven, and does not make use of existing linguistic knowledge such as International Phonetic Alphabet, which is a waste.

Table 6.1: The universal phone set (UPS) and the phonemes' usage in three South African languages.

IPA	UPS	Afr	Ses	siS
Affricates				
tɸ	tp_b	×	×	✓
ʧ’	tS_^	×	✓	×
ʧ^h	tS_h	×	✓	×
ʦ’	ts_^	×	✓	✓
ʦ^h	ts_h	×	✓	✓
kx	kx	×	✓	×
kɬ	kK_^	×	×	✓
ps^h	ps_h	×	✓	×
pʃ^h	pS_h	×	✓	×
pʃ’	pS_^	×	✓	×
ʤ	d_0Z	×	✓	✓
dz	dz	×	×	✓
dβ	dB	×	×	✓
Fricatives				
f	f	✓	✓	✓
v	v	✓	×	✓
s	s	✓	✓	✓
z	z	✓	×	✓
ʃ	S	✓	✓	✓
x	x	✓	✓	✓
h	h	×	×	✓
ɦ	h_b	✓	✓	✓
ɬ	K	×	✓	×
ɮ	K_b	×	×	✓
fʃ	fS	×	✓	×
Clicks				
ǀ	!!_b	×	×	✓
ǃ	!_b	×	✓	×
Diphthongs				
əi	@i	✓	×	×
œy	u_y	✓	×	×
əu	@u	✓	×	×
iə	i@	✓	×	×
uə	u@	✓	×	×
Trills&Flaps				
r	r	✓	✓	✓
—	sil	✓	✓	✓

IPA	UPS	Afr	Ses	siS
Stops				
p	p	✓	×	✓
p^h	p_h	×	✓	✓
p’	p_^	×	✓	×
b	b	✓	✓	✓
t	t	✓	×	✓
t^h	t_h	×	✓	×
t’	t_^	×	✓	×
d	d	✓	×	✓
k	k	✓	×	✓
k^h	k_h	×	✓	✓
k’	k_^	×	✓	×
g	g	✓	×	✓
tl’	tl_^	×	✓	✓
tl^h	tl_h	×	✓	✓
Nasals				
m	m	✓	✓	✓
n	n	✓	✓	✓
J	J	×	✓	✓
N	N	✓	✓	✓
Vowels				
i	i	✓	✓	✓
y	y	✓	×	×
u	u	✓	✓	✓
I	I	×	✓	×
ɸ	eu_:	✓	×	×
ɛ	E	✓	×	✓
ɔ	O	✓	✓	✓
a	a	✓	✓	✓
a:	A:	✓	×	×
ə	@	✓	×	×
æ	{	✓	×	×
ʊ	U	×	✓	×
œ	u_	✓	×	×
Approximants				
l	l	✓	✓	✓
j	j	✓	✓	✓
w	w	✓	✓	✓

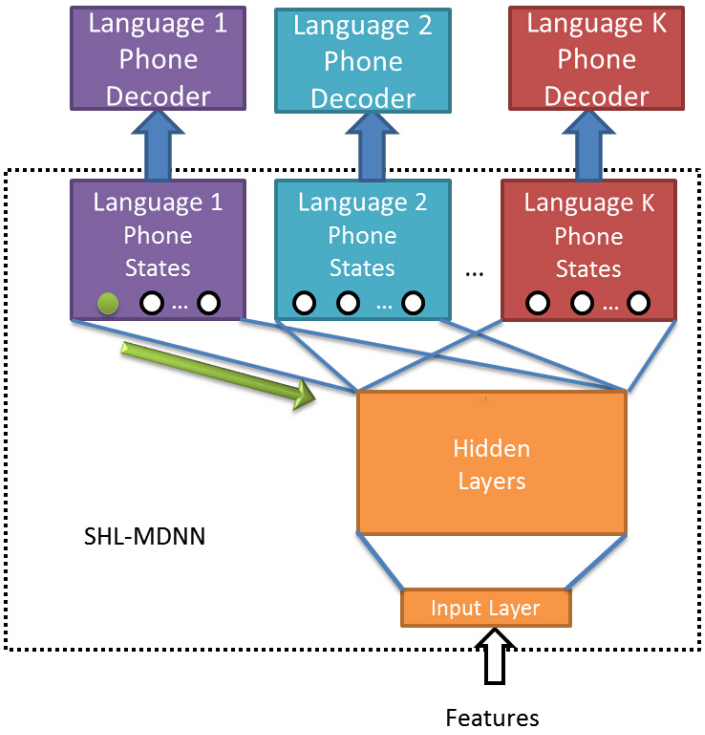

Figure 6.1: Multi-lingual deep neural network with shared hidden layers.

6.2 Universal Phone Set (UPS)

Many multi-lingual ASR techniques utilize a global phone inventory such as the International Phonetic Alphabet (IPA) [101], or a smaller *universal phone set* (UPS) which can be written in the ASCII format and is derived from the IPA. During multi-lingual acoustic modeling, phones from different languages having the same UPS phonetic symbol will share their training data. In this chapter, the three South African languages under investigation come from the Lwazi project [77] which already provides their IPA phoneme sets [102]. Thus, we simply unify their phoneme sets (after removing any duplicates) to form the UPS. Table 6.1 shows the final UPS of 67 phonemes and their uses in the three South African languages in our experiments. Phonemes of the 3 languages are highly overlapping, indicating close relationship between the languages. It is believed that the UPS modeling task is related to language-specific phone modeling tasks. Thus, we investigate learning them together using MTL-DNN.

