Incorporating Context Information into Deep Neural Network Acoustic Models

Yajie Miao

July 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Florian Metze, Chair (Carnegie Mellon University)
Alan W Black (Carnegie Mellon University)
Alex Waibel (Carnegie Mellon University)
Jinyu Li (Microsoft)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Acoustic Models, Deep Neural Networks, Context Information

Abstract

The introduction of deep neural networks (DNNs) has advanced the performance of automatic speech recognition (ASR) tremendously. On a wide range of ASR tasks, DNN models show superior performance than the traditional Gaussian mixture models (GMMs). Although making significant advances, DNN models still suffer from data scarcity, speaker mismatch and environment variability. This thesis resolves these challenges by fully exploiting DNNs' ability of integrating heterogeneous features under the same optimization objective. We propose to improve DNN models under these challenging conditions by incorporating context information into DNN training.

On a new language, the amount of training data may become highly limited. This data scarcity causes degradation on the recognition accuracy of DNN models. A solution is to transfer knowledge from other languages to the low-resource condition. This thesis proposes a framework to build cross-language DNNs via language-universal feature extractors (LUFEs). Convolutional neural networks (CNNs) and deep maxout networks (DMNs) are employed to improve the quality of LUFEs, which enables the generation of invariant and sparse feature representations. This framework notably improves the recognition accuracy on a wide range of low-resource languages.

The performance of DNNs degrades when the mismatch between acoustic models and testing speakers exists. A form of context information which encapsulates speaker characteristics is i-vectors. This thesis proposes a novel framework to perform feature-space speaker adaptive training (SAT) for DNN models. A key component of this approach is an adaptation network which takes i-vectors as inputs and projects DNN inputs into a normalized feature space. The DNN model fine-tuned in this new feature space rules out non-speech variability and becomes more independent of specific speakers. This SAT method is applicable to different feature types and model architectures.

The proposed adaptive training framework is further extended to incorporate distance- and video-related context information. The distance descriptors are extracted from deep learning models which are trained to distinguish distance types on the frame level. Distance adaptive training (DAT) using these descriptors captures speaker-microphone distance dynamically on the frame level. When performing ASR on video data, we naturally have access to both the speech and the video modality. Video- and segment-level visual features are extracted from the video stream. Video adaptive training (VAT) with these visual features results in more robust acoustic models that are agnostic to environment variability. Moreover, the proposed VAT approach removes the need for frame-level visual features and thus achieves audio-visual ASR on truly open-domain videos.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, automatic speech recognition (ASR) systems have seen evident improvement on their performance and rapid expansion on their applicability. A major driving force for this advancement is the introduction of deep neural networks (DNNs) as acoustic models. Compared to the traditional Gaussian mixture models (GMMs), the advantage of DNN models has been confirmed on a wide variety of ASR tasks [13, 38, 101]. Applications of DNNs generally fall into two categories. In *hybrid systems*, DNNs are trained to classify tied context-dependent (CD) states and estimate their posterior probabilities. In *tandem systems*, we use DNNs to generate phone posteriors or bottleneck features (BNF), and build normal GMM models with the discriminative front-end [24, 25]. In addition to DNNs, other deep structures, such as convolutional neural networks (CNNs) [3, 4, 95, 96, 107, 117] and recurrent neural networks (RNNs) [30, 62, 97, 98], have also been exploited as acoustic models. More details about these architectures will be presented in Chapter 3.

1.1 Current Challenges for DNN Acoustic Models

Although making significant advances, the performance of DNN acoustic models still suffers from challenges such as noise, channel mismatch, speaker mismatch [43]. This thesis focuses on alleviating the effects of the following three challenges.

- **Data Scarcity.** In the hybrid DNN-HMM approach, the DNN models normally consist of a series of hidden layers. Therefore, DNN models tend to have much more parameters than GMM models. For example, in [124], the hybrid system with a 5-hidden-layer fully-connected DNN has 12 times more parameters than its corresponding GMM model. When

the amount of transcribed speech becomes limited (e.g., less than 10 hours), the large parameter space of DNNs can cause overfitting easily during DNN training. This may degrade the recognition performance of DNN models greatly on unseen testing data.

- **Speaker Mismatch.** Another long-standing issue for ASR is the mismatch between the acoustic models and testing speakers. A degradation of recognition accuracy is typically observed when porting a recognizer to a testing set where the speakers have not been included in the training set. An effective step to mitigate this mismatch is to perform speaker adaptation during decoding. There are two types of speaker adaptation. Model-space adaptation modifies the speaker-independent (SI) model towards particular testing speakers, whereas feature-space adaptation transforms the features of testing speakers towards the acoustic model.
- **Environment Variability.** Real-world applications require ASR systems to handle various types of environment variability such as noise and reverberation. In recent years, DNN models have dramatically advanced the recognition accuracy on clean, close-talking speech. However, robustness still remains to be a challenge for DNNs. It is revealed in [43] that as with GMMs, the performance of DNNs drops significantly as the SNR decreases. One example of environment variability is on amateur videos (e.g., YouTube videos) where the distance between the speakers and the microphones varies frequently. Also, the scenes (e.g., car, office, street) of the conversations may differ a lot among videos. Because of these factors, previous work [44, 59] has reported the state-of-the-art WER of around 40% on transcribing YouTube videos.

1.2 Thesis Statement

In comparison to the conventional GMMs acoustic models, DNN models can take input features of large dimensions. For example, the inputs of DNNs are normally concatenation of neighbouring frames whose dimension can go easily up to hundreds. This nice property enables DNNs to combine information from different sources.

This thesis conducts a systematic study to investigate the incorporation of heterogeneous context information into DNNs acoustic models. We aim to answer two questions. First, what kind of additional information can we leverage to resolve the aforementioned challenges? Second, how can we incorporate the additional information into DNN models in an effective manner? Our research can be described more specifically by the following aspects.

1.2.1 Language Context: Cross-Language DNNs with Language-Universal Feature Extractors

DNN models generally have more parameters than GMMs. The performance of DNNs typically degrades when we have limited training data. From a transfer learning perspective, DNN models under low-resource conditions can benefit from sharing knowledge among languages. Previous work [36, 41, 68] has studied multilingual DNNs to realize knowledge transfer across languages. The application of multilingual DNNs for cross-language acoustic modeling is also briefly investigated in [41]. The contribution of this thesis is to propose a more flexible framework for cross-language DNN acoustic modeling. More importantly, we further extend this framework from different aspects. These extensions are orthogonal to choices of acoustic modeling methods. Therefore, they are also applicable to previous proposals such as [41].

- We establish our framework to build cross-language DNN models via language-universal feature extractors (LUFEs). A LUFE consists of the shared hidden layers of the multilingual DNN. Given a new language, DNN models are built over the outputs from the LUFE. Our approach differs from [41] in that on the new language, we are learning a complete DNN model instead of a single softmax layer. This gives us greater modeling flexibility, as well as better recognition results, on the new language.
- The quality of LUFE is improved by two techniques. First, we propose to train LUFEs with CNNs. Due to local filters and max-pooling layers, CNNs normalize spectral variation in the speech signal more effectively than DNNs. Thus, CNN-based LUFEs give us more invariant feature representations. Second, we introduce sparsity into the LUFE feature representations by taking advantage of the deep maxout networks (DMNs) [28]. Our previous work [72] makes the first attempt to apply maxout networks to ASR. In a DMN, units at each hidden layer are divided into groups, and each group generates a single output with max-pooling. With a *non-maximum masking* operation, feature representations with truly-zero sparsity can be generated from the maxout layer.
- We propose the DistModel strategy to accelerate training of LUFEs over multiple GPUs. Learning LUFEs can be highly expensive because training data have to contain multiple languages. In DistModel, the multilingual data are split into equally-sized partitions. A complete LUFE is trained on a GPU and using one of the data partitions. After a particular number of mini-batches, the different LUFE model instances are averaged to form the starting model for the subsequent training. After configuration optimization, this time-

synchronous method results in over 2 times speed-up on LUFE training and negligible WER loss on the new language.

1.2.2 Speaker Context: Speaker Adaptive Training of DNN Models using I-vectors

For GMM models, an effective procedure to alleviate the effect of speaker mismatch is speaker adaptation [23, 53]. Model-space adaptation modifies the SI model towards particular testing speakers, while feature-space adaptation transforms the features of testing speakers towards the SI model. Another technique closely related with speaker adaptation is speaker adaptive training (SAT) [5, 6]. When carried out in the feature space, SAT performs adaptation on the training set and projects training data into a speaker-normalized space. Parameters of the acoustic models are estimated in this new feature space. Acoustic models trained this way become independent of specific training speakers and thus generalize better to unseen testing speakers. In practice, SAT models generally give better recognition results than SI models, when speaker adaptation is applied to both of them.

For DNN models, a large amount of previous work has been dedicated to speaker adaptation. For example, in [55, 121], SI-DNN models are augmented with additional speaker-specific layers which are trained on the adaptation data. Also, [92, 101] achieve adaptation of DNNs by training DNNs using speaker-adaptive features, and [58] adapts the entire SI-DNNs to testing speakers with DNN fine-tuning. In comparison to speaker adaptation, past work has made fewer attempts on SAT of DNNs. Training DNNs with SA features [92, 101, 114] or additional speaker-specific information [33, 99, 103] can be treated as a form of SAT. In [120], Xue *et al.* append speaker codes [1, 2] to the hidden and output layers of the DNN model. SAT is achieved by jointly learning the speaker-specific speaker codes and the SI-DNN. In [84], speaker availability is normalized by allocating certain layers of the DNN as the SD layers that are learned on a speaker-specific basis. Over different speakers, the other layers are adaptively trained by picking the SD layer corresponding to the current speaker. Although showing promising results, the application of these proposals is constrained to specific feature types or model structures. For example, the approach in [120] is not applicable to CNNs because it is infeasible to append speaker codes to the hidden convolution layers. Also, in these methods, adaptation of the resulting SAT models generally needs multiple decoding passes, which undermines the decoding efficiency.

In this thesis, we propose a novel framework to carry out feature-space SAT for DNNs.

Building of SAT-DNN models starts from an initial SI-DNN model that has been trained over the entire training set. Then, our framework uses i-vectors extracted at the speaker level as a compact representation of each speaker's acoustic characteristics. An adaptation neural network is learned to convert i-vectors to speaker-specific linear feature shifts. Adding the shifts to the original DNN input vectors (e.g., MFCCs) produces a speaker-normalized feature space. The parameters of the SI-DNN are updated in this new feature space, which finally gives us the SAT-DNN model. This thesis explores the optimal configuration of SAT-DNNs for large vocabulary continuous speech recognition (LVCSR) tasks. Apart from hybrid models, we demonstrate the extension of our SAT framework to CNNs and BNF generation. Furthermore, we study the combination of SAT and speaker adaptation for DNNs. During decoding, model-space adaptation is applied atop of SAT-DNN models for further improved recognition results.

1.2.3 Distance Context: Robust Speech Recognition with Distance-Aware DNNs

Environment variability such as noise and reverberation poses special challenges for ASR systems. A critical type of variability comes from the distance between speakers and microphones. A performance degradation is typically observed when we port DNN models from close to distant speech [112]. A number of techniques have been developed to enhance the robustness of DNN models on far-field speech. For instance, in [60, 112], DNN models are improved by combining speech signals from multiple distant microphones via concatenation or beamforming [63]. Also for DNN models, [122] evaluates the existing environmental robustness methods on a distant-microphone meeting transcription task. A robust front-end is derived by integrating different enhancement approaches and feature types. Although showing promising results, these techniques have the limitation that they only deal with constantly distant speech. That is, the speaker-microphone distance (SMD) is assumed to be unchanged throughout the course of recording. To our knowledge, no previous work has dealt with dynamic distance information.

In many real-world scenarios, the SMD can be quite dynamic. For example, a speaker is likely to walk around when talking to a far-field microphone located at a fixed position. This thesis proposes to construct distance-aware DNNs by explicitly incorporating the SMD information into DNN training. The SMD descriptors are extracted dynamically on the frame level using a SMD extractor. The extractor is a deep architecture that is trained on an external meeting corpus. The adaptive training approach, which is originally proposed for SAT of DNNs, is ported to incorporate the SMD information into DNNs models. Applying this approach enables

us to perform distance adaptive training (DAT) for DNNs. Extensive investigations are additionally conducted to optimize the DAT-DNN models from various aspects, e.g., the architectures serving as the SMD extractor.

1.2.4 Video Context: Open-Domain Audio-Visual Speech Recognition using Deep Learning

When performed on video data, speech recognition naturally has access to two modality: audio and video. This motivates researchers to take advantage of visual features to enhance ASR, especially under noisy conditions [10, 21, 31, 49]. However, these previous proposals have limitations in that they generally adopt visual features which are extracted from the speaker's mouth region, including lip contours and mouth shapes. Although available in highly constrained videos, these features cannot always be obtained from open-domain videos. For example, in a large portion of the YouTube videos, the speakers do not appear in the video frames at all. Another limitation of traditional audio-visual ASR is the requirement for alignments between the speech and video frames. Since the speech and video streams have different sampling rates, aligning them may introduce inaccurate visual features into acoustic modeling.

The contribution of this thesis is to relax these constraints and extend audio-visual ASR to open-domain videos. Instead of frame-level, highly restrictive mouth/lip features, we extract video- or segment-level visual features. Extraction of these visual features is achieved with computer vision models trained on external datasets. In particular, we take advantage of deep CNN architectures to obtain informative visual features from the raw pixels of open-domain videos. The adaptive training approach continues to be extended to incorporate visual features in a more flexible way. Together with the speaker and distance context, we prove the generality of this adaptive training framework in fusing different types of additional information.

1.3 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews fundamentals of ASR and Chapter 3 reviews deep learning approaches/architectures for acoustic modeling. Chapters 4, 5, 6 and 7 correspond to the four statements in Section 1.2 respectively.

- Chapter 2 reviews basic principles and components for ASR.
- Chapter 3 reviews acoustic modeling using deep learning models. In addition to hybrid

models, we also review how DNNs are used for BNF extraction. Other deep learning architectures such as CNNs and RNNs are also explained.

- Chapter 4 presents our work on cross-language DNNs with language-universal feature extractors.
- Chapter 5 presents our work on SAT of DNN models using i-vectors. In this chapter, we perform feature-space SAT of DNNs by taking advantage of i-vectors as the speaker representations.
- Chapter 6 improves the robustness of DNN models by incorporating speaker-microphone distance information dynamically on the frame level.
- Chapter 7 describes the incorporation of visual features when our task is to transcribe video data. We investigate what visual features are useful for ASR and how we can fuse visual features effectively into acoustic models.
- Chapter 8 summarizes this thesis.