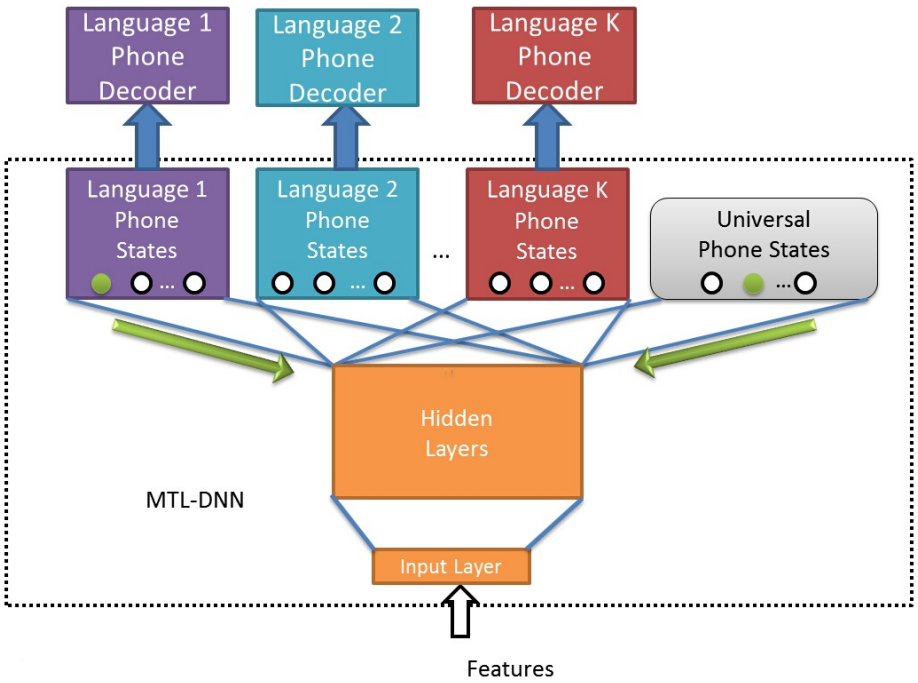

Figure 6.2: A multi-lingual MTL-DNN system (ML-MTL-DNN-UPS) with shared hidden layers and an extra output layer of UPS states. Outputs, labelled as green, from 2 separate tasks are turned "on" by an input acoustic vector.

6.3 Multi-lingual MTL-DNN With an Extra UPS Learning Task

To develop a multi-lingual ASR system with K languages, $K+1$ tasks will be jointly learned in our multi-lingual MTL-DNN system:

$\mathcal{T}_1$: posteriors of triphone senones of the first language

$\mathcal{T}_2$: posteriors of triphone senones of the second language

$\vdots$

$\mathcal{T}_K$: posteriors of triphone senones of the Kth language

$\mathcal{T}_u$: posteriors of monophone states of the UPS

With respect to the kth language, $\mathcal{T}_k$ is the primary task and the remaining K tasks $(\mathcal{T}_1, \ldots, \mathcal{T}_{k-1}, \mathcal{T}_{k+1}, \ldots, \mathcal{T}_K, \mathcal{T}_u\})$ are the secondary tasks with the UPS task $\mathcal{T}_u$ being a common task for the learning of *any* language. The proposed multi-lingual

MTL-DNN system is shown in Fig. 6.2. The model architecture is similar to SHL-MDNN in [49, 57] except for the additional output layer for learning the posteriors of UPS monophone states; we will call our multi-lingual MTL network *ML-MTL-DNN-UPS*. Without the additional UPS task, one simply hopes that the shared hidden layers will automatically captures the phonetic-acoustic relationships among the multiple languages during MTL-DNN training so that the recognition performances of all languages are enhanced. Our additional UPS learning task forces this to happen: the weights in the shared hidden layers are trained to cause acoustic vectors from different languages that are mapped to the same UPS phone to activate the same target in the UPS output layer. Moreover, instead of directly defining $O(K^2)$ phone mappings between any two of the K languages, one only needs to map the phones of each of the K languages to the UPS phones.

Let's denote the training data set from the kth language as $\mathcal{D}^{(k)}$ and its set of N_k triphone senones as $s_i^{(k)}$, $i = 1, \ldots, N_k$, and $k = 1, \ldots, K$. Similarly, the set of N_u UPS monophone states is denoted as $s_i^{(u)}$. For each input vector $\mathbf{x}^{(k)} \in D^{(k)}$, only two tasks are involved: the triphone senones of the kth language ($\mathcal{T}_k$) and the UPS monophone states ($\mathcal{T}_u$) are activated using the softmax function of Eq. (4.1). Their corresponding per-frame cross-entropies, $\mathcal{E}_k(\mathbf{x}^{(k)}; \lambda_0, \lambda_k)$ and $\mathcal{E}_u(\mathbf{x}^{(k)}; \lambda_0, \lambda_u)$, where λ_u consists of the weights in the output layer of the UPS states, are given by Eq. (4.2). Finally, the training objective function over all data of the multiple languages is modified from Eq. (3.1) as follows:

$$
\begin{aligned}
&\mathcal{E}(\{\mathcal{D}^{(k)}\}, \Lambda) \\
&\quad = \sum_{k=1}^{K} \sum_{\mathbf{x}^{(k)}} \left(\beta_k \, \mathcal{E}_k(\mathbf{x}^{(k)}; \lambda_0, \lambda_k) + \beta_u \, \mathcal{E}_u(\mathbf{x}^{(k)}; \lambda_0, \lambda_u) \right) \\
&\quad = \underbrace{\sum_{k=1}^{K} \left(\beta_k \sum_{\mathbf{x}^{(k)}} \mathcal{E}_k(\mathbf{x}^{(k)}; \lambda_0, \lambda_k) \right)}_{\text{as in SHL-MDNN}} \\
&\qquad + \beta_u \underbrace{\left(\sum_{k=1}^{K} \sum_{\mathbf{x}^{(k)}} \mathcal{E}_u(\mathbf{x}^{(k)}; \lambda_0, \lambda_u) \right)}_{\text{regularization}} .
\end{aligned}
\tag{6.1}
$$

Eq. (6.1) shows that our multi-lingual MTL-DNN training is different from SHL-MDNN training and may be considered as a regularized version of the latter — a form of regularized MTL [103]. If the language task weights are large, it will be the same as SHL-MDNN training; if the UPS task weight is large, it will be reduced to UPS training. Since UPS models are usually not as good as language-specific models [96], the learned UPS output layer will not be used in recognition, and it is only used to help enforce the cross-lingual phone mappings during MTL-DNN training. The training procedure of ML-MTL-DNN-UPS is similar to that of MTL-DNN-PG in Chapter 4.

From the perspective of regularization, we prefer simpler regularizer and thus we use UPS monophone states instead of UPS triphone senones as the common task. In some preliminary experiments, we also empirically found that they gave similar results.