Chapter 2

Fundamentals of Automatic Speech Recognition

Given a human speech segment, the goal of automatic speech recognition (ASR) is to transcribe the speech into texts, i.e., perform a speech-to-text task. In this chapter, we first present the fundamentals on the ASR framework and key components in the framework. Then we describe the GMM-HMM paradigm for acoustic modeling.

2.1 ASR Framework

2.1.1 Basic Principle

Suppose that the components required by speech recognition have been trained or constructed. Figure 2.1 depicts the general framework for ASR. During the recognition stage, we first extract speech features from the given speech segments (i.e., utterances). Using specific search algorithms, the recognizer incorporates various components (e.g., acoustic models, language models, dictionaries) and finds the best word hypothesis.

More formally, we denote the given speech segment as O. Based on the Bayes decision theory, speech recognition is to find the word sequence W that maximizes $Pr(W|O)$, i.e., the posterior probability of the word sequence given the observations. According to the Bayes rule, we can further derive this posterior as follows:

$$Pr(W|O) = \frac{Pr(O|W)Pr(W)}{Pr(O)} \propto Pr(O|W)Pr(W) \tag{2.1}$$

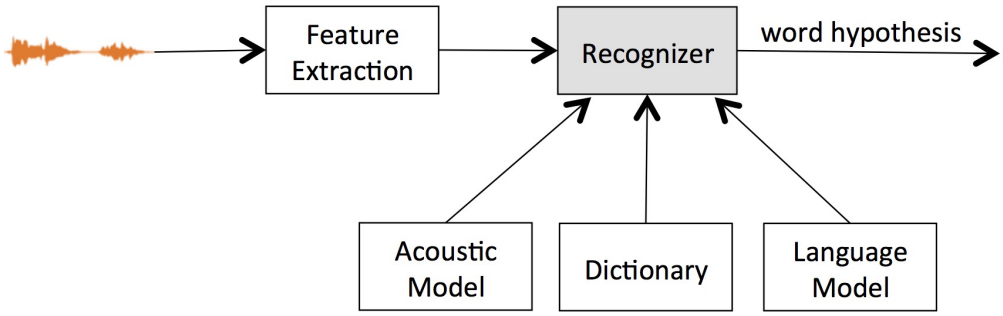


Figure 2.1: The basic framework of ASR.

Given the speech segment, the prior probability $Pr(O)$ is constant with respect to W and thus can be eliminated from Equation 2.1. Then the task of ASR can be further described as

$$\hat{W} = argmax_W Pr(W|O) = argmax_W Pr(O|W)Pr(W) \tag{2.2}$$

On the right-hand side of this equation, $Pr(O|W)$ represents the likelihood of the observations O given the word hypothesis W. This score is normally calculated from an acoustic model. $Pr(W)$ represents the prior probability of the word sequence W which is computed from a language model.

2.1.2 Acoustic Model

An acoustic model evaluates how likely a speech segment O is generated from a possible word hypothesis W. It provides a statistical perspective into speech production with respect to word transcripts. In the traditional Gaussian mixture models-hidden Markov model (GMM-HMM) paradigm for acoustic modeling, the sequential property of speech is modeled by the HMM and the likelihood of speech generation is evaluated by the GMM. In the state-of-the-art deep learning-based acoustic modeling, deep learning models are adopted as acoustic models. More details regarding the GMM-HMM and deep learning paradigms will be presented in Section 2.2 and 3. Acoustic models play a central role in ASR systems. This thesis focuses on how to improve the performance of acoustic models.

2.1.3 Dictionary

During acoustic modeling, the training data usually consist of the speech segments (utterances) and their corresponding word sequences (transcripts). For LVCSR systems, modeling the map-

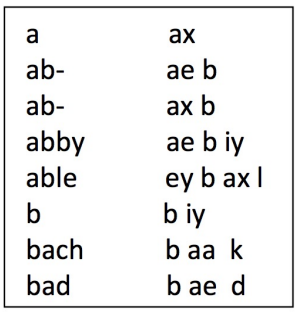


Figure 2.2: A snippet of a dictionary. Each line contains a word (e.g., "able") which is followed by a sequence of phones (e.g., "ey b ax l") corresponding to the word. A single word is likely to have multiple different pronunciations, e.g., the two "ab-" in the figure.

ping from speech to words directly is not realistic. It is necessary to have a dictionary to convert written words into their pronunciations in the form of phones. Phones are the smallest units to comprise the pronunciation of a spoken word. Figure 2.2 shows a snippet of an English dictionary. We can see that the mapping from phones to letters is not necessarily one-to-one. For English, this mapping can be many-to-one or one-to-many. On some other languages such as Spanish, this mapping displays more regularity. When used in ASR systems, dictionaries, also known as lexicons, encode how a sequence of phone labels are mapped to a word.

2.1.4 Language Model

A language model is used to estimate the prior probability of generating any sentence. Suppose the sentence W contains k words, i.e., $W = W_1, W_2, ..., W_k$. Language modeling is to assign a probability to this sentence $Pr(W_1, W_2, ..., W_n)$. Language modeling is a common problem shared by a variety of language processing and understanding areas, such as speech recognition, natural language processing, information retrieval, parsing, etc. The probability $Pr(W)$ can be decomposed as [42]:

$$\begin{aligned} Pr(W) &= Pr(W_1, W_2, ..., W_n) \\ &= Pr(W_1)Pr(W_2|W_1)Pr(W_3|W_1, W_2)...Pr(W_n|W_1, W_2, ..., W_{n-1}) \\ &= \prod_{i=1}^{k} Pr(W_i|W_1, W_2, ..., W_{i-1}) \end{aligned} \tag{2.3}$$

where $Pr(W_i|W_1, W_2, ..., W_{i-1})$ is the probability of the word W_i appearing following the word history $W_1, W_2, ..., W_{i-1}$. This probability can be estimated from a text corpus by counting

the co-occurrences of $W_1, W_2, ..., W_{i-1}$ and W_i. In practice, with a large- or moderate-sized vocabulary, this co-occurrences can be none or relative rare, resulting in unreliable estimate of this probability. To tackle this issue, we restrict the word history to at most n words, and thus construct n-gram language models. In this case, the current word has dependency only on the past $n-1$ words, i.e., $Pr(W_i|W_1, W_2, ..., W_{i-1}) = Pr(W_i|W_{i-n+1}, ..., W_{i-1})$. The probability $Pr(W)$ can now be computed as

$$Pr(W) = \prod_{i=1}^{k} Pr(W_i|W_{i-n+1}, ..., W_{i-1}) \tag{2.4}$$

Depending on different values we adopt for n, we can have unigram ($Pr(W_i)$) where the current word has no dependency on the word history, bigram ($Pr(W_i|W_{i-1})$) where the current word depends only on the previous word, trigram ($Pr(W_i|W_{i-2}, W_{i-1})$) where the current word depends on the previous two words, and even higher order. Building of n-gram language models can be achieved by taking a text corpus as training data. For example, from a training text corpus, we observe the count of the word pair $C(W_{i-1}, W_i)$, and the count of the singleton word $C(W_i)$. Then the probability $Pr(W_i|W_{i-1})$ can be estimated as:

$$Pr(W_i|W_{i-1}) = \frac{C(W_{i-1}, W_i)}{C(W_i)} \tag{2.5}$$

Estimating n-gram language models this way suffers from the data sparsity problem. A large portion of the co-occurring word sequences $W_{i-n+1}, ..., W_{i-1}, W_i$ may have low counts or frequencies in the training corpus, leaving the probability computation unreliable. This data sparsity problem becomes more severe when the order of n-gram becomes higher. To alleviate this issue, a large number of smoothing methods have been proposed to smooth the probabilities. A potential solution is backoff smoothing, where we turn to a lower-order n-gram distribution if the higher-order n-gram has no occurrences in the training corpus. Commonly-used backoff smoothing approaches include Good-Turing smoothing, Katz smoothing, Kneser-Ney smoothing, etc.

In LVCSR systems, we usually store a backoff-smoothed language model into the ARPA format. Figure 2.3 shows a snippet of the ARPA-formatted bigram language model. The n-grams are grouped according to their length, and each group starts with the keyword \n-gram. Each line starts with the logarithm (base 10) of the probability of the n-gram, which is followed by the sequence of words in the n-gram. At the end of the line, we optionally have the logarithm (base 10) of the backoff probability. A number of toolkits exist in the community which enable

```
\1-grams:
-2.23389    a           -0.596971
-4.24122    ab-         -0.27366
-5.11687    abby        -0.241424
-3.91358    able        -0.557586
-4.90084    bach        -0.299549

\2-grams:
-2.99082    a baby      -0.180048
-3.45831    a bag       -0.0845801
-0.51861    abby and    -0.142565
-0.243617   able to     -0.451501
-1.61405    bach you
```

Figure 2.3: A snippet of a bigram language model in the ARPA format. Each line starts with the logarithm (base 10) of the probability of the n-gram, which is followed by the sequence of words and optionally the logarithm (base 10) of the backoff probability.

us to build ARPA-formatted language models easily. Examples of widely-used toolkits include the SRI Language Modeling Toolkit (SRILM) [109] and the IRST Language Modeling Toolkit (IRSTLM).

2.1.5 Decoding

After we train the acoustic model and language model, we can fuse these components in the decoding (i.e., testing) stage. Given a testing speech segment, the goal of decoding is to find a sequence of words whose acoustic and language model scores match the speech signal best. As a result, the decoding process can also be viewed as a search process where the language model defines the initial search space. ASR systems generally employ two search algorithms, i.e., the Viterbi search and A* search.

For the experiments in this thesis, we apply the Viterbi search whose implementation is based on weighted finite-state transducers (WFSTs). A WFST is a finite-state acceptor (FSA) in which each transition has an input symbol, an output symbol and a weight. A path through the WFST takes a sequence of input symbols and emits a sequence of output symbols with the corresponding weight. WFSTs have been used widely for ASR decoding [80, 89]. In modern ASR systems, decoding generally requires four individual components: the language model, the lexicon, the context dependency, and the HMM structure. The idea of WFST-based decoding is to compose these components into individual WFSTs, and then compose these WFSTs into a comprehensive

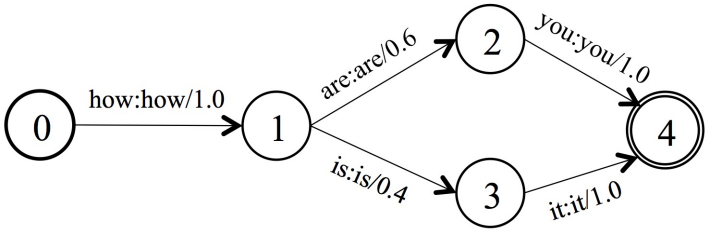


Figure 2.4: A toy example of the grammar (language model) WFST. The arc weights are the probability of emitting the next word when given the previous word. The node 0 is the start node, and the double-circled node is the end node.

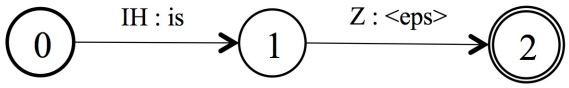


Figure 2.5: The WFST for the phone-lexicon entry "is IH Z". The "<eps>" symbol means no inputs are consumed or no outputs are emitted.

search graph. We represent the four WFSTs corresponding to the four components as G, L, C and H respectively. Then composition of the search space can be formally denoted as:

$$S = min(det(H \circ det(C \circ det(L \circ G)))) \tag{2.6}$$

where $\circ$, det and min denote composition, determinization and minimization respectively. The search graph S encodes the mapping from a sequence of HMM states emitted on speech frames to a sequence of words.

To exemplify WFSTs construction, we dive into the details on the language model and lexicon WFSTs. Here we take English as the example, although the same procedures hold for other languages. The language model WFST, also known as grammar WFST, encodes the permissible word sequences in a language/domain. The WFST shown in Figure 2.4 represents a toy language model which permits two sentences "how are you" and "how is it". The WFST symbols are the words, and the arc weights are the language model probabilities. With this WFST representation, ASR decoding in principle can leverage any language models that can be converted into WFSTs. This conversion can be readily achieved for the ARPA-formatted n-gram language models. A lexicon WFST encodes the mapping from sequences of phones to words. Figure 2.5 shows an example for a lexicon which contains a single word "is". When containing multiple words, the lexicon WFST can also assign different weights to different words, or different pronunciations for the same word.

2.1.6 Speech Front-end

So far we have represented the speech segment as a variable O. In practice, the ASR system takes audio waveform files as the inputs. A waveform simply contains a sequence of values which are sampled from the microphone input at a sampling rate (e.g., 8kHz, 16kHz, etc.). We process the waveform into a sequence of feature vectors before feeding it into ASR. Specifically, a window is applied to the samples which normally has a length of 25 milliseconds. On each window of samples, we extract a feature vector using signal processing methods. Each time the window is moved by a step size of 10 milliseconds, leaving an overlap of 15 milliseconds between every pair of neighbouring frames. From each of these speech frames, a feature vector can be extracted using standard signal processing techniques. Commonly-used speech features include mel frequency Cepstral coefficients (MFCCs) and perceptual linear prediction (PLP).

After obtaining the raw acoustic features, various processing techniques can be adopted to enhance the feature quality towards ASR tasks. These techniques can be roughly categorized as follows.

- Adding deltas and double-deltas. Deltas are derived as the difference between the raw features of neighboring frames, and double-deltas are the difference between the deltas of neighboring frames. These two sets of additional coefficients encode the temporal evolution of speech signals, and thus can be useful for ASR. The deltas and double-deltas are normally appended to the original features.
- Mean and variance normalization. To reduce some variability, we normalize the data so that the resulting feature vectors have zero mean and unit variance on each dimension. This normalization can be applied to different levels, e.g., the speaker level or the utterance level.
- Vocal tract length normalization (VTLN). The frequency bins are warped with factors differing across speakers. With different acoustic characteristics, speakers have different VTLN factors. VTLN helps to reduce speaker variability [128].
- Applying linear transforms. In ASR, a critical method for feature normalization is to apply linear transforms to feature vectors. Commonly-used transforms include linear discriminant analysis (LDA) [34] which is intended for class separation, and feature-space maximum likelihood linear regression (fMLLR) [23] that is meant for speaker adaptation.