6.4 Extensions

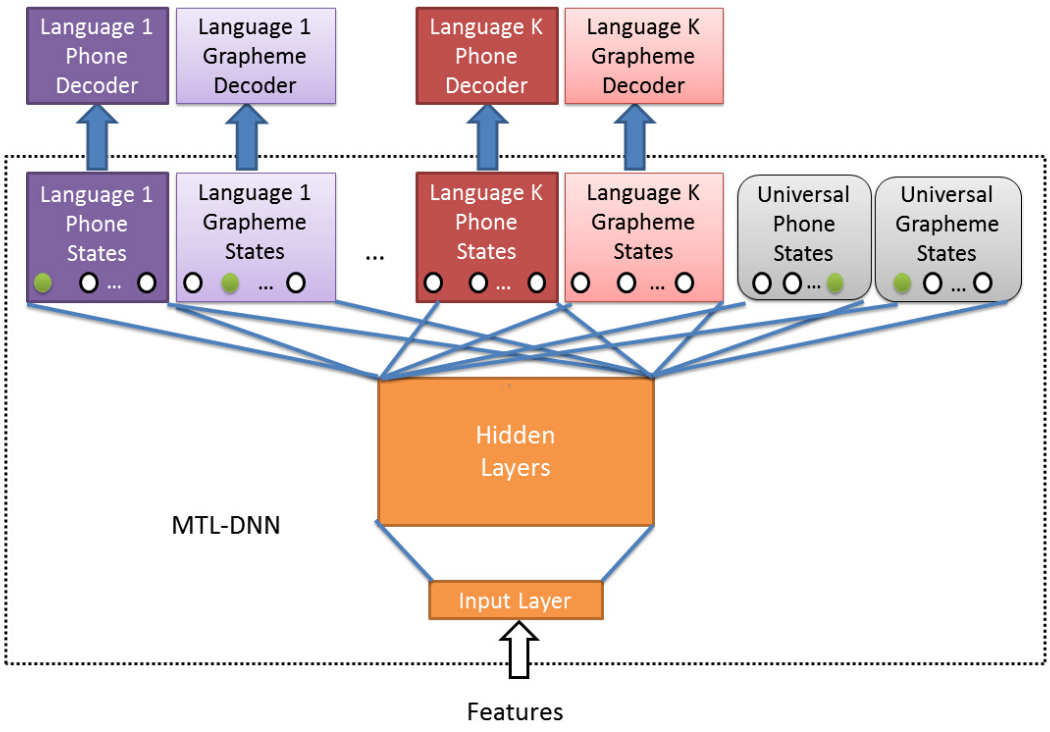

Figure 6.3: A multi-lingual MTL-DNN system (ML-MTL-DNN-UPS-UGS) modeling the triphone and trigrapheme senones of K languages, together with their universal phone states and universal grapheme states.

As said in Chapter 4, grapheme-based acoustic modeling is a viable solution for low-resource language ASR. Method 2 can be easily modified to use graphemes in-

stead of phones as the modeling units. A universal grapheme set (UGS) may again be created by simply taking the union of the grapheme sets of all the languages under investigation. The UGS for the three languages in our experiments consists of 30 graphemes including one that denotes silence. We will call the grapheme-based multi-lingual MTL-DNN with the extra UGS learning task *ML-MTL-DNN-UGS*.

Obviously, one may further combine Method 1 and Method 2 to jointly model multi-lingual phones and graphemes using the UPS and UGS as the extra learning tasks, and we will label such network as *ML-MTL-DNN-UPS-UGS*. As is shown in Fig. 6.3, if there are K languages to learn simultaneously, then there will be totally $2K + 2$ (softmax) output layers in the model. Each input acoustic vector from the kth language will activate four targets: one triphone senone and one trigrapheme senone of the kth language, one universal (mono)phone state, and one universal (mono)grapheme state.

6.5 Experiment Evaluation

The Lwazi corpora provides data sets from a group of low-resource languages. The data were recorded under similar environment and channel, with the same format. Thus it provides a suitable experimental setting to examine multi-lingual ASR approaches. MTL-DNN with universal phone modeling task was evaluated and compared with the existing SHL-MDNN on multi-lingual acoustic modeling of the three South African languages tested in last chapter.

6.5.1 MTL-DNN Training

The training of the various DNNs in Method 2 is similar to that in Method 1 except that now the training data of all the three languages were pooled together to jointly train their acoustic models. Specifically, the following models were trained and compared:

- *ML-STL-DNN*: multi-lingual STL-DNN of universal (mono)phones or (mono)graphemes;
- *SHL-MDNN*: multi-lingual phonetic shared-hidden-layer DNN [57] (with a total of 3 learning tasks);

- *ML-MTL-DNN-UPS*: multi-lingual phonetic MTL-DNN using universal phone modeling as the extra learning task (for a total of 4 learning tasks);
- *ML-MTL-DNN-UGS*: multi-lingual graphemic MTL-DNN using universal grapheme modeling as the extra learning task (for a total of 4 learning tasks); and
- *ML-MTL-DNN-UPS-UGS*: multi-lingual MTL-DNN using universal phone modeling and universal grapheme modeling as the extra learning tasks (for a total of 8 learning tasks).

All these DNNs were initialized from the same DBN which was pre-trained by training data from all the three languages. Afterwards, multiple softmax output layers were added to the DNNs, one for each learning task. Thus, the number of softmax layers in the various DNNs are: 1 for ML-STL-DNN, 3 for SHL-MDNN, 4 for ML-MTL-DNN-UPS/-UGS, and 8 for ML-MTL-DNN-UPS-UGS. As a result, during back-propagation fine-tuning, each training frame will activate 1, 1, 2, and 4 output nodes in ML-STL-DNN, SHL-MDNN, ML-MTL-DNN-UPS/-UGS, and ML-MTL-DNN-UPS-UGS respectively. Because of the use of multiple languages and MTL, some parts of the training procedure were modified. Firstly, the use of data from multiple languages requires the training utterances to be shuffled randomly so that the fine-tuning process would not be biased to a particular language at any time during training. Secondly, since more than one output node may be activated, the learning rate of the weights in the hidden layers were reduced by a factor equal to the number of activated output nodes. Otherwise, the training procedure is the same as that of MTL-DNN in Chapter 4.

6.5.2 Results and Discussions

Table 6.2 and Table 6.3 summarize the recognition performance of the various systems trained on the full training sets and the reduced training sets of all the three languages respectively. Performance of the previous mono-lingual STL-DNNs are repeated in the tables for comparison.

- The performance of multi-lingual STL-DNN (ML-STL-DNN) of the universal phones (graphemes) is far inferior to the triphone (trigrapheme) STL-DNN

Table 6.2: Lwazi: WERs (%) of **MULTI-LINGUAL** systems trained on the full training sets.

State Model	Units	Afrikaans	Sesotho	siSwati
mono-lingual	triphones	7.2	20.1	18.0
STL-DNN	trigraphemes	8.0	20.4	18.2
SHL-MDNN	triphones	6.4	19.7	17.8
	trigraphemes	7.8	19.9	17.6
ML-MTL-DNN				
-UPS	triphones only	6.1	19.0	17.3
-UGS	trigraphemes only	7.5	19.1	17.0
-UPS-UGS	triphones only	**5.6**	**18.9**	17.2
	trigraphemes only	7.3	19.0	**16.8**

Table 6.3: Lwazi: WERs (%) of **MULTI-LINGUAL** systems trained on $\sim$1-hour small training sets.

State Model	Units	Afrikaans	Sesotho	siSwati
mono-lingual	triphones	9.5	23.1	21.4
STL-DNN	trigraphemes	11.8	23.5	19.8
SHL-MDNN	triphones	9.1	22.0	20.2
	trigraphemes	11.4	23.1	19.3
ML-MTL-DNN				
-UPS	triphones only	8.6	21.5	19.3
-UGS	trigraphemes only	11.2	22.2	18.8
-UPS-UGS	triphones only	**8.3**	**21.3**	19.0
	trigraphemes only	11.0	22.0	**18.7**

baseline. Similar finding was reported in [96]. Although the UPS/UGS models may share data among the various languages, the data become impure and they may fail to model the language specificities. Moreover, co-articulatory effects were not modeled as the targets in our ML-STL-DNNs are only monophones/monographemes states.

- On the other hand, multi-lingual models based on SHL-MDNN outperform their STL-DNN counterparts and reduce the WER by ∼2–11% relative in the full training set and ∼4–10% relative in the reduced training set. The improvements agree fairly well with the findings in [57] where the WER reductions are ∼3–5% relative. It is believed that the shared internal representation captures cross-lingual knowledge among the training languages.

- The multi-lingual MTL-DNN (ML-MTL-DNN) with an extra UPS (UGS) output layer further outperforms the corresponding phonetic (graphemic) SHL-MDNN. For example, in the case of reduced training set, the WER reduction improves from ∼4–10% relative in SHL-MDNN to ∼7–12% relative in ML-MTL-DNN-UPS/-UGS. In SHL-MDNN, the benefit of MTL is achieved only by learning common weights in the hidden layers, whereas in our ML-MTL-DNN-UPS/-UGS, the learning of the weights in the output layer of each language is further regularized by the learning of the weights in the output layer of UPS (UGS).

- Finally, based on the results of MTL-DNN-PG and above, we put the learning of phone models and grapheme models of the three languages together with the UPG and UGS, and obtained the best results that reduce the WER by ∼6–22% relative in the full set and ∼8–13% in the reduced set over the STL-DNN baselines. The improvements obtained by the ML-MTL-DNN-UGS-UPS are about twice of that from the respective SHL-MDNNs. All these are obtained without additional language resources.

6.6 Summary

Multi-lingual ASR is one of the important research directions to address the data scarcity problem. This chapter focuses on improving existing multi-lingual deep neural network. Instead of directly mapping language-specific phones to universal phones

and training all of them together in a STL-DNN, we take an indirect approach by further introducing an acoustic modeling task learning a set of universal phones, so that the problems of losing language-specific phone identities and phone context are avoided. Compared to SHL-MDNN, our approach further utilizes the universal phone sets, and forces the MTL-DNN to learn the multi-lingual phone mappings. Thus, it gains extra phone mapping information from linguistics, meanwhile keeping the purity of language-specific phones.

The method can be further combined with joint modeling of phone and grapheme proposed in Chapter 4. In our experiments, we get an MTL-DNN with 8 output layers, and each input frame is used to train 4 tasks at the same time. However, this does not add too much burden to the model size, nor does it degrade training speed much, due to the fact that most parameters in DNN are in the hidden layers.

In previous works, cross-lingual model adaptation has also been proved to be effective. In [57], a well-trained SHL-MDNN is adapted to a novel language. The MTL-DNN in this chapter works for better learning multiple langauges together, and it outperforms SHL-MDNN. Thus, we believe its hidden layers are also able to better transfer knowledge to the novel language. On the other hand, many related works on multi-lingual ASR uses ANN as feature extractor for back-end systems. ML-MTL-DNN-UPS is expected to extract better features, since its hidden layers incorporates information in the UPS, which is from linguistics.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Hybrid deep neural network hidden Markov model (DNN-HMM) outperforms conventional GMM-HMM by a large margin. Multi-task learning (MTL) is a prospective approach to further improve the powerful DNN for speech recognition. In this thesis, we propose and illustrate three methods in the MTL framework using DNN to train phonetic models *without* requiring additional resources, which are among the initial works of applying MTL-DNN to ASR. The resulting phonetic models are believed to generalize better to unseen data because the extra learning task(s) can exploit extra information from the training data to provide a representation bias to the original phonetic modeling task. This is made possible because both the inputs and hidden layers are shared by the multiple learning tasks.

7.1 Contributions

Table 7.1: Summary of three proposed methods in this thesis.

Method	MTL-DNN-PG	MTL-DNN-DTM	ML-MTL-DNN-UPS
Primary task	Monophone or senone modeling	Senone acoustic modeling	Language specific phone modeling
Secondary task	Grapheme acoustic modeling	DTS acoustic modeling	Universal phone modeling
Setting	Mono-lingual	Mono-lingual	Multi-lingual
Extra information	Phone-to-grapheme mappings	Contextual information	Multi-lingual phone mappings

This thesis studies secondary tasks that benefit the training of phonetic acoustic modeling under the MTL framework. Table 7.1 lists the three MTL methods. More specifically, the monophone or senone acoustic modeling task is estimated with three other acoustic modeling tasks based on various kinds of acoustic units:

(a) **Graphemes**: units from the writing system

Unlike phonetic units, graphemes do not directly correspond to acoustic sounds, but may be used as complementary units for acoustic modeling. For single-language ASR we propose using grapheme acoustic modeling as the additional learning task to learn the language's phone models using an MTL-DNN. Even though it is well-known that the grapheme-to-phoneme mappings in English are not simple, we show the method also works well for TIMIT phone recognition task, and even for the WSJ large-vocabulary ASR task where there are adequate amount of training data. We then further evaluated it on recognition tasks of three low-resource South African languages for which MTL is expected to be more effective in previous literatures. Experimental results show that MTL training consistently outperforms common single-task learning (STL) training. Thus, we believe the method can be applied in other general ASR tasks. Furthermore, we analyze and visualize the correlation between task-specific parameters of the two tasks, to verify our assumption that MTL-DNN learns the implicit phone-to-grapheme mappings.

(b) **Distinct triphones**: phonetic units with much more detailed contextual information

Distinct triphone modeling suffers the data scarcity problem, due to the large number of infrequent triphones. In the second method we applied the is reference model weighting (RMW) technique to robust distinct triphone modeling in a DNN-HMM under the multi-task learning (MTL) framework. Distinct triphone states (DTS's) were jointly trained with senones (and perhaps monophone states) using an MTL-DNN. Their joint training has three benefits. Firstly, monophone states modeling and senones modeling tasks provide better inductive bias for the DTS modeling task to reach a better optimum, preventing the excessively complex DTS DNN from overfitting by sharing the hidden layers. Secondly, the DTS modeling task embeds more contextual information into the hidden layers of the MTL-DNN. Lastly, the shared hidden layers make it easy to implement an activation transformation based on the RMW technique by simply adding an additional layer between the last hidden layer and the DTS output layer for each state cluster. Using the method we were able to reduce relative phone or word error rate by 6.7% and 17.8% on TIMIT task and WSJ0 task respectively.