2.1.7 Evaluation of ASR

The quality of the ASR outputs can be evaluated by the word error rate (WER) metric. Given a pair of reference and hypothesis word sequences, WER can be computed as:

$$word\_error\_rate = \frac{edit\_distance}{\#reference\_words} \tag{2.7}$$

where $edit\_distance$ is the edit distance (Levenshtein distance) of the hypothesis against the reference, and $\#reference\_words$ is the number of words in the reference. Edit distance means the minimum number of substitutions, insertions and deletions required to convert the hypothesis into the reference. A WER of 10% means that the ASR system makes 10 edit-distance errors with respect to 100 reference words.

2.2 The GMM-HMM Paradigm

A critical component of a model ASR system is the hidden Markov model (HMM). The HMM is a powerful method to characterize the generation of sequential data (e.g., speech signals) from hidden states. The data samples in the sequences can be discretely or continuously distributed. Due to its generality, the HMM has been successfully used in a wide variety of speech and language processing tasks, including speech recognition, speech synthesis, speech enhancement, language modeling, machine translation, part-of-speech tagging, spoken language understanding, and machine translation.

2.2.1 Hidden Markov Models

The hidden states in the HMM form a Markov chain. In the context of speech recognition, the HMM describes the stochastic process of generating a sequence of acoustic feature vectors from the hidden states, where each state has an emission distribution function. Specifically, the HMM consists of the following components:

- States. The set of hidden states in the HMM are denoted as $S = \{1, 2, 3, ..., N\}$. Given a data sequence, each time step t has a state label s_t. In the case of speech recognition, the realization of an acoustic unit (e.g., phone) is formulated as a HMM, where the states represent sub-units inside the acoustic unit.
- Initial probabilities. The initial state distribution describes the probability of each state serving as the initial state for a given sequence. More formally, the initial probability π_i

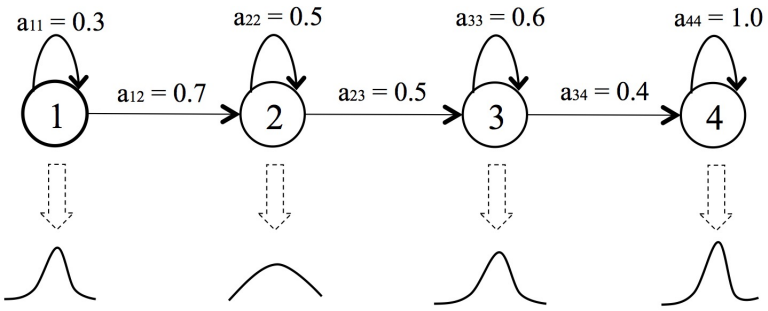


Figure 2.6: An example diagram of the HMM for the phone "A".

for state i is $\pi_i = Pr(s_0 = i)$ with the constraint $\sum_{i=1}^{N} \pi_i = 1$.

- Transition probabilities. The transition probabilities describe the probability of one state transitioning to another state in the state space. More formally, the transition probability a_{ij} is represented as $a_{ij} = Pr(s_t = j|s_{t-1} = i)$ with the constraint $\sum_{j=1}^{N} a_{ij} = 1$.
- Output distribution. The output distribution $b_i(\mathbf{o}_t)$ characterizes the probability of emitting an observation $\mathbf{o}_t$ from a state i. In the case of speech recognition, the observation is an acoustic feature vector, and the distribution follows a continuous Gaussian or mixture of Gaussians distribution.

In order to further illustrate the way the HMM works, we show an example for the phone "A" in Figure 2.6. Following this topology, the HMM always starts with the state 1. Each state (except the ending state) can either jump to the next state or stick to the current state, with different probabilities. For example, the HMM can either jump to the state 2 with the probability of 0.7, or self-transition to itself with the probability of 0.3. It is worth noting that the constraint of these two branches only applies to the HMM used for speech processing. General-purpose HMMs do not have this constraint. An exception for the transition behavior is the ending state, e.g., state 4 in Figure 2.6. When the HMM reaches this ending state, the transition can only be self-transitions looping over the ending state. Each state has an attached distribution function, e.g., a Gaussian distribution as illustrated in Figure 2.6. At each step, after the state is sampled, the observation is generated from the state according to the corresponding distribution. All the states in Figure 2.6 have an emitting function. In practice, we may place non-emitting states in the topology, representing entry into and exit from the HMM. The HMM topology in Figure 2.6 only models a single phone. In continuous speech recognition, an utterance contains multiple phones which are modeled by connecting the individual HMMs into a sequence of HMMs.

2.2.2 Gaussian Mixture Models

In the tradition GMM-HMM paradigm for ASR, the emission probability function $b_i(\mathbf{o}_t)$ is realized via a Gaussian mixture model, i.e., a weighted sum of Gaussian component densities. Specifically, $b_i(\mathbf{o}_t)$ can be computed as:

$$b_i(\mathbf{o}_t) = \sum_{m=1}^{M_i} c_{im} N(\mathbf{o}_t; \mu_{im}, \Sigma_{im}) \tag{2.8}$$

where M_i is the total number of Gaussian components in the state i, c_{im} is the mixture weight for Gaussian m of state i. Since the observation $\mathbf{o}_t$ is a vector, N is a multivariate normal distribution which has the mean vector μ_{im} and covariance matrix Σ_{im}. The mixture weights within a state need to sum to 1, i.e., $\sum_{m=1}^{M_i} c_{im} = 1$.

With the GMM-HMM model parameterized, we can evaluate the likelihood of an observation sequence O, i.e., the probability of O being generated by the GMM-HMM model. For a particular observation sequence, the state sequence S is unknown. For this evaluation, we need to sum up the probabilities with respect to all the possible state sequences. Due to the Markov property, the likelihood can be decomposed onto the level of each step. Specifically, the likelihood of O can now be computed as

$$Pr(O|\Omega) = \sum_{S} \prod_{t} b_{s_t}(\mathbf{o}_t) a_{s_t s_{t+1}} \tag{2.9}$$

where Ω represents all the models parameters in the GMM-HMM, including both the HMM and GMM parameters.

2.2.3 Training of GMM-HMM

When building an ASR system, training of the acoustic model requires training data which are collected in the form of speech segments as well as their corresponding word transcripts. With a dictionary, the word-level transcripts can be converted into phone-level. The HMMs for individual phones are concatenated to form the HMM sequence for the entire utterance. To train the GMM-HMM model, we need either to estimate the posterior probability of each observation being generated from every state via the commonly-used Baum-Welch algorithm [91], or to derive the hidden state sequence via the Viterbi algorithm [91]. With the state occupancy, statistics are collected from the training observations. Then the parameters of the GMM model are estimated through the standard expectation-maximization (EM) algorithm.

Chapter 3

Deep Learning for Speech Recognition

Deep learning has become the state of the art for acoustic modeling in ASR. On a wide range of tasks, deep learning has shown notable improvements over the traditional GMM-HMM paradigm. In this chapter, we first give a brief review of DNN acoustic models, for both hybrid models and BNF generation. Then, more advanced deep learning models, i.e., CNNs and RNNs, are described for the task of acoustic modeling.

3.1 Overview

Figure 3.1 illustrates the ASR pipeline where we apply deep learning models as the acoustic models. The pipeline always starts by building an initial GMM-HMM model. On top of the GMM-HMM, we can train deep learning models as the acoustic models. In practice, the GMM-HMM provides frame-level alignments from which we can derive the target labels (usually context-dependent states) for each speech frame. Generally, there are two principal ways by which deep learning architectures can be used. First, given the training data, we treat deep learning models as phonetic states classifiers. The models are trained to classify speech frames into phonetic states. After training, the models can be applied during decoding directly. This approach is referred to as the *hybrid* system, as it is a hybrid between the HMM and deep learning models. Second, the deep architectures are used as discriminative feature extractors. A commonly-used manner is to place a bottleneck layer in the architecture. After model training, activations from the bottleneck layer are taken as new feature representations, on which the traditional GMM-HMM models are constructed. In previous work, the resulting GMM-HMMs are referred to as *tandem* systems. More details regarding these two paradigms can be found in Section 3.

As can be seen from Figure 3.1, the ASR pipeline, even using the deep learning models, is

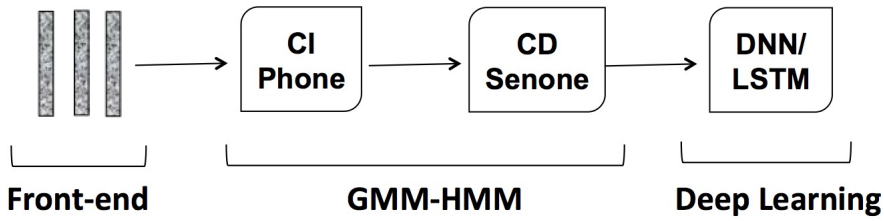


Figure 3.1: Pipeline of building ASR systems when deep learning is exploited for acoustic modeling.

intrinsically complicated. The complexity comes from different levels. First of all, the pipeline contains multiple training stages. Deep learning models can be applied only after the GMM-HMM model has been completely built. For building the GMM-HMM, we start with building the context-independent model, and then move on to the context-dependent model. Even within the same stage, we might need to perform multiple sub-stages. For example, on top of the initial GMM-HMM, more advanced training steps, e.g., speaker adaptive training (SAT) and discriminative training (DT) may need to be adopted. Second, for acoustic modeling, we need to prepare various resources, e.g., dictionaries and phonetic decision trees. These resources are not always available, especially for low-resource languages. The lack of these resources can hamper the deployment of ASR to these languages. Last but not least, we have quite a few hyper-parameters in the pipeline, e.g., the number of HMM states and the number of Gaussian components inside each state. Deciding the values of these hyper-parameters highly relies on the knowledge and expertise of ASR experts.

3.2 DNN Models

The architecture of the DNN we use is shown in Figure 3.2. A DNN is an multilayer perceptron (MLP) which consists of many hidden layers before the softmax output layer. Each hidden layer computes the outputs of conditionally independent hidden units given the input vector. We denote the feature vector at the *t*-th frame as $\mathbf{o}_t$. Normally $\mathbf{o}_t$ is the concatenation of multiple neighbouring frames centered at *t*. The quantities shown in Figure 3.2 can be computed as:

$$\mathbf{a}_t^i = \mathbf{W}_i \mathbf{x}_t^i + \mathbf{b}_i \qquad \mathbf{y}_t^i = \sigma(\mathbf{a}_t^i) \qquad 1 \leq i \leq L \tag{3.1}$$

where L is the total number of layers, the weight matrix $\mathbf{W}_i$ connects the *i*-1-th and *i*-th layers, $\mathbf{b}_i$ is the bias vector of the *i*-th layer, $\sigma(x)$ is the activation function. For $1 \leq i < L$ and $i = L$,

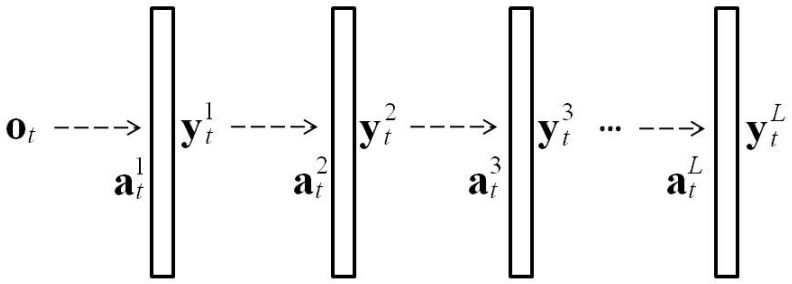


Figure 3.2: Architecture of the DNN model.

$\sigma(x)$ takes the form of the logistic sigmoid and softmax functions respectively. The inputs to the i-th layer $\mathbf{x}_t^i$ can be formulated as:

$$\mathbf{x}_t^i = \begin{cases} \mathbf{o}_t & i = 1 \\ \mathbf{y}_t^{i-1} & 1 < i \leq L \end{cases} \tag{3.2}$$

When applied as a hybrid model, the DNN is trained to classify each speech frame to CD tied states. Suppose that we use the cross-entropy as the loss function and the training set contains T frames. DNN training involves minimizing the following objective:

$$\mathbb{L} = -\sum_{t=1}^{T}\sum_{s=1}^{S} \mathbf{g}_t(s) \log \mathbf{y}_t^L(s) \tag{3.3}$$

where S is the total number of CD states (classes), $\mathbf{g}_t$ is the ground-truth label vector on frame t which is obtained via forced alignment with an existing GMM/DNN model, $\mathbf{y}_t^L$ is the output vector of the softmax layer. Error back-propagation is commonly adopted to optimize this objective. The gradients of the model parameters can be derived from the derivatives of the objective function with respect to the pre-nonlinearity outputs $\mathbf{a}_t^i$. At the softmax layer, the error vector for frame t is:

$$\boldsymbol{\epsilon}_t^L = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_t^L} = \mathbf{y}_t^L - \mathbf{g}_t \tag{3.4}$$

At each of the previous layers, we have the errors as

$$\boldsymbol{\epsilon}_t^i = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_t^i} = \mathbf{W}_{i+1}^T \boldsymbol{\epsilon}_t^{i+1} \odot \mathbf{y}_t^i \odot (1 - \mathbf{y}_t^i) \tag{3.5}$$

where $\odot$ represents element-wise multiplication. In practice, we use mini-batch based stochastic gradient descent (SGD) as the optimizer. In this case, model parameters are updated with gradients accumulated over the entire mini-batches.

Outputs from the whole DNN architecture represent the posterior probabilities of HMM states given the input $\mathbf{o}_t$. During decoding, we in fact need the emission probability of the feature vector with respect to each state. According to the Bayes rule, the observation probability given each state can be computed as:

$$p(\mathbf{o}_t|s) \propto \mathbf{y}_t^L(s)/p(s) \tag{3.6}$$

where $p(s)$ is the prior probability of state s which can be estimated from the alignment of the training data.

In hybrid models, DNNs are used as classifiers with respect to CD states. Alternatively, DNNs can also be employed as discriminative feature extractors. In the traditional tandem systems [37], a DNN (or MLP) is trained to classify context-independent (CI) states. The outputs from the DNN are projected down to a low-dimensional space with principal component analysis (PCA). The projected features are treated as the news features, over which the standard GMM models can be built. To avoid loss of discriminative information during PCA, [94] achieves dimension reduction by adopting a deep autoencoder following the DNN. The autoencoder network takes the DNN outputs as the inputs, and is trained to minimize the difference between these inputs and the outputs from this autoencoder. Outputs from the narrow layer of this autoencoder network are fed to GMMs as the new features.