(c) **Universal phones**: phones derived from phone sets of multiple languages

Lastly, when the phone models of multiple low-resource languages were trained together, we proposed using the acoustic modeling of a set of universal phones/graphemes (UPS/UGS) as the additional learning task. From the optimization perspective, the UPS task serves as a regularizer for the phonetic modeling of all the involved languages. From the language perspective, it forces the multi-lingual MTL-DNN to implicitly encode a mapping among the phones of all the languages. Finally, by combining ith with the first method, we were able to reduce the WERs of monolingual STL-DNN baselines by $\sim$8–13% relative when only an hour of training data were available from each of the three South African languages, and $\sim$7–22% relative when 3–8 hours of data are available. Additional memory and computational requirements are only required during MTL training; during recognition, the softmax layer(s) due to any extra tasks may be discarded. Furthermore, since our multi-lingual MTL-DNN has the same architecture as the multi-lingual SHL-MDNN but performs better than the latter, and the latter had been shown to be effective in cross-lingual model adaptation [49, 57, 58], we believe that our multi-lingual MTL-DNN will also perform better in cross-lingual model adaptation as well.

Nevertheless, the contribution of this thesis is not limited to the utilization of particular secondary tasks chosen here. More importantly, we try to answer the question of *how to choose helpful secondary tasks*. Below are the guidelines we impose to choose secondary tasks:

(a) secondary task should be positively related to the primary phonetic modeling task so that they can share the same internal representation;

(b) secondary task shares the same set of inputs with primary task;

(c) secondary task should not require extra language resources;

(d) preparation for training the secondary task(s) should be as convenient as possible.

7.2 Future Works

(a) **Multi-task learning with other regularization methods**

Our MTL methods aim at improving the generalization of phonetic DNNs. There are many other ways to do this, and perhaps the most well-known one is the dropout method [13] which had been applied successfully to low-resource ASR [104] as well. Both our MTL methods and dropout are regularization methods but they use different mechanisms: dropout prevents overfitting by efficiently and approximately combining an exponentially many different neural network architectures, whereas our MTL methods exploit extra information from the data using additional learning task(s) which share(s) some commonality with the primary learning task and provide(s) a representation bias towards a better local optimum. Other ways such as weight pruning [105] and large-margin optimization [106, 107] have also been proposed, and it will be interesting to see if these methods are complementary to our proposed MTL methods.

(b) **Multi-task relationship learning**

Multi-task learning can be a powerful learning method if the tasks involved are truly related. We expect more and more secondary tasks to be explored in near the future to improve ASR performance. If we treat the hidden layers in neural network as high-level feature extractors, training hidden layers of MTL-DNN can be regarded as a kind of multi-task feature learning [108, 109]. Similar to multi-task feature learning, MTL-DNN takes it for granted that the learned tasks are positively related and share a homogeneous feature representation without explicitly modeling the relationship between them.

In this thesis, the multiple tasks are carefully sought and their positive relationships are assumed, based on common knowledge. In the future, we would like to formulate the task relationships mathematically and make use of them in the MTL algorithm to further improve the ensuing model. In the machine learning community, this is known as multi-task relationship learning (MTRL), and MTRL for simple linear regression tasks had been investigated [54, 110]. A possible way to conduct multi-task relationship learning on DNN is to add a regularization term to the original objective function during DNN training, and the regularizer depends on already known and fixed relationship, or relationship learned on the fly. However, how to define task relationships on the complicated MTL-DNN, and how to do MTRL for complex tasks like speech recognition need further investigation.

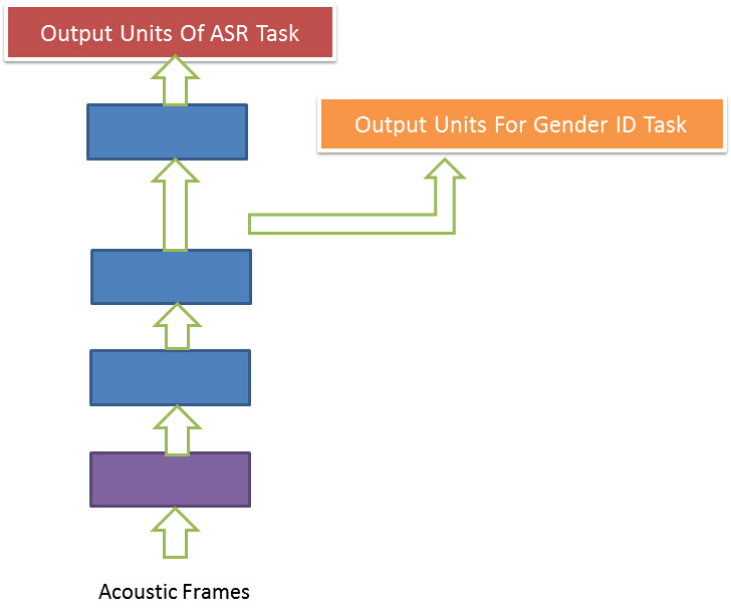

Figure 7.1: An MTL-DNN learning ASR acoustic modeling task and speaker gender identification task together on features of different levels.

(c) **Learning multiple tasks on features at different hidden layers**

Methodologies in this thesis are based on MTL-DNN in which the hidden layers are shared, and multiple task-specific output layers are put on top of the hidden layers. Thus, tasks are learned on the same level of features extracted by the hidden layers.

However, from the view of bionics, this does not match the structure of human brain, nor does it match our common sense in real life. For example, speaker gender identification is a secondary task related to speech recognition, but speaker gender identification is a much simpler task, and it makes use of relatively lower level features compared to speech recognition. Thus MTL-DNN should be more flexible and allow tasks to be learned on features at different hidden layers. Fig. 7.1 displays a possible MTL-DNN architecture in which the ASR acoustic modeling task are learned as usual, but the speaker gender identification task only makes use of lower level of features extracted by the first 3 hidden layers. Therefore, for simpler tasks that do not need high level features as ASR, they can be learned on the features output by hidden layers at the bottom of network.

(d) **Multi-task learning on other deep architectures**

As is mentioned in 2.3.3, research on other deep architectures such as deep con-

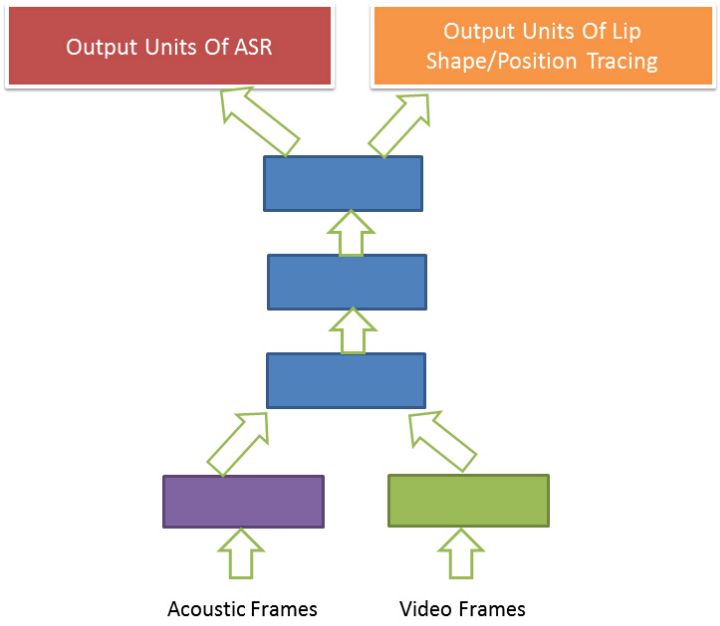

Figure 7.2: An MTL-DNN for audio-visual automatic speech recognition.

volutional neural network (deep-CNN) and deep recurrent neural network (deep-RNN) for ASR is also very active. In fact, recent progress on long-short term memory recurrent neural network (LSTM-RNN) has surpassed DNN and broken its records on TIMIT benchmark [15] and SWitchBoard large vocabulary speech recognition task [111].

Applying the same MTL approach to deep-CNN and deep-RNN is straightforward. [56] learns speech enhancement tasks and other classification tasks together, but learning ASR task with other tasks using MTL-RNN is not explored. The RNN in this paper is also shallow. It would be of sufficient interest to evaluate the effect on MTL on deep-CNN and deep-RNN for ASR.

(e) **MTL for auditory-visual automatic speech recognition**

Selected and investigated secondary tasks in this thesis are all acoustic modeling tasks, but candidates of secondary tasks should not be limited to speech and language tasks. On the contrary, if a *related* secondary task comes from a totally different field, the extra information it brings will be more valuable.

Human speech perception is usually related to other cognitive procedure. In psychology, the McGurk effect is a strong evidence that our brain processes speech with other senses. It is a perceptual phenomenon that demonstrates an interaction

between auditory and visual systems during speech perception. For example, when the syllable /ba-ba/ is spoken with the lip movements of /ga-ga/, the perception of the audience will be /da-da/ surprisingly. In neural science, this can be explained by the structure of the biological neural network [112]. When conducting speech perception, the human brain combines different sources of information to make the best judgment. As a benefit, many people can read out the "speech" from a silent video of lip and articular movement.

Research on the auditory-visual interaction in biology gives hint to improve performance of speech applications with the aid of vision information. [113] and [114] report promising results on speaker verification using lip movement information as inputs to back-end HMM classifier. A more recent paper also works on audio-visual automatic speech recognition (AVASR) using deep learning [115]. MTL-DNN is an ideal structure to merge different sources of information similarly to human brain. We believe MTL-DNN is effective to combine auditory and vision information to improve AVASR performance. Fig. 7.2 shows a possible variant of MTL-DNN for AVASR. The MTL-DNN learns acoustic modeling task together with lip tracing task from both acoustic frames and video frames.

This thesis proposes general guidelines in selecting various secondary tasks to improve phonetic acoustic modeling under the multi-task learning framework. It is not definitely the end of exploration on this issue. Hopefully, it will stimulate more and more MTL applications in ASR. With the support of biological and neural science, we are confident that MTL is a promising approach in general machine learning tasks that are not only limited to automatic speech recognition.

REFERENCES

[1] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, et al., *The HTK book*, vol. 2, Entropic Cambridge Research Laboratory Cambridge, 1997.

[2] Steve J Young and Philip C Woodland, "The use of state tying in continuous speech recognition.," in *Proceedings of the European Conference on Speech Communication and Technology*, 1993.

[3] V Valtchev, JJ Odell, PC Woodland, and SJ Young, "Lattice-based discriminative training for large vocabulary speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1996, vol. 2, pp. 605–608.

[4] P.C. Woodland and D. Povey, "Large scale discriminative training of hidden Markov models for speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 25–47, 2002.

[5] Daniel Povey, "Discriminative training for large vocabulary speech recognition," *Cambridge, UK: Cambridge University*, vol. 79, 2004.

[6] Mark JF Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, no. 2, pp. 75–98, 1998.

[7] Guoli Ye, Brian Mak, and Man-Wai Mak, "Fast gmm computation for speaker verification using scalar quantization and discrete densities.," in *Proceedings of Interspeech*, 2009, pp. 2327–2330.

[8] Edmondo Trentin and Marco Gori, "A survey of hybrid ann/hmm models for automatic speech recognition," *Neurocomputing*, vol. 37, no. 1, pp. 91–126, 2001.

[9] G. Zweig and P. Nguyen, "SCARF: A segmental conditional random field toolkit for speech recognition," in *Proceedings of Interspeech*, 2010.

[10] Mari Ostendorf, Vassilios V Digalakis, Owen Kimball, et al., "From HMM's to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 360–378, 1996.

[11] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[12] Dong Yu and Li Deng, *Automatic Speech Recognition: A Deep Learning Approach*, Springer, 2015.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn, "Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2012, pp. 4277–4280.

[15] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[16] Martin T Hagan, Howard B Demuth, Mark H Beale, et al., *Neural network design*, Pws Pub. Boston, 1996.

[17] Daniel J Felleman and David C Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral cortex*, vol. 1, no. 1, pp. 1–47, 1991.

[18] R. Caruana, *Multitask Learning*, Ph.D. thesis, Carnegie Mellon University, USA, 1997.

[19] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the International Conference on Machine Learning*. ACM, 2008, pp. 160–167.

[20] G. Tur, "Multitask learning for spoken language understanding," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006, pp. 585–588.

[21] Y. Huang, W. Wang, L. Wang, and T. Tan, "Multi-task deep neural network for multi-label learning," in *Proceedings of the IEEE International Conference on Image Processing*, 2013, pp. 2897–2900.

[22] M. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 6965–6968.

[23] P. Cohen, S. Dharanipragada, J. Gros, M. Monkowski, C. Neti, S. Roukos, and T. Ward, "Towards a universal speech recognizer for multiple languages," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, Dec 1997, pp. 591–598.