When the DNN outputs are used in tandem systems, each of its hidden layers has the same number of units. A large amount of work [24, 25, 32, 123] has attempted to train the DNN with a bottleneck layer, a hidden layer which is significantly narrower than the other hidden layers. When training finishes, outputs from this narrow layer are taken as the new BNF features. Training of the BNF-DNN follows the same protocol as training of the standard DNN. The bottleneck layer acts to squeeze the discriminative information into a highly low-dimensional space.

In our previous work [25], we have established the deep BNF (DBNF) architecture for more effective BNF extraction. Our method differs from the previous BNF proposals [32, 123] in that the hidden layers are arranged in a non-symmetric manner around the bottleneck layer. In the DNN architecture, we insert multiple hidden layers prior to the bottleneck layer, whereas only one hidden layer is placed between the bottleneck and the softmax layers. As discovered in [125], activations from the higher layers of a DNN are more robust to variations and distortions from the speech signal. Therefore, placing the bottleneck layer at a high layer generates both discriminative and invariant feature representations that benefit the subsequent GMM training. Figure 3.3 depicts the architecture of our DBNF network.

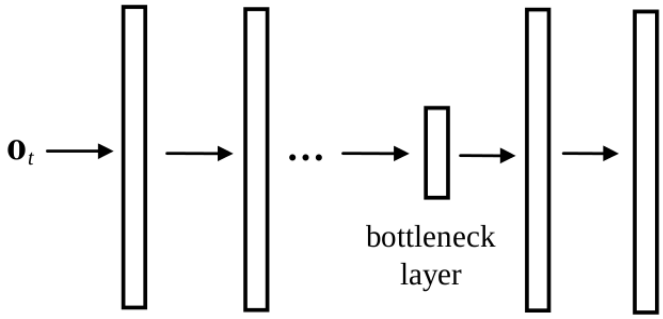


Figure 3.3: Architecture of the DBNF network.

3.3 CNN Models

CNNs have been applied widely in the areas of image processing and computer vision [51]. In the time-delay neural networks (TDNNs) for phone recognition [117], the convolution operation is applied on the time dimension of acoustic frames. In recent years, CNNs have been proved to outperform DNNs on large scale acoustic modeling tasks [3, 4, 95, 96, 107]. Instead of using fully-connected parameter matrices, CNNs are characterized by parameter sharing and local feature filtering. The local filters help to capture locality along the frequency bands. On top of the convolution layer, a max-pooling layer is usually added to normalize spectral variations. As a result, the CNN hidden activations become invariant to various types of speech and non-speech variability.

Figure 3.4 exemplifies a convolution layer, as well as a max-pooling layer applied atop. In the convolution layer, we only consider filters along frequency, assuming that the time variability can be modeled by HMM. Inputs into CNNs are N neighbouring frames of acoustic features (e.g., filterbanks), where each frame $\mathbf{v}_i$ is a one-dimensional feature map. The hidden outputs from this layer contain J vectors ($[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_J]$). The trainable one-dimensional filter $\mathbf{r}_{ji}$ connects input feature map $\mathbf{v}_i$ and output feature map $\mathbf{h}_j$, and is shared across the frequency axis along $\mathbf{v}_i$. Outputs from this convolution layer can be computed as

$$\mathbf{h}_j = \sigma(\sum_{i=1}^{N} \mathbf{r}_{ji} * \mathbf{v}_i + \mathbf{b}_j) \tag{3.7}$$

where $*$ represents the one-dimensional discrete convolution operator, and $\mathbf{b}_j$ is the trainable bias attached to $\mathbf{h}_j$. In this chapter, we use the logistic sigmoid activation function σ.

Then, a max-pooling layer is added on top of the convolution layer. Max-pooling is carried out in a vector-wise mode. More formally, for each vector $\mathbf{h}_j$, we divide its units into non-

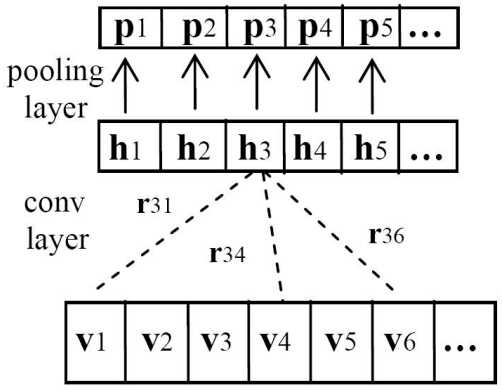


Figure 3.4: Convolution and max-pooling layers in the CNN architecture.

overlapping groups and output the maximum activation within each group. When the pooling size is k, the size of each after-pooling feature map $\mathbf{p}_j$ is $1/k$ of the size of the before-pooling $\mathbf{h}_j$. The convolution and pooling layers together are called a *convolution stage*. In our setups, CNNs stack two such stages where outputs from the lower pooling layer are propagated to the higher convolution layer. Multiple fully-connected DNN layers and finally the softmax layer are added over these two stages. From the feature learning perspective, the convolution and pooling layers in this structure are trained to extract invariant features, while the fully-connected layers use these high-level features to better classify HMM states.

3.4 RNN Models

DNNs and the follow-up CNNs have set the state of the art for large-scale acoustic modeling tasks. However, both DNNs and CNNs can only model the limited temporal dependency within the fixed-size context window. To resolve this limitation, previous work [30, 62, 97, 98] has studied the application of RNNs to acoustic modeling.

Compared to the standard feedforward architecture, RNNs have the advantage of learning complex temporal dynamics on sequences. Given an input sequence $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_T)$, a traditional recurrent layer iterates from $t = 1$ to T to compute the sequence of hidden states $\mathbf{H} = (\mathbf{h}_1, ..., \mathbf{h}_T)$ via the following equations:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{3.8}$$

where $\mathbf{W}_{hx}$ is the input-to-hidden weight matrix, $\mathbf{W}_{hh}$ is the hidden-to-hidden (recurrent) weight matrix. In addition to the inputs $\mathbf{x}_t$, the hidden activations $\mathbf{h}_{t-1}$ from the previous time step are

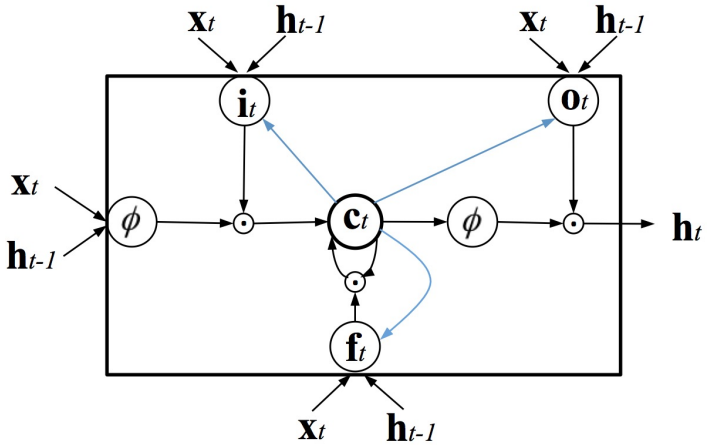


Figure 3.5: A memory block of LSTM.

fed to influence the hidden outputs at the current time step. Learning of RNNs can be done using back-propagation through time (BPTT). However, in practice, training RNNs to learn long-term temporal dependency can be difficult due to the well-known vanishing and exploding gradients problem [7]. Gradients propagated though the many time steps (recurrent layers) decay or blow up exponentially. The LSTM architecture [40] provides a solution that partially overcomes the weakness of RNNs. LSTM contains memory cells with self-connections to store the temporal states of the network. Additionally, multiplicative gates are added to control the flow of information: the input gate controls the flow of inputs into the memory cells; the output gate controls the outputs of memory cells activations; the forget gate regulates the memory cells so that their states can be forgotten. Furthermore, as research on LSTM has progressed, the architecture is enriched with peephole connections [26]. These connections link the memory cells to the gates to learn precise timing of the outputs.

Given the input sequence, a LSTM layer computes the gates (input, output, forget) and memory cells activations sequentially from t = 1 to T. The computation at the time step t can be described as:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{3.9a}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{3.9b}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \phi(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{3.9c}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \tag{3.9d}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \phi(\mathbf{c}_t) \tag{3.9e}$$

where $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$ are the activation vectors of the input gate, output gate, forget gate and memory cell respectively. The $\mathbf{W}_{.x}$ terms denote the weight matrices connecting the inputs with the units. The $\mathbf{W}_{.h}$ terms denote the weight matrices connecting the memory cell outputs from the previous time step $t-1$ with different units. The terms $\mathbf{W}_{ic}$, $\mathbf{W}_{oc}$, $\mathbf{W}_{fc}$ are diagonal weight matrices for peephole connections. Also, σ is the logistic sigmoid nonlinearity which squashes its inputs to the [0,1] range, whereas ϕ is the hyperbolic tangent nonlinearity squashing its inputs to [-1, 1]. The operation $\odot$ represents element-wise multiplication of vectors.

In Section 6.2.3, we propose to perform the extraction of the speaker-microphone distance information with RNNs. However, we only consider DNNs and CNNs as the acoustic models when various types of context information are incorporated into acoustic modeling. Excluding RNNs as acoustic models is mainly because RNNs normally take single frames as network inputs, whereas the inputs of DNNs and CNNs consist of concatenation of multiple neighboring frames. When combined with the single-frame inputs, the context information (e.g., 100-dimensional i-vectors, distance and visual features) can easily outweigh the acoustic features (e.g., 40-dimensions filterbanks). This combination disproportionately undermines the weight of the acoustic features and eventually hurt the performance of ASR systems.

Chapter 4

Cross-Language DNNs with Language-Universal Feature Extractors

As discussed in Section 1.1, DNN acoustic models normally contain much more parameters than their GMM counterparts. To get good recognition performance, training of DNN models generally requires a large amount of training data. However, adequate transcribed speech is not always available, e.g., when we construct ASR systems on a low-resource language or a new domain. This data scarcity can cause overfitting easily during DNN training, which degrades the performance of DNN models on unseen testing data. The sensitivity of DNN training to data scarcity is experimentally demonstrated in [72], where DNNs fail to outperform GMMs with only 10 hours of training speech.

In this chapter, we focus on improving DNN models under low-resource languages. Our solution is to perform cross-language DNN acoustic modeling by borrowing knowledge from other languages. Knowledge transfer across languages is achieved via language-universal feature extractors (LUFEs) trained over a group of source languages. After reviewing related work, we first establish our cross-language DNN framework. Then, a series of improvements are made to enhance the quality of LUFEs and speed up the training of LUFEs. Our work described in this chapter has been published in [68, 69, 72, 76]

4.1 Related Work

Previous work has proposed various methods to improve DNNs under low-resource conditions. A potential solution is to build sparse DNNs [124], either through regularizing hidden layer

parameters or through rounding tiny parameters to zero. Although speeding up model training, sparse DNNs fail to improve recognition performance. Meanwhile, dropout is presented as a useful strategy to prevent overfitting in DNN fine-tuning [39]. Random dropout is observed to perform effectively on phone recognition [14] and LVCSR [68, 129], displaying special benefits when language resources become highly limited. In [72], the maxout networks are applied as an alternative to standard DNNs for acoustic modeling. The hidden units at the maxout layer are divided into disjoint groups and each group outputs a single activation. This reduces the number of hidden-layer outputs and therefore model parameters. Maxout networks are demonstrated to be particularly effective for low-resource acoustic modeling.

Another long-standing solution is to share/borrow knowledge across languages. This knowledge transfer has been traditionally realized by the use of a global phone set shared by all the languages [100]. For GMM models, the subspace Gaussian mixture models (SGMMs) [87] have been exploited extensively for multilingual and cross-language acoustic modeling. Instead of learning GMM parameters, the SGMM learns low-dimensional subspaces which capture the main phonetic and speaker variability. In a multilingual setting, the SGMM subspace parameters can be estimated with combined statistics from multiple languages [9]. Then, these subspace parameters are transferred to a low-resource target language on which only the non-subspace parameters need to be estimated [61]. This effectively reduces the number of parameters on the target language and improves the robustness of model training. Along this line of work, other techniques [74] have been proposed to increase the flexibility of the multilingually-trained subspace parameters on the target language. In [82], a systematic study is conducted to investigate the effect of adding multilingual data in different training stages, e.g., network pre-training or network fine-tuning. Also, it is interesting to observe the impact of the selection of the languages and the amount of training speech from each language. In order to make better use of multilingual training data, [81] proposes to explicitly provide a language code to DNNs. With this language information, DNN training becomes aware of various languages and learns language-specific representation. The resulting language-adaptive DNN (LA-DNN) model gains consistent improvement over the baseline monolingual acoustic models.

In [125], the hidden layers of DNNs are treated as a series of nonlinear transforms that convert the original input features into a high-dimensional space. The final softmax layer is added as a linear classifier for state classification. It is revealed in [125] that the effectiveness of DNNs comes largely from the invariance of the representations to variability such as speakers, environments and channels. Following this feature learning formulation, [41, 68] view the DNN hidden layers as a deep feature extractor that is jointly trained over multiple languages. The resulting

multilingual DNN outperforms the monolingual DNN trained using language-specific data.

Our work in this thesis follows the feature learning formulation and focuses on cross-language acoustic modeling with DNNs. In [41], the authors investigate the application of multilingual DNNs for cross-language modeling. Specifically, the feature extractor that has been trained with multilingual data is transferred to a new language. On this new language, a softmax layer is added atop of the feature extractor and fine-tuned with the new-language data. Although giving nice gains [41], this approach has limitations: only fine-tuning a single softmax layer may not give us enough modeling power. Our thesis extends this approach to a more general framework that is characterized by greater flexibility on the new language. More importantly, we develop our framework in different ways, from improving recognition results on the new language to speeding up the multilingual training of DNNs.

4.2 Cross-Language DNNs with LUFEs

Our framework is illustrated in Figure 4.1. On the left of Figure 4.1, a multilingual DNN is learned over a group of source languages. The hidden layers of the multilingual DNN are shared across all the languages, whereas each language has its own softmax layer to classify CD states specific to that language. Fine-tuning of the multilingual DNN is carried out using the standard mini-batch based SGD. The difference is that each epoch traverses data from all the source languages, instead of one single language. The SGD estimator loops over languages iteratively, each time picking one mini-batch from a language. At the same time, it switches to the softmax layers and class labels corresponding to the language from which the current mini-batch comes. Parameters of the shared layers are updated with gradients accumulated from all the languages.