[24] Slava M Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[25] Reinhard Kneser and Hermann Ney, "Improved backing-off for m-gram language modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1995, vol. 1, pp. 181–184.

[26] Lalit Bahl, Peter Brown, Peter V de Souza, and Robert Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1986, vol. 11, pp. 49–52.

[27] Robert Hecht-Nielsen, "Theory of the backpropagation neural network," in *International Joint Conference on Neural Networks*. IEEE, 1989, pp. 593–605.

[28] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[29] A. Mohamed, G.E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[30] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[31] George E. Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[33] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky, "RNNLM-Recurrent neural network language modeling toolkit," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 2011, pp. 196–201.

[34] K-F Lee, "Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 4, pp. 599–609, 1990.

[35] Steven Greenberg, "Speaking in shorthand–a syllable-centric perspective for understanding pronunciation variation," *Speech Communications*, vol. 29, no. 2, pp. 159–176, 1999.

[36] T. Ko and B. Mak, "Eigentriphones: A basis for context-dependent acoustic modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2011, pp. 4892–4895.

[37] Enrico Bocchieri and Brian Mak, "Subspace distribution clustering hidden Markov model," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 264–275, 2001.

[38] M-Y Hwang and Xuedong Huang, "Shared-distribution hidden Markov models for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 414–420, 1993.

[39] Dongpeng Chen and Brian Mak, "Distinct triphone modeling by reference model weighting," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7150–7154.

[40] Sheri Hunnicutt, Helen M Meng, Stephanie Seneff, and Victor W Zue, "Reversible letter-to-sound sound-to-letter generation based on parsing word morphology," in *Proceedings of the European Conference on Speech Communication and Technology*, 1993, pp. 763–766.

[41] Ernst Günter Schukat-Talamazzini, Heinrich Niemann, Wieland Eckert, Thomas Kuhn, and S Rieck, "Automatic speech recognition without phonemes," in *Proceedings of the European Conference on Speech Communication and Technology*, 1993.

[42] S. Kanthak and H. Ney, "Context-dependent acoustic modeling using graphemes for large vocabulary speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002, vol. 1, pp. 845–848.

[43] P. Charoenpornsawat, S. Hewavitharana, and T. Schultz, "Thai grapheme-based speech recognition," in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. ACL, 2006, pp. 17–20.

[44] S. Stüker, *Acoustic Modeling for Under-Resourced Languages*, Ph.D. thesis, University of Karlsruhe, Germany, 2009.

[45] Tom Ko and Brian Mak, "Eigentrigraphemes for under-resourced languages," *Speech Communications*, vol. 56, pp. 132–141, 2014.

[46] Sinno Jialin Pan and Qiang Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.

[47] W. Byrne, P. Beyerlein, J. M. Huerta, S. Khudanpur, B. Marthi, J. Morgan, N. Peterek, J. Picone, D. Vergyri, and T. Wang, "Towards language independent acoustic modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2000, vol. 2, pp. 1029–1032.

[48] J. Kohler, "Multi-lingual phoneme recognition exploiting acoustic-phonetic similarities of sounds," in *Proceedings of the International Conference on Spoken Language Processing*, 1996.

[49] A. Ghoshal, P. Swietojanski, and S. Renals, "Multilingual training of deep-neural netowrks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7319–7323.

[50] S. Thrun and L. Pratt, *Learning to Learn*, Kluwer Academic Publishers, November 1997.

[51] J.Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.

[52] S. Ben-David and R. Schuller, "Exploiting task relatedness for multiple task learning," in *Conference on Learning Theory*, 2003, pp. 567–580.

[53] Tsuyoshi Kato, Hisashi Kashima, Masashi Sugiyama, and Kiyoshi Asai, "Multi-task learning via conic programming," in *Advances in Neural Information Processing Systems*, 2008, pp. 737–744.

[54] Yu Zhang and Dit-Yan Yeung, "A convex formulation for learning task relationships in multi-task learning," *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, July 2010.

[55] Hongliang Fei and Jun Huan, "Structured feature selection and task relationship inference for multi-task learning," *Knowledge and information systems*, vol. 35, no. 2, pp. 345–364, 2013.

[56] S. Parveen and P. D. Green, "Multitask learning in connectionist ASR using recurrent neural networks," in *Proceedings of the European Conference on Speech Communication and Technology*, 2003, pp. 1813–1816.

[57] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7304–7308.

[58] G. Heigold, V. Vanhoucke, A Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2013, pp. 8619–8623.

[59] Peter T Daniels and William Bright, *The world's writing systems*, Oxford University Press, 1996.

[60] Stephan Kanthak and Hermann Ney, "Multilingual acoustic modeling using graphemes," in *Proceedings of Interspeech*, 2003.

[61] Mirjam Killer, Sebastian Stüker, and Tanja Schultz, "Grapheme based speech recognition.," in *Proceedings of Interspeech*, 2003.

[62] J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, 1997, pp. 347–354.

[63] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2000, vol. 3, pp. 1635–1638.

[64] Janez Kaiser, Bogomir Horvat, and Zdravko Kacic, "A novel loss function for the overall risk criterion based discriminative training of HMM models," in *Proceedings of the International Conference on Spoken Language Processing*, 2000.

[65] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah, "Boosted MMI for model and feature-space discriminative training," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2008, pp. 4057–4060.

[66] Abdel-rahman Mohamed, Dong Yu, and Li Deng, "Investigation of full-sequence training of deep belief networks for speech recognition.," in *Proceedings of Interspeech*, 2010, pp. 2846–2849.

[67] Karel Veselỳ, Arnab Ghoshal, Lukávs Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of Interspeech*, 2013, pp. 2345–2349.

[68] Hang Su, Gang Li, Dong Yu, and Frank Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription.," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 6664–6668.

[69] Brian Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2009, pp. 3761–3764.

[70] Dongpeng Chen, Brian Mak, Cheung-Chi Leung, and Sunil Sivadas, "Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 5592–5596.

[71] Victor Zue, Stephanie Seneff, and James Glass, "Speech database development at mit: Timit and beyond," *Speech Communications*, vol. 9, no. 4, pp. 351–356, 1990.

[72] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N

Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[73] Li Deng and John C Platt, "Ensemble deep learning for speech recognition," in *Proceedings of Interspeech*, 2014.

[74] Douglas B Paul and Janet M Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[75] A. Stolcke et al., "SRILM–an extensible language modeling toolkit.," in *Proceedings of Interspeech*, 2002, pp. 901–904.

[76] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáv**s** Burget, Ond**v**rej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlív**c**ek, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," 2011.