After the multilingual DNN is trained, the shared hidden layers serve as the LUFE. Suppose that we have a new language as shown on the right of Figure 4.1. This LUFE is applied to this new language, transforming the raw acoustic features (e.g., MFCCs or filterbanks) to high-level feature representations. A hybrid DNN model is trained on this new language to classify the CD states. This DNN model takes feature representations generated from the LUFE as the inputs. The LUFE is fixed during the process of new-language DNN training. That is, the parameters of the LUFE are not re-updated on the new language. This manner of cross-language acoustic modeling enables knowledge transfer across languages. Thus, it improves the recognition performance on the new language, especially when the new language has limited transcribed speech. This framework differs from the method in [41] in that [41] estimates a softmax layer on the new

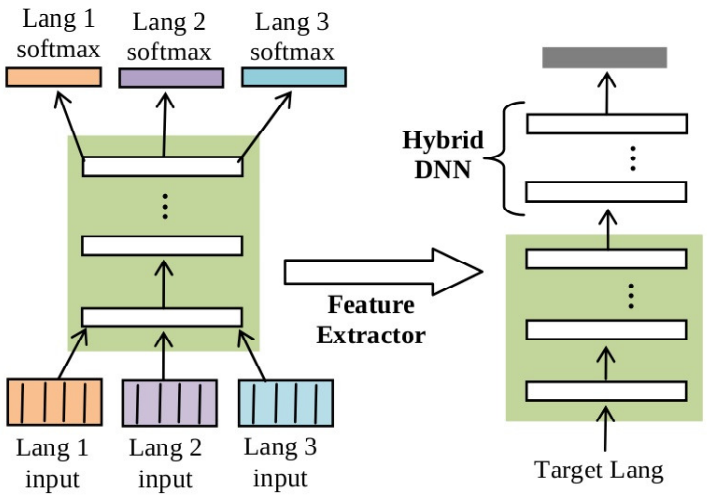


Figure 4.1: Cross-language DNNs with the LUFE.

language. In comparison, in our framework, a fully-connected DNN model is constructed on the new language, with LUFE outputs as DNN inputs. This modification results in greater flexibility and is empirically proved to give notable improvement on the new language.

4.3 Improving LUFEs with Deep Convolutional and Maxout Networks

The quality of LUFEs plays a crucial role in our cross-language DNN framework. LUFEs have been trained conventionally with the multilingual DNNs. In this section, we explore two strategies to further improve LUFEs. First, we replace the standard sigmoid nonlinearity with the recently proposed maxout units. The resulting maxout LUFEs have the nice property of generating sparse feature representations. Second, the CNN architecture is applied to obtain more invariant feature space.

4.3.1 LUFEs with Convolutional Networks

As discussed in Section 3.3, CNNs can normalize variability of the speech signal more effectively than DNNs. Therefore, CNNs outperform DNNs on a variety of acoustic modeling tasks [3, 4, 95, 96, 107]. This motivates us to apply CNNs as the building block of LUFEs. Our CNN architecture used in LUFE training follows the previous studies [3, 95, 96, 107]. It has two convolution layers each of which is followed immediately by a max-pooling layer. Multiple

fully-connected hidden layers are added atop of the final max-pooling layer. Finally we have the softmax layer for classification.

When using CNNs, learning of the LUFE involves training a multilingual CNN. The training process also exploits the parameter sharing idea. The hidden convolution and fully-connected layers are shared and collaboratively trained over the source languages. When the training finishes, these shared hidden layers are taken as the LUFE and transferred to the new language. The structure of the convolution and max-pooling layers have been described in Section 3.3.

4.3.2 Sparse Feature Extraction with Maxout Networks

Sparse feature learning is an active research topic in the machine learning field. On the complex speech signal, sparse features (e.g., sparse coding) [52, 105, 106] tend to give better classification accuracy compared with the raw, non-sparse features. Within the LUFE framework, we propose to achieve sparse feature extraction by taking advantage of the deep maxout networks (DMNs) [72]. Maxout networks [28] are originally presented as an alternative to DNNs for object recognition. In [72], a first attempt is made to apply maxout networks to acoustic modeling. The application of maxout networks (and their variants) to acoustic modeling is then further extended in [130].

Figure 4.2 depicts the i-th layer in a maxout network. The hidden units are divided into non-overlapping groups. We denote the number of unit groups as U and the group size (how many units each group contains) as g. Given the input feature vector $\mathbf{o}_t$, the maxout function is imposed to generate this layer's activation $\mathbf{y}_t^i$=[$\mathbf{y}_t^i(1)$, $\mathbf{y}_t^i(2)$, ..., $\mathbf{y}_t^i(U)$] is a U-dimensional vector. By following the notations in Section 3.2, we compute the maxout-layer outputs as:

$$\mathbf{y}_t^i(u) = \max_j(\mathbf{a}_t^i(j)) \qquad (u-1) \times g + 1 \leq j \leq u \times g \tag{4.1}$$

We can see that the maxout function in fact applies a max-pooling operation on the pre-activation hidden outputs $\mathbf{a}_t^i$. The maximum value within each group is taken as the output from the i-th layer. A DMN can be constructed by connecting multiple maxout layers consecutively.

In this thesis, we employ DMNs as sparse feature extractors. Sparse representations can be generated from any of the maxout layers via a non-maximum masking operation, as exemplified by Figure 4.2. Specifically, given an input frame, all the units within each group have their individual outputs, instead of being pooled together into one output. However, only the maximum value in this group is retained. All the other non-maximum values are rounded to 0. It is worth

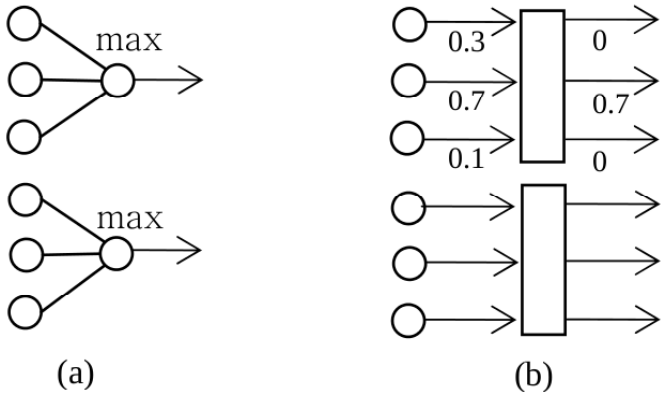


Figure 4.2: An example for (a) maxout layer and (b) non-maximum masking.

noting that non-maximum masking only happens during the feature extraction stage. The training stage always applies max-pooling.

In order to measure feature sparsity quantitatively, we compute *population sparsity* [83] for each feature type. If the t-th frame has the feature vector $\mathbf{f}_t$, then population sparsity

$$pSparsity = \| \frac{\mathbf{f}_t}{\|\mathbf{f}_t\|_2} \|_1 \tag{4.2}$$

measures how sparse the features are on this example. In our experiments, we report the averaged value of this metric over the entire new-language training set. Unless otherwise stated, population sparsity is shortened as *pSparsity* throughout this thesis. A lower pSparsity means higher sparsity of the features.

So far, we have examined CNNs and DMNs separately as LUFEs. A natural extension is to combine DMNs and CNNs together, which enables LUFEs to generate both sparse and invariant feature representations. The resulting LUFE is structured in the similar way as the CNN-based LUFE. The only difference is that the fully-connected layers in CNNs are replaced with maxout layers. In Section 4.3.3, we experimentally show that this combined feature extractor ends up to be the best LUFE studied in this work.

4.3.3 Experiments

Experimental Setup

Our experiments follow the setup in [72], using the multilingual corpus collected under the IARPA BABEL research program [11, 12, 66]. We aim at improving ASR on the Tagalog (TG, IARPA-babel106-v0.2f) *limited language pack* (LimitedLP). This is a low-resource condition

Table 4.1: Statistics of the BABEL multilingual datasets.

	Target	Source		
	TG	CN	TU	PS
#training speakers	132	120	121	121
training (hours)	10.7	17.8	9.8	9.8
dictionary size	8k	7k	12k	8k
#classes	1920	1867	1854	1985

because only 10 hours of telephone conversational speech are available for system building. Moreover, the data collection covers a variety of acoustic conditions, speaking styles and dialects. A large portion of the audio data are either non-speech events (e.g., breath, laughter and cough) or non-lexical speech (e.g., hesitation and fragment). All these factors make acoustic modeling under this condition a very difficult task. Therefore, we get higher WERs [24, 25, 72] than on other benchmark datasets such as Switchboard.

On the target language Tagalog LimitedLP, WERs are reported on a testing set of 2 hours of speech. The training and testing sets have no over-lapping speakers. During decoding, we use a trigram language model built from training transcriptions. The source languages, on which LUFEs are trained, include the LimitedLP sets of Cantonese (CN, IARPA-babel101-v0.4c), Turkish (TU, IARPA-babel105b-v0.4) and Pashto (PS, IARPA-babel104b-v0.4aY). LimitedLP sets of the four languages have statistics summarized in Table 4.1.

On each language, we build the GMM models with the same recipe. An initial maximum likelihood model is first trained using 39-dimensional PLPs (plus deltas and double deltas) with per-speaker mean normalization. Then 9 frames are spliced together and projected down to 40 dimensions with linear discriminant analysis (LDA). Then, SAT is performed based on fMLLR. The number of triphone states for each language is shown in the last row of Table 4.1. Training of the DNN and CNN models is performed with the PDNN framework [67].

Inputs of DNNs and CNNs include 11 consecutive frames of 30-dimensional log-scale filterbanks with per-speaker mean and variance normalization. The DNN model has 5 hidden layers and 1024 units at each layer. DNN parameters are initialized with stacked denoising autoencoders (SDAs) [115] using masking noise and the denoising factor of 0.2. Each layer in the DNN corresponds to a denoising autoencoder (DA) which minimizes the difference between the reconstruction of corrupted input and the clean version of it. Pre-training of each layer has the learning rate of 0.01 and runs for 10 epochs. During network fine-tuning, we start from a learning rate of 0.08 which keeps unchanged for 15 epochs. Then the learning rate is halved at each

Table 4.2: WER(%) of monolingual DNN and CNN on the target language.

Models	WER%
Monolingual DNN	70.8
Monolingual CNN	68.2

Table 4.3: WER(%) of various LUFEs on the target language.

LUFE Models	WER%	pSparsity
DNN-LUFE	69.6	21.3
Method in [41]	70.1	21.3
CNN-LUFE	67.1	20.4
DMN-LUFE	67.5	17.7
CNN-DMN-LUFE	65.9	16.6

epoch until the cross-validation accuracy on a held-out set stops to improve.

The structure of the convolution layers in CNN-based LUFEs follows our description in Section 3.3 and Figure 3.4. We adopt one-dimensional convolution only along the frequency axis. The size of the filter vectors ($\mathbf{r}_{ji}$ in Equation (3.7)) is constantly set to 5. We use a pooling size of 2, meaning that the pooling layer shrinks convolution outputs by half. After tuning the CNN architecture, we observe that the best setting has two convolution stages and three fully-connected layers. The first and second convolution layers contain 100 and 200 feature maps respectively. Each of the fully-connected hidden layers contains 1024 units. Continuing to augment the convolution and fully-connected layers brings no further gains. Table 4.2 shows the performance of the monolingual DNN and CNN models on the target language. The CNN achieves 2.6% absolute WER improvement over the DNN model, which verifies the advantage of CNNs for acoustic modeling.

Results of DNN-based, CNN-based and DMN-based LUFEs

LUFEs are trained over the three source languages. For DNNs and DMNs, we take their hidden layers together as the LUFE. For CNNs, we take the two convolution layers (together with their corresponding max-pooling layers) and the lowest fully-connected layer as the LUFE. On the target language Tagalog LimitedLP, a DNN-based hybrid model is built on the feature representations from the LUFE. For fair comparison, the identical DNN topology, i.e., 4 hidden layers each with 1024 units, is used for hybrid model over different feature extractors. Table 4.3 shows the results of the hybrid models when various LUFEs are applied. We can see from Table 4.2 that cross-language models based on LUFEs give consistently better results than the monolin-

gual DNN. On the same setup, we also show the WER obtained by the method described in [41], where a single softmax layer is fine-tuned atop of the LUFE on the target language. We observe that our framework, which trains a complete DNN model on the target language, performs better than this method. Compared with the DNN-based LUFE, the CNN-based LUFE gives 2.5% absolute improvement, whereas the improvement obtained by the DMN-based LUFE is 2.1% absolute.

As with [72], during training of DMNs, the dropout technique [39] is applied in order to prevent overfitting. We impose dropout on each hidden layer by following the implementation described in [68]. The drop factor, which governs the binomial distribution for hidden activation masking, is constantly set to 0.2. For DMNs, we have 512 maxout units and the group size of 2. This configuration keeps the number of units at each hidden layer approximately to be 1024. Table 4.3 compares different LUFEs on pSparsity and target-language WERs. We can see that the DMN-based LUFE results in better WERs on the target language than the DNN-based LUFE. Also, the application of DMNs generates features with lower pSparsity, indicating that we are indeed extracting more sparse features from the DMN architecture.

Results of Combining CNNs and DMNs

Finally, we examine the effectiveness of combining CNNs and DMNs for better feature extraction. The structure of the convolution layers remains the same. We replace the sigmoid fully-connected layers with maxout layers. During multilingual training, the convolution layers and maxout layers use the starting learning rates of 0.08 and 0.1 respectively. Features are generated from the lowest maxout layer on top of the convolution layers. We can see from Table 4.3 that compared with the CNN-based LUFE, the CNN+DMN based LUFE generates sparse features, as well as reduction of target-language WER. This combined LUFE obtains 3.7% absolute WER improvement (65.9% vs 69.6%) over the baseline DNN-based LUFE.

More Results

For more complete evaluations, we further expand the previous experiments from the following two aspects. First, to make our experiments consistent, we train the LUFEs by taking the FullLP sets of Cantonese, Turkish and Pashto as the source languages. The total amount of LUFEs training data is around 240 hours. Inputs into the LUFE networks are 11 consecutive frames of 30-dimensional log-scale filterbanks with per speaker mean and variance normalization. On the target language, activations from the last hidden layer (which is followed by the classification

layer) are taken as the new feature representations. Second, we evaluate the LUFE framework on both the LimitedLP and FullLP Tagalog as the target languages. This gives us insight on how our framework performs under various levels of data availability on the target language. The configurations for the individual LUFE models are as follows.

- **DNN**. The DNN-based LUFE has 6 hidden layers, each of which contains 1024 hidden units. All the hidden layers use the sigmoid activation function.
- **CNN**. The CNN architecture consists of 2 convolution and 4 fully-connected (FC) layers. We apply 2-dimensional convolution over both time and frequency. The CNN inputs are 30-dimensional filterbank features with their deltas and double-deltas features, with a temporal context of 11 frames. The first convolution layer filters the inputs using 256x3 kernels each of which has the size of 9x9. This is followed by a max-pooling layer only along the frequency axis, with the pooling size of 3. The second convolution layer takes as inputs the outputs from the pooling layer and filters them with 256x256 kernels with the size of 4x3. We then place 4 FC hidden layers and finally the softmax layer on top of the convolution layers. Both the convolution and the FC layers use the logistic sigmoid activation function.
- **CNN+DMN**. The CNN+DMN based LUFE has a similar architecture to the CNN. The only difference is to replace the 4 FC layers with maxout layers. Each maxout layer has 512 maxout units and the group size of 2.

The results are presented in Table 4.4, from which we can get the following observations. First, the comparisons between the LUFE models are consistent with our observations when the LUFEs are trained with LimitedLP sets of the source languages. That is, the CNN-based LUFE outperforms the vanilla DNN-based LUFE, and combining CNNs and maxout networks gives us the best LUFE. Second, by comparing the two target-language conditions, we observe that the improvement of applying LUFEs on the FullLP Tagalog becomes less significant than on the LimitedLP Tagalog. This is understandable because the LimitedLP Tagalog is a typical low-resource condition, where incorporating knowledge from other languages via LUFEs is most beneficial.

4.4 Distributed Training

Ideally, LUFEs are trained over multiple languages with adequate speech data. However, SGD-based optimization is sequential and hard to be parallelized. This makes LUFE learning an

Table 4.4: Performance of various LUFE models trained with the FullLP sets of the source languages. The target languages are the LimitedLP and FullLP conditions of Tagalog respectively.

LUFE Models	Target(LimitedLP Tagalog) WER%	Target(FullLP Tagalog) WER%
None	65.8	49.3
DNN	59.6	46.7
CNN	58.1	44.9
CNN+DMN	57.0	44.6

expensive task, even with the powerful GPU cards. In this section, we aim at speeding up LUFE training with multiple GPUs, and a parallelization scheme is presented to accomplish this.

4.4.1 DistModel: Distribution by Models

Our DistModel strategy is developed based on the model averaging idea. Model averaging has been exploited for distributed learning problems, for both convex and non-convex models [64, 130]. We port this idea to distributed training of LUFEs on GPUs. The implementation is straightforward. Training data of each language is partitioned evenly across all the GPU threads. In Figure 4.3, we show an example which includes 3 source languages. Each of the languages contains 90 hours of training data. When distributing training to 3 GPUs, we assign to each GPU 90 hours of data which consist of 30 hours from each source language. Different GPUs have no overlapping on their data. Then, each GPU trains a LUFE as described in Section 4.2. After a specified number of mini-batches, the instances of LUFEs from the individual GPUs are averaged into a unified model. We refer to the number of mini-batches between two consecutive averaging operations as *averaging interval*. Note that both the language independent (shared hidden layers) and the language specific (softmax layer) parameters are averaged. The averaged parameters are sent back to each GPU as the new starting model for the subsequent training.

DistModel is inherently time synchronous in that the parallel threads have to wait for each other to perform model averaging. This tends to cause delay, especially when the frequency of model averaging is high or certain computing nodes run slowly. Compared with the more popular asynchronous methods [15, 36], time synchronous methods generally achieve worse acceleration. However, we discover that on this particular LUFE learning task, DistModel is robust to large averaging interval up to 2000 mini-batches. This is partly because multi-task learning performed by each GPU acts as strong regularization for LUFE training. This prevents the multilingual DNN models from getting stuck into local optima. As a result, the averaged LUFE still provides unbiased feature representations, even after SGD has processed many mini-batches of training

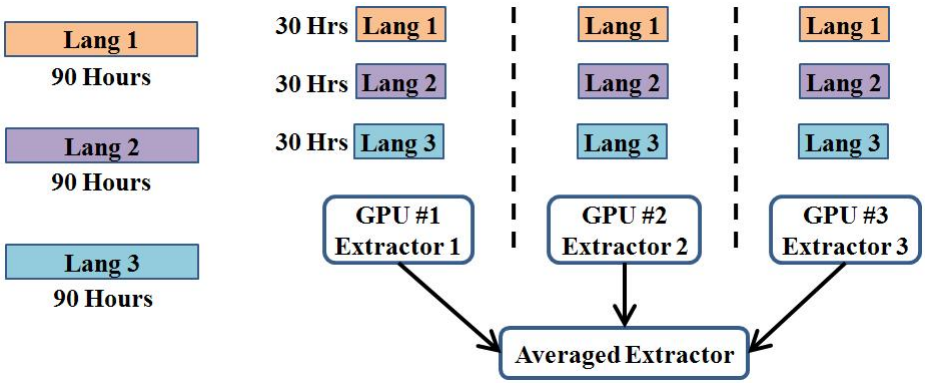


Figure 4.3: The DistModel distributed learning strategy for LUFEs.

examples on each GPU. Because of this, delay resulting from model averaging ends up to be a tiny fraction of the entire training time.

4.4.2 Experiments

As with Section 4.3.3, the effectiveness of the DistModel strategy is evaluated with the BABEL corpus. The source languages include the FullLP sets of Cantonese (IARPA-babel101-v0.4c), Turkish (IARPA-babel105b-v0.4) and Pashto (IARPA-babel104b-v0.4aY). Each source language consists of approximately 80 hours of transcribed speech. Our target language is the LimitedLP condition of Tagalog which has 10 hours of data. GMM and DNN models are built using the same recipes as described in Section 4.3.3. On the target language, the monolingual DNN hybrid model has a WER of 65.8% on the 2-hour testing set. Note that the WER obtained by our DNN baseline differs from the WER of the DNN baseline in Section 4.3.3. This is because we are using a different testing set and the officially-released scoring setup. With the DNN-based LUFE, we are able to reduce the WER down to 59.6% if the LUFE is trained on a single GPU. That is, cross-language acoustic modeling with LUFEs brings 9.4% relative improvement (59.6% vs. 65.8%).

We evaluate DistModel over 3 GPUs. A key variable in the DistModel scheme is the averaging interval. Table 4.5 shows speed-up of LUFE training and WERs of the target language when DistModel adopts various values for averaging interval. Speed-up is measured by the ratio of the training time taken using a single GPU to the time using 3 GPUs. As expected, with larger averaging interval, we obtain monotonically better speed-up because of less model averagings. The change of WER displays more fluctuation, especially for averaging interval less than 2000. When averaging interval equals 2000, LUFE learning with 3 GPUs is $2.6\times$ faster than using a

Table 4.5: Impact of averaging interval on WERs and training speed-up.

Averaging Interval	WER%	Training Speed-up
300	59.4	1.32
600	60.0	1.89
1000	59.8	2.25
1500	60.5	2.49
2000	59.7	2.66
3000	60.6	2.84
Epoch	61.2	2.9

Table 4.6: DistModel applied to monolingual Tagalog FullLP DNN training.

Method	WER%	Training Speed-up
Single GPU (baseline)	49.3	—
DistModel - 300	50.1	1.5
DistModel - 600	50.5	1.9
DistModel - 1000	50.5	2.2
DistModel - 2000	50.3	2.5
DistModel - 3000	50.8	2.7

single GPU. At the same time, the trained LUFE achieves a WER of 59.7% on the target language. This corresponds to 0.1% absolute degradation which can be considered negligible given the baseline 59.6%. Continuing to increase averaging interval gives further speed-up but significantly worse WERs. Thus, setting averaging interval to 2000 is a good balance between training efficiency and recognition performance. A contrast experiment is to train the LUFE with a single GPU and only one third of the data. In this case, we can get perfectly $3\times$ speed-up. However, the WER on the target language goes up to 62.2%.

To investigate DistModel more closely, we apply DistModel to the Tagalog FullLP set for the normal monolingual DNN training. The resulting DNN model is used directly as hybrid models, rather than for feature extraction. Table 4.6 shows speed-up of DNN training and the resulting WERs with averaging interval varying from 300 to 3000. In this scenario, DistModel achieves similar speed-up as for multilingual DNN training. However, WER degradation caused by parallelization becomes more significant compared with Table 4.5. This reveals that our proposed DistModel strategy is particularly suited for the task of LUFE learning.

Table 4.7: Performance of DistModel with 5 source languages. Distributed training uses 3, 4 and 5 GPUs respectively. WER(%) is reported on the Bengali 2-hour testing set.

Method	WER%	Training Speed-up
Monolingual DNN	72.5	——
Single GPU (baseline)	65.7	——
DistModel with 3 GPUs	66.2	2.4
DistModel with 4 GPUs	66.7	3.1
DistModel with 5 GPUs	66.8	3.4

Table 4.8: Performance of DistModel by taking other languages from the BABEL corpus as the target language. The LUFE feature extractor remains the same. WER(%) is reported on a 2-hour testing set for each individual target language.

Target Language	Monolingual WER%	Cross-lingual WER%
Bengali	72.5	66.2
Assamese	69.4	63.1
Haitian	64.3	60.9

4.4.3 Larger-Scale Evaluations

In this section, we extend DistModel to larger-scale evaluations by adding Tagalog (IARPA-babel106-v0.2f) and Vietnamese (IARPA-babel101-v0.4c) into the source languages. This finally gives us 460 hours of speech data for feature extractor training. Our target language now is the LimitedLP condition of Bengali (IARPA-babel103b-v0.4b). Also, 2 hours of speech from the Bengali decoding data are selected as the testing set. DistModel adopts the optimal configuration found in Section 4.4.1. Table 4.7 shows how DistModel performs when we scale it up to 5 GPUs. We observe consistent acceleration, although the speed-up fails to improve linearly with the number of GPUs. Meanwhile, pooling more GPUs into distributed learning causes WER degradation, which is likely to be mitigated by further optimization (e.g., learning rate, feature dimension) on DistModel.

Finally we select more languages released in the second-year period of BABEL as the target languages. The LUFE feature extractor remains the same, which has been trained on 5 source languages with 3 GPUs in parallel. Table 4.8 shows the results of the monolingual and corss-lingual DNN models on various target languages. We can see that across the different targets, we are able to achieve consistent improvement by transferring knowledge from source to target languages via the LUFE feature extractors. This proves the effectiveness of our cross-lingual DNN training framework in improving the ASR performance on low-resource languages.

4.5 Summary

In this chapter, we have proposed a framework to perform cross-language DNN acoustic modeling with LUFEs. This framework is further extended from two aspects. First, we have attempted to improve the quality of the LUFEs by applying CNNs and DMNs as the building block for LUFEs. These two architectures have the advantage of generating invariant and sparse feature representations respectively. Combining these two architectures gives us the best LUFE signified by the lowest WER on the target language. Second, we have proposed a distributed learning strategy DistModel to speed up training of LUFEs. DistModel accelerates LUFE learning significantly, while causing negligible WER loss on the target language.

Chapter 5

Speaker Adaptive Training of DNN Models using I-vectors

As discussed in Section 1.1, both GMMs and DNNs suffer from the mismatch between acoustic models and testing speakers. An effective procedure to alleviate the effect of speaker mismatch is speaker adaptation. For GMM models, speaker adaptation is generally performed by estimating the linear MLLR/fMLLR transforms specific to testing speakers [23, 53, 73]. Recently, a large amount of work has been dedicated to speaker adaptation of DNNs [58, 111, 121]. Another technique closely related with speaker adaptation is SAT [5, 6]. SAT performs adaptation on the training set and projects all the data into a speaker-adaptive space. In this new feature space, parameters of the acoustic models are further estimated. Acoustic models trained this way become independent of specific training speakers and thus generalize better to unseen testing data.

In this thesis, we propose a novel framework to carry out feature-space SAT for DNNs. Building of SAT-DNN models starts from an initial SI-DNN model that has been trained over the entire training set. Then, our framework uses i-vectors extracted at the speaker level as a compact representation of each speaker's acoustic characteristics. An adaptation neural network is learned to convert i-vectors to speaker-specific linear feature shifts. Adding the shifts to the original DNN input vectors (e.g., MFCCs) produces a speaker-normalized feature space. The parameters of the SI-DNN are updated in this new feature space, and this finally gives us the SAT-DNN model. This thesis explores the optimal configuration of SAT-DNNs for LVCSR tasks. Apart from hybrid models, we demonstrate the extension of our SAT framework to CNNs and BNF generation. Furthmore, we study the combination of SAT and speaker adaptation for DNNs. During decoding, model-space adaptation is applied on top of SAT-DNN models for further improved recognition results. Our work described in this chapter has been published in

[69, 77]

5.1 Related Work

This section reviews previous work that is related to our work. These previous work is divided into three aspects: speaker adaptation and SAT of the traditional GMM-HMM models, speaker adaptation and SAT of DNNs, and extraction of i-vectors.

5.1.1 Speaker Adaptation and SAT of GMM-HMM Models

Speaker Adaptation of GMM-HMMs

An issue that acoustic models, both GMMs and DNNs, encounter is the mismatch between acoustic models and testing speakers. Acoustic models experience a degradation of recognition accuracy when ported from training speakers to unseen testing speakers. Speaker adaptation has proven to be effective in mitigating the effects of this mismatch [23, 53]. In general, speaker adaptation modifies speaker-independent (SI) models towards particular testing speakers or transforms the features of testing speakers towards the SI models. Two commonly-used speaker adaptation methods for GMM-HMM models are maximum likelihood linear regression (MLLR) and feature-space MLLR (fMLLR).

MLLR is originally presented as a model-space adaptation technique. MLLR applies speaker-specific linear transforms to the mean vectors of the Gaussian components in the GMM model. For the component m in the state i, MLLR can be described as

$$\mu_{im} \rightarrow \mathbf{A}^{(s)}\mu_{im} + \mathbf{b}^{(s)} \tag{5.1}$$

where μ_{im} is the mean vector for the Gaussian component, $\mathbf{A}^{(s)}$ is a linear transform specific to the speaker s, and $\mathbf{b}^{(s)}$ is a bias vector specific to s. This can be rewritten into a more compact form

$$\mu_{im} \rightarrow \mathbf{W}^{(s)}\mu_{im}^{+} \tag{5.2}$$

where $\mathbf{W}^{(s)} = [\mathbf{A}^{(s)}; \mathbf{b}^{(s)}]$, μ_{im}^{+} is an expanded form of μ_{im}, i.e., $\mu_{im}^{+} = [\mu_{im}^{T}, 1]^{T}$.