[77] Jaco Badenhorst, Charl van Heerden, Marelie Davel, and Etienne Barnard, "Collecting and evaluating speech recognition corpora for 11 south african languages," *Language resources and evaluation*, vol. 45, no. 3, pp. 289–309, 2011.

[78] M.Davel M. Tempest, "Dictionarymaker 2.16 user manual," `http://dictionarymaker.sourceforge.net/`, 2009.

[79] Ji Ming, Peter O'Boyle, Marie Owens, and F Jack Smith, "A bayesian approach for building triphone models for continuous speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 6, pp. 678–684, 1999.

[80] Kai-Fu Lee, *Automatic Speech Recognition: The Development of the Sphinx Recognition System*, vol. 62, Springer, 1989.

[81] Andrej Ljolje, "High accuracy phone recognition using context clustering and quasi-triphonic models," *Computer Speech and Language*, vol. 8, no. 2, pp. 129–151, 1994.

[82] Hung-An Chang and James R Glass, "A back-off discriminative acoustic model for automatic speech recognition.," in *Proceedings of Interspeech*, 2009, pp. 232–235.

[83] Satoshi Takahashi and Shigeki Sagayama, "Four-level tied-structure for efficient representation of acoustic modeling," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1995, vol. 1, pp. 520–523.

[84] XD Huang and MA Jack, "Semi-continuous hidden markov models for speech signals," *Computer Speech & Language*, vol. 3, no. 3, pp. 239–251, 1989.

[85] Daniel Povey, Lukas Burget, Mohit Agarwal, Pinar Akyazi, Kai Feng, Arnab Ghoshal, Ondrej Glembek, Nagendra K Goel, Martin Karafiát, Ariya Rastrow, et al., "Subspace Gaussian mixture models for speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2010, pp. 4330–4333.

[86] George Saon and Jen-Tzung Chien, "Bayesian sensing hidden Markov models," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 43–54, 2012.

[87] Tom Ko and Brian Mak, "Eigentriphones for context-dependent acoustic modeling.," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 6, pp. 1285–1294, 2013.

[88] Matthew Turk, Alex P Pentland, et al., "Face recognition using eigenfaces," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1991, pp. 586–591.

[89] Roland Kuhn, Jean-Claude Junqua, Patrick Nguyen, and Nancy Niedzielski, "Rapid speaker adaptation in eigenvoice space," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 6, pp. 695–707, 2000.

[90] Brian Mak, Tsz-Chung Lai, and Roger Hsiao, "Improving reference speaker weighting adaptation by the use of maximum-likelihood reference speakers," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2006, vol. 1, pp. I–I.

[91] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proceedings of the IEEE In-*

ternational Conference on Acoustics, Speech, and Signal Processing, 1992, pp. 517–520.

[92] A. Stolcke, "SRILM - An extensible language modeling toolkit," in *Proceedings of the International Conference on Spoken Language Processing*, 2002, pp. 901–904.

[93] R. Haeb-Umbach and H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992, pp. 13–16.

[94] M.J.F. Gales, "Semi-tied covariance matrices," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1998, pp. 657–660.

[95] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Proceedings of the International Conference on Spoken Language Processing*, 1996, pp. 1137–1140.

[96] Hui Lin, Li Deng, Dong Yu, Yi-Fan Gong, Alex Acero, and Chin-Hui Lee, "A study on multilingual acoustic modeling for large vocabulary ASR," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 2009, pp. 4333–4336.

[97] L. Burget, P. Schwarz, M. Agarwal, P. Akyazi, Kai Feng, A Ghoshal, O. Glembek, N. Goel, M. Karafiat, D. Povey, A Rastrow, R.C. Rose, and S. Thomas, "Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 2010, pp. 4334–4337.

[98] Frantivsek Grézl, Martin Karafiat, and Milovs Janda, "Study of probabilistic and bottle-neck features in multilingual environment," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 359–364.

[99] Karel Vesely, Martin Karafiát, Frantisek Grezl, Marcel Janda, and Ekaterina Egorova, "The language-independent bottleneck features," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 336–341.

[100] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals, "Unsupervised cross-lingual knowledge transfer in DNN-based LVCSR," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, 2012, pp. 246–251.

[101] International Phonetic Association, *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*, Cambridge University Press, 1999.

[102] "Lwazi phone set," `ftp://hlt.mirror.ac.za/Phoneset/Lwazi.Phoneset.1.2.pdf`, 2009.

[103] Theodoros Evgeniou and Massimiliano Pontil, "Regularized multi-task learning," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004, pp. 109–117, ACM.

[104] Yajie Miao and Florian Metze, "Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training.," in *Proceedings of Interspeech*, 2013, pp. 2237–2241.

[105] Dong Yu, Frank Seide, Gang Li, and Li Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4409–4412.

[106] Yichuan Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[107] Renqiang Min, David A Stanley, Zineng Yuan, Anthony Bonner, and Zhaolei Zhang, "A deep non-linear feature mapping for large-margin knn classification," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 357–366.

[108] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil, "Convex multi-task feature learning," *Machine Learning*, vol. 73, no. 3, pp. 243–272, 2008.

[109] Andreas Argyriou, Massimiliano Pontil, Yiming Ying, and Charles A Micchelli, "A spectral regularization framework for multi-task structure learning," in *Advances in Neural Information Processing Systems*, 2007, pp. 25–32.

[110] Wenliang Zhong and James Kwok, "Convex multitask learning with flexible task clusters," in *Proceedings of the International Conference on Machine Learning*, 2012.

[111] George Saon, Hagen Soltau, Ahmad Emami, and Michael Picheny, "Unfolded recurrent neural networks for speech recognition," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[112] Audrey R Nath and Michael S Beauchamp, "A neural basis for inter-individual differences in the McGurk effect, a multisensory speech illusion," *Neuroimage*, vol. 59, no. 1, pp. 781–787, 2012.

[113] H Ertan Cetingul, Yücel Yemez, Engin Erzin, and A Murat Tekalp, "Discriminative analysis of lip motion features for speaker identification and speechreading," *Image Processing, IEEE Transactions on*, vol. 15, no. 10, pp. 2879–2891, 2006.

[114] Juergen Luettin, Neil Thacker, Steve W Beet, et al., "Speaker identification by lipreading," in *Proceedings of the International Conference on Spoken Language Processing*. IEEE, 1996, vol. 1, pp. 62–65.

[115] Kuniaki Noda, Yuki Yamaguchi, Kazuhiro Nakadai, Hiroshi G Okuno, and Tetsuya Ogata, "Audio-visual speech recognition using deep learning," *Applied Intelligence*, vol. 42, no. 4, pp. 722–737, 2015.