The MLLR transform is typically estimated in a Maximum Likelihood fashion based either on known transcripts of a speakers utterances (so-called supervised adaptation) or on the text output from a speech recognition system (unsupervised adaptation). Suppose we have some

adaptation data on which we either have known transcripts of a speakers utterances (supervised adaptation) or have the recognition outputs from an ASR system (unsupervised adaptation). The MLLR transform is typically estimated in a maximum likelihood estimation (MLE) manner. The likelihood of each observation from the adaptation data of speaker s can be expressed as

$$b_i(\mathbf{o}_t) = \sum_{m=1}^{M_i} c_{im} N(\mathbf{o}_t; \mathbf{W}^{(s)} \mu_{im}^{+}, \Sigma_{im}) \tag{5.3}$$

The adaptation parameters $\mathbf{W}^{(s)}$ can be estimated by maximizing the likelihood of the adaptation data.

In addition to model parameters, this linear transform can also be applied to feature vectors. For the feature vector$\mathbf{o}_t$ at time step t, fMLLR can be described as

$$\mathbf{o}_t \rightarrow \mathbf{A}^{(s)} \mathbf{o}_t + \mathbf{b}^{(s)} \tag{5.4}$$

where $\mathbf{A}^{(s)}$ and $\mathbf{b}^{(s)}$ are the transformation parameters specific to speaker s. Similarly this can be rewritten into a more compact form

$$\mathbf{o}_t \rightarrow \mathbf{W}^{(s)} \mathbf{o}_t^{+} \tag{5.5}$$

where $\mathbf{o}_t^{+} = [\mathbf{o}_t^T, 1]^T$ is an expanded form of $\mathbf{o}_t$. The adaptation parameters $\mathbf{W}^{(s)}$ can be similarly estimated on the adaptation data via MLE.

SAT of GMM-HMMs

Another technique closely related with speaker adaptation is speaker adaptive training (SAT) [5, 6]. When carried out in the feature space, SAT performs fMLLR adaptation on the training set and projects training data into a speaker-normalized space. Parameters of the acoustic models are estimated in this new feature space. Acoustic models trained this way become independent of specific training speakers and thus generalize better to unseen testing speakers. In practice, SAT models generally give better recognition results than SI models, when speaker adaptation is applied to both of them. For GMM-HMM models, SAT has been a well-studied, long-standing technique. The idea of feature-space SAT for the GMM-HMM is demonstrated in Figure 5.1. The procedures of training and decoding of the SAT model are detailed as follows. Note that during decoding, we assume to perform unsupervised adaptation. In this case, two passes of decoding need to be conducted.

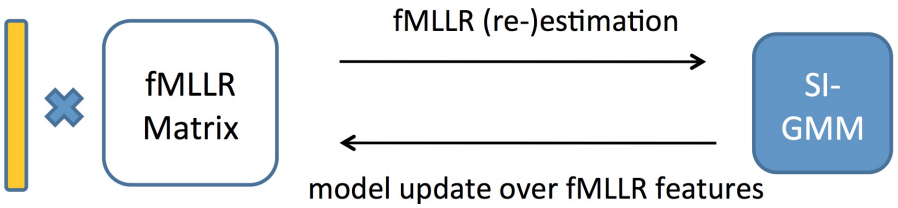


Figure 5.1: Illustration of SAT for GMM-HMMs.

Training

1. Step 1. Train the SI GMM-HMM acoustic model over all the training data.
2. Step 2. Estimate the fMLLR transforms for the training speakers, based on the SI model.
3. Step 3. Apply the fMLLR transforms to the original features so as to obtain the fMLLR feature space.
4. Step 4. Update the parameters of the GMM-HMM in the fMLLR feature space.
5. Repeat steps 2-4 iteratively until the training process converges.

Decoding

1. Step 1. Do a first pass of decoding using the SI model.
2. Step 2. Perform adaptation using the SAT model and the first-pass decoding outputs.
3. Step 3. Decode the SAT model in the adapted fMLLR feature space.

5.1.2 Speaker Adaptation and SAT of DNNs

Speaker adaptation acts as an effective procedure to reduce mismatch between training and testing conditions. Previous work has proposed a number of techniques for speaker adaptation of DNN models. These methods can be categorized into 3 classes. The first class augments the SI-DNN model with additional SD layers, and the parameters of these layers are learned on the target speakers. In [55, 101], a layer is inserted between the input features and the first hidden layer. This additional layer plays the similar role to feature-space maximum likelihood linear regression (fMLLR) transforms as estimated in GMMs. Alternatively, the SD layer can be inserted after the last hidden layer [55]. Yao *et al.* [121] demonstrate that transforming the outputs of the final hidden layer is equivalent to adapting the parameters of the softmax layer. After carrying out singular value decomposition (SVD) over DNN weight matrices, Xue *et al.* [119] perform

adaptation by learning SD matrices that are inserted between the decomposed weight matrices. In the recently proposed learning hidden unit contributions (LHUC) approach [111], a SD vector is attached to every hidden layer of the SI-DNN and learned on a particular testing speaker. These SD vectors are applied to the SI-DNN hidden outputs with element-wise multiplication, regulating the contributions of hidden units. This LHUC approach is further extended to achieve SAT for DNN models [110].

The second class of adaptation methods trains (and decodes) DNNs over SA features. For example, features transformed with vocal tract length normalization (VTLN) and fMLLR have been applied successfully as DNN features, showing notable advantage over non-adaptive features [92, 101]. Another way of normalizing speaker variability is to augment the input features with additional speaker-specific information. Speaker i-vectors [16, 17] have been explored for this purpose. For example, Saon *et al.* [99] append i-vectors to the original features at each speech frame. The effectiveness of incorporating i-vectors is further verified in [33, 103]. I-vector based adaptation is further developed in [48, 57] where separate vectors are used to represent finer-grained acoustic factors such as speakers, environments and noise. In [1, 2], the speaker representation (referred to as speaker codes) is learned on each speaker through network fine-tuning, instead of being extracted prior to DNN training.

The third class adapts the parameters of the SI-DNN model directly, without changing the architecture of the SI-DNN. For instance, [59] examines how the performance of DNN adaptation is affected by updating different parts (the input layer, the output layer, or the entire network) of the SI-DNN. Updating the entire DNN may suffer from overfitting, especially when the amount of adaptation data is limited. To address this issue, Yu *et al.* [126] propose to regularize speaker adaptation with the Kullback-Liebler (KL) divergence between the output distributions from the SI and the adapted DNNs. This regularizer prevents the parameters of the adapted DNN from deviating too much from the SI-DNN. Price *et al.* [90] add a softmax layer with context-independent (CI) states on top of the softmax layer with CD states. This helps alleviate the problem of rare CD classes having no or few examples on the adaptation data. In [104], instead of model parameters, the shape of the activation function used in the SI-DNN is adjusted to match the testing conditions.

In comparison to speaker adaptation, past work has made fewer attempts on SAT of DNNs. Training DNNs with SA features [92, 101, 114] or additional speaker-specific information [33, 99, 103] can be treated as a form of SAT. In [120], Xue *et al.* append speaker codes [1, 2] to the hidden and output layers of the DNN model. SAT is achieved by jointly learning the speaker-specific speaker codes and the SI-DNN. In [35], adaptive training is achieved by first constructing

a (shallow) neural network model for each speaker. The classification outputs from these speaker-dependent networks are fused using a Meta-Pi network. As an early effort to model speaker dependency in neural network based ASR, this approach is hard to scale up due to the building of the speaker-dependent components. In [84], speaker availability is normalized by allocating certain layers of the DNN as the SD layers that are learned on a speaker-specific basis. Over different speakers, the other layers are adaptively trained by picking the SD layer corresponding to the current speaker. Although showing promising results, the application of these proposals is constrained to specific feature types or model structures. For example, the approach in [120] is not applicable to CNNs because it is infeasible to append speaker codes to the hidden convolution layers. Also, in these methods, adaptation of the resulting SAT models generally needs multiple decoding passes, which undermines the decoding efficiency. This motivates us to propose a more general solution for SAT of DNNs. The framework we present in this chapter can be applied to different feature types and model structures. Furthermore, due to the incorporation of speaker i-vectors, speaker adaptation of the SAT models becomes both efficient and robust.

5.1.3 I-Vector Extraction

Recently, the application of the i-vector paradigm has resulted in significant advancement in the speaker verification community [16, 17]. The i-vector approach differs from the earlier JFA [50] in that it has a single variability subspace to model different types of variability emerging from the speech signal. We assume to have a GMM model, referred to as universal background model (UBM), which consists of K Gaussian components. The *t*-th frame $\mathbf{o}_t$ from the *s*-th speech segment is formulated to be generated from the UBM model as follows:

$$\mathbf{o}_t \sim \sum_{k=1}^{K} r_t(k) N(\boldsymbol{\mu}_k + \mathbf{T}_k \mathbf{w}_s, \boldsymbol{\Sigma}_k) \tag{5.6}$$

where $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean vector and covariance matrix of the *k*-th Gaussian, the total variability matrices $\mathbf{T}_k$ span a subspace for the shifts by which the UBM means are adapted to particular segments, $r_t(k)$ represents the posterior probability of the *k*-th Gaussian given the speech frame. The latent vector $\mathbf{w}_s$ follows a standard normal distribution and describes the coordinates of the mean shift in the total variability subspace. The maximum a posterior (MAP) point estimate of the vector $\mathbf{w}_s$, called an i-vector, represents salient information about all types of variability in the speech segment. The readers can refer to [16, 17] for more details.

Given a collection of speech segments, the UBM model can be trained using the standard

MLE. To build the i-vector model, sufficient statistics are accumulated on each speech frame based on the posteriors with respect to the UBM. These statistics are used to learn the total variability matrices and extract the i-vectors. After the i-vectors are obtained, a scoring model, e.g., probabilistic linear discriminant analysis (PLDA), can be applied to the i-vectors for speaker recognition/verification. When extracted at the speaker (rather than segment) level, i-vectors provide a low-dimensional representation of the acoustic characteristics of individual speakers. Previous work [33, 47, 99, 103] has applied i-vectors successfully to speaker adaptation of both GMM and DNN acoustic models. In this thesis, we leverage i-vectors to perform adaptive training of DNNs. It is worth noting that although called SAT, adaptive training performed in this work also pertains to acoustic conditions because i-vectors encapsulate information regarding speakers and conditions (noise, channel, etc).

5.2 Speaker Adaptive Training of DNNs

This section introduces our SAT-DNN framework. We first present the overall architecture and then give details about the two training steps: training the adaptation network and updating the DNN parameters. Finally, we elaborate on how to decode the SAT-DNN models.

5.2.1 Architecture of SAT-DNNs

For GMM-based acoustic modeling, speaker-specific fMLLR transforms are estimated on the training speakers if SAT is performed in the feature-space. The GMM parameters are then updated in the fMLLR-transformed feature space. We port this idea to DNN models, and the whole SAT-DNN architecture is shown in Figure 5.2. Suppose that an i-vector has been extracted for each speaker as described in Section 5.1.3 where the segment now contains all the frames from the speaker. As with SAT of GMMs, SAT of DNNs starts from a SI-DNN model (on the right of Figure 5.2) that has been well trained over the entire training set. Two steps are then taken to build the SAT-DNN model. First, with the SI-DNN fixed, a smaller adaptation neural network (on the left of Figure 5.2) is learned to take i-vectors as inputs and generate speaker-specific linear shifts to the original DNN feature vectors. In Figure 5.2, the t-th input vector $\mathbf{o}_t$ comes from speaker s whose i-vector is denoted as $\mathbf{i}_s$. The layers of the adaptation network are indexed by -I, -(I-1), ..., -2, -1, where I is the total number of hidden layers in the adaptation network. The

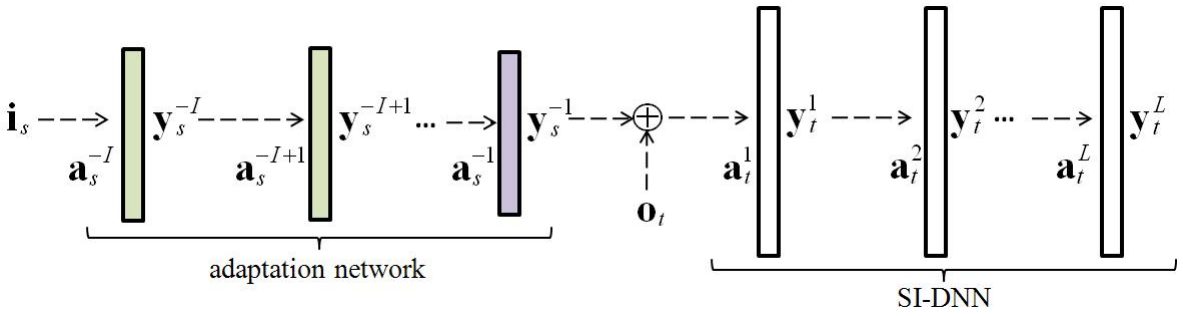


Figure 5.2: Architecture of the SAT-DNN model.

values attached to the adaptation network can be computed as:

$$\mathbf{a}_s^i = \mathbf{W}_i \mathbf{x}_s^i + \mathbf{b}_i \qquad \mathbf{y}_s^i = \phi(\mathbf{a}_s^i) \qquad -I \leq i \leq -1 \tag{5.7}$$

where ϕ is the activation function of each layer in the adaptation network, $\mathbf{x}_s^i$ is the inputs to the *i*-th layer and can be represented as:

$$\mathbf{x}_s^i = \begin{cases} \mathbf{i}_s & i = -I \\ \mathbf{y}_s^{i-1} & -I < i \leq -1 \end{cases} \tag{5.8}$$

After the adaptation network is trained, the speaker-specific output vector $\mathbf{y}_s^{-1}$ is added to each of the original feature vector $\mathbf{o}_t$ from speaker *s*:

$$\mathbf{z}_t = \mathbf{o}_t \oplus \mathbf{y}_s^{-1} \tag{5.9}$$

where $\oplus$ represents element-wise addition. This gives us a new feature space which is expected to be more speaker normalized.

The second step involves updating the parameters of the DNN model in the new feature space. We keep the adaptation network unchanged, and the SI-DNN is fine-tuned by taking $\mathbf{z}_t$ as the new feature vector at each frame *t*. This finally generates the SAT-DNN model that is more independent of specific speakers. Note that during this updating step, the DNN parameters use the original SI-DNN as the initial values, instead of being learned from scratch.

The formulation of the adaptation network outputs as linear feature shifts imposes two constraints. First, the output layer of the adaptation network has to use the identity activation function $\phi(x) = x$ by which we are not restricting the direction and value range of the feature shifts. Second, the number of units at the output layer (i.e., the dimension of $\mathbf{y}_s^{-1}$) has to equal the di-

mension of the original feature vector $\mathbf{o}_t$. Both training of the adaptation network and updating of the DNN model can be realized with the standard error back-propagation. We will give more details in the following two subsections.

5.2.2 Training of the Adaptation Networks

The adaptation network is learned in a supervised manner, with errors back-propagated from the SI-DNN. The objective function remains to be Equation 3.2, i.e., the cross-entropy computed with the outputs from the DNN softmax layer and the ground-truth labels. However, the inputs of the DNN are now the new input features $\mathbf{z}_t$, instead of the original feature $\mathbf{o}_t$. Our goal is to optimize this objective with respect to the parameters of the adaptation network. The derivatives of the objective function with respect to the new feature vector $\mathbf{z}_t$ can be written as:

$$\frac{\partial \mathbb{L}}{\partial \mathbf{z}_t} = \mathbf{W}_1^T \boldsymbol{\epsilon}_t^1 \tag{5.10}$$

where $\boldsymbol{\epsilon}_t^1$ represents the error vector back-propagated to the first hidden layer of the DNN, $\mathbf{W}_1$ is the weight matrix connecting the input features and the first hidden layer of the DNN. At the output layer of the adaptation network, we have $\mathbf{y}_s^{-1} = \mathbf{a}_s^{-1}$ because of the identity activation function. This output layer has the error vector:

$$\boldsymbol{\epsilon}_s^{-1} = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_s^{-1}} = \frac{\partial \mathbb{L}}{\partial \mathbf{y}_s^{-1}} = \frac{\partial \mathbb{L}}{\partial \mathbf{z}_t} \tag{5.11}$$

which is further back-propagated into the adaptation network. The parameters of the adaptation network are updated with gradients derived from the error vectors. We can see that these gradients depend not only on the speaker i-vectors but also on the DNN input features $\mathbf{o}_t$. Therefore, the training data of the adaptation network consist of the combination of i-vectors and their corresponding speech frames. The number of training examples equals the number of speech frames, rather than the number of speakers. Although we also get the gradients of the DNN parameters, the DNN parameters are not updated.

5.2.3 Updating of the DNN Model

After the adaptation network is trained, updating of the DNN model is straightforward to accomplish. Parameters of the DNN are initialized with parameters of the SI-DNN model and fine-tuned with the cross-entropy objective. The only difference is that the inputs of the DNN

are now the speaker-normalized features $\mathbf{z}_t$. Parameters of the adaptation network are kept fixed during this step. When fine-tuning terminates, we get the final SAT-DNN model.

From its training process, we can see that our SAT-DNN approach poses a general framework. It does not depend on specific choices of the original DNN input features. Although called SI-DNN, the initial DNN model can be trained on either SI features (e.g., filterbanks, MFCCs, etc) or speaker-adapted features (e.g., fMLLRs, filterbanks with VTLN, etc). Moreover, in addition to DNNs, the approach can be applied naturally to other types of deep learning models such as CNNs [3, 4, 95, 96, 107] and RNNs [30, 62, 97, 98]. In our experiments, we will demonstrate the extension of SAT-DNN to various model and feature types.

5.2.4 Decoding of SAT-DNN

Decoding of SAT models generally requires speaker adaptation on the testing data. For SAT-DNN models, speaker adaptation simply involves extracting the i-vector for each testing speaker and feeding the i-vector into the adaptation network. This produces the linear feature shift specific to this speaker. Adding the feature shift to the original feature vectors generates a speaker-adapted feature space, in which the SAT-DNN model is decoded. We summarize the major steps for training and decoding of SAT-DNN models as follows:

Training

1. Train the SI-DNN acoustic model over all the training data.
2. Re-align the training data with the SI-DNN model. The alignment *align-dnn* serves as new targets.
3. Extract the i-vector $\mathbf{i}_s$ for each training speaker.
4. Fix the parameters of the SI-DNN. Learn the adaptation network with the input features $\mathbf{o}_t$, i-vectors $\mathbf{i}_s$ and the supervision *align-dnn*.
5. Fix the parameters of the adaptation network. Update the parameters of the DNN model with the new features $\mathbf{z}_t$ and the supervision *align-dnn*.

Decoding

1. Extract the i-vector of each testing speaker.
2. Feed the i-vector to the adaptation network. Produce the speaker-specific feature shifts.
3. Decode the SAT-DNN model in the speaker-adapted feature space $\mathbf{z}_t$.

5.3 Experiments

The proposed approach is evaluated on a LVCSR task of transcribing TED talks. We show our experimental setup that is followed by results and analysis.

5.3.1 Experimental Setup

Dataset

Our experiments use the benchmark TEDLIUM dataset [93] which was released to advance ASR on TED talks. This publicly available dataset contains 774 TED talks that amount to 118 hours of speech data. We take this dataset as our training set and each TED talk is treated as a speaker. Decoding is done on the dev2010 and tst2010 test sets defined by the ASR track of the previous IWSLT evaluation campaigns. The ASR task of IWSLT aims at recognition of TED talks which acts as a critical component in the end-to-end speech translation systems. For comprehensive evaluation, we merge the two sets into a single test set which contains 19 TED talks, i.e., 4 hours of speech.

GMM Models

We first train the initial MLE model using 39-dimensional MFCCs+$\triangle$+$\triangle\triangle$. Then 7 frames of MFCCs are spliced together and projected down to 40 dimensions with LDA. A maximum likelihood linear transform (MLLT) is applied on the LDA features and generates the LDA+MLLT model. Discriminative training with the boosted maximum mutual information (BMMI) objective [86] is finally performed on top of the LDA+MLLT system. The GMM model has 3980 clustered triphone states and an average of 20 Gaussian components per state.

DNN Baseline

We build the DNN model using our PDNN implementation [67]. The class label on each speech frame is generated by the GMM model through forced alignment. We use a DNN with 6 hidden layers, each of which has 1024 units and the logistic sigmoid activation function. The last softmax layer has 3980 units corresponding to the states. The inputs are 11 neighboring frames of 40-dimensional log-scale filterbank coefficients with per-speaker mean and variance normalization. The DNN parameters are initialized with the SdAs.

Table 5.1: WERs(%) of the SI-DNN and SAT-DNN models.

Model	Alignment	WER(%)	Rel. Imp. (%)
BMMI GMM	—	24.1	—
SI-DNN	from GMM	20.0	—
SAT-DNN	from GMM	18.1	9.5
SAT-DNN	from DNN	17.7	11.5

For fine-tuning, we optimize the cross-entropy objective using the exponentially decaying newbob learning rate schedule. Specifically, the learning rate starts from a big value 0.08 and remains unchanged until the increase of the frame accuracy on a cross-validation set between two consecutive epochs falls below 0.2%. Then the learning rate is decayed by a factor of 0.5 at each of the subsequent epochs. The whole learning process terminates when the frame accuracy fails to improve by 0.2% between two successive epochs. We use the mini-batch size of 256 for SGD, and a momentum of 0.5 for gradient smoothing. During decoding, the original Kaldi tedlium recipe relies on a generic CMU language model. In our setup, we train a lightweight trigram language model using 180k sentences of TED talk transcripts. This in-domain language model is pruned aggressively for decoding efficiency.

SAT-DNN Baseline

The DNN model which has been trained serves as the SI-DNN for constructing the SAT-DNN model. The adaptation network contains 3 hidden layers each of which has 512 units and uses the logistic sigmoid activation function. The output layer has 440 (the dimension of the DNN input features) units and uses the identify activation function. Parameters of the adaptation network are randomly initialized. Training of the adaptation network and updating of the DNN follow the same fine-tuning setting as training of the SI-DNN.

I-vector extraction is conducted with Kaldi's in-built i-vector functionality. The i-vector extractor takes 19-dimensional MFCCs and log-energy as the features. Computing deltas and accelerations finally gives a 60-dimensional feature vector on each frame. Both the UBM model and the total variability matrices are trained on the entire training set. For each training and testing speaker, we extract a 100-dimensional i-vector which has been found to give the optimal recognition performance in our previous studies [75, 77].

5.3.2 Basic Results

Table 5.1 compares the WERs of the SI-DNN and SAT-DNN models on the test set. The last column shows the relative improvement of SAT-DNNs over SI-DNN. The SI-DNN model gets a WER of 20.0% which is significantly better than the discriminatively trained GMM model. When training the SAT-DNN model, we can employ the frame-level alignment generated from the GMM model, i.e., the same alignment as used by the SI-DNN training. In this case, Table 5.1 shows that the SAT-DNN has a WER of 18.1%, that is 9.5% relative improvement over the SI-DNN. Alternatively, we can re-align the training data with the SI-DNN and use the newly-generated alignment during SAT-DNN training (both learning of the adaptation network and updating of the DNN). This re-alignment step improves the WER of the SAT-DNN further to 17.7%. Also, we observe that SAT-DNN training with the new DNN-generated alignment converges faster than with the old GMM-generated alignment. Thus, unless otherwise stated, training of the SAT-DNN models always uses the new alignment in the rest of our experiments.

5.3.3 Bridging I-vector Extraction with DNN Training

I-vector extraction in Section 5.1.3 aims to optimize the objective of speaker modeling. The UBM model and the total variability matrices are trained with MLE. In contrast, DNN and SAT-DNN models try to distinguish phonetic classes and are trained in a discriminative manner. The separate training of these two parts with respect to different objectives may hurt the performance of the SAT-DNN models. Past work [54] has studied approaches to leveraging DNN models in i-vector extraction. In [54], the UBM used in i-vector extraction is replaced with a DNN that has been trained for acoustic modeling. In this case, the probabilities $r_t(k)$ are posteriors of states outputted by the DNN, instead of posteriors of the Gaussians from the UBM. This manner of incorporating DNNs into i-vector extraction results in significant improvement on a benchmark speaker recognition task [54].

In this subsection, we bridge i-vector extraction with DNN training from the front-end perspective. Prior to building the i-vector extractor, we learn a DNN model that is designed for BNF generation. Instead of MFCCs, outputs from the bottleneck layer of such a BNF-DNN are taken as the features for training the i-vector extractor (including the UBM and total variability matrices). Only changing the front-end enables us to take advantage of the existing i-vector extraction pipeline, without making major modifications. The incorporation of the DNN-based features adds phonetic discrimination ability to the extracted i-vectors, which potentially benefits the following SAT-DNN training.

Table 5.2: WERs(%) of SAT-DNN models with i-vectors from MFCC and BNF feature respectively.

Model	Features for I-vectors	WER(%)	Rel. Imp. (%)
SAT-DNN	MFCCs	17.7	11.5
SAT-DNN	BNFs	17.3	13.5

Following our DBNF architecture [24, 25], the BNF-DNN has 6 hidden layers in which the 5th layer is a bottleneck layer. To be consistent with MFCCs, the bottleneck layer has 60 nodes whereas each of the other hidden layers has 1024 nodes. Inputs to the BNF-DNN are 11 neighboring frames of 40-dimensional filterbank features. In Table 5.2, we show the results of SAT-DNNs using i-vectors extracted from MFCCs and BNFs respectively. The last column shows the relative improvement of SAT-DNNs over the SI-DNN baseline. Within the SAT-DNN framework, BNF-based i-vectors result in slight improvement (0.4% absolute) over MFCC-based i-vectors. Applying the BNF-based i-vectors brings the WER of the SAT-DNN model down to 17.3% that is 13.5% relative improvement compared with the SI-DNN baseline.

5.3.4 Variable Data Amount for I-vector Extraction

In the thesis, our experiments use all the data to estimate i-vectors for adaptation, i.e., on average 780 seconds of speech data per speaker. In certain scenarios (e.g., online decoding), however, the adaptation data may become highly limited. We examine how the amount of adaptation data affects the recognition performance of SAT-DNNs. For convenience of formulation, we use the variable β to denote the average amount of adaptation data per speaker in terms of seconds. By tuning the maximum number of frames one speaker can have, we systematically reduce β from 780 to 12, by a factor of 2 each time. Note that we only vary the data amount for adaptation (i.e., i-vector estimation), keeping the data to be eventually decoded unchanged. Fig. 5.3 depicts the variation of WERs with different values of β. Reducing β up to 25 leads to either zero or minor WER degradation. We start to observe notable degradation when β equals 12. This suggests that SAT-DNN performs robustly to reduced amount of adaptation data, and thus is applicable to online decoding.

5.3.5 Application to fMLLR Features

So far, we have looked at the filterbank features as the DNN inputs. This subsection examines how the SAT-DNN model works when the DNN inputs are speaker-adapted fMLLR features.

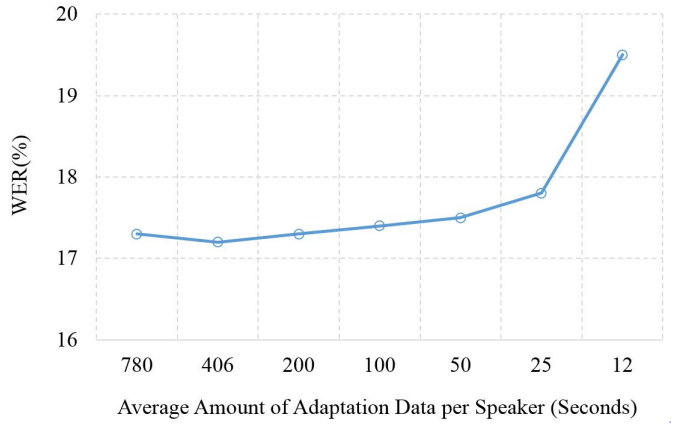


Figure 5.3: The WERs of SAT-DNN models when β is reduced from 780 to 12.

Table 5.3: WERs(%) of the DNN and SAT-DNN when the inptus are fMLLR features.

Model	WER(%)	Rel. Imp. (%)
DNN (baseline)	18.9	—
SAT-DNN	17.4	7.9

We perform SAT over the LDA+MLLT GMM model, which produces the SAT-GMM model and fMLLR transforms of the training speakers. DNN inputs include 11 neighboring frames of 40-dimensional fMLLR features. Training of the DNN with fMLLRs follows the same protocol as training of the DNN with filterbanks, using alignment generated by the SAT-GMM model. During SAT-DNN training, the frame-level class labels come from re-alignment with the new fMLLR-based DNN model, and the i-vectors from the BNF features. Table 5.3 shows the performance of the resulting SAT-DNN models. The last column shows the relative improvement of SAT-DNNs over the DNN baseline. Compared with the DNN model, the SAT-DNN obtains 1.5% absolute improvement (17.4% vs 18.9%) in terms of WER, which is equivalent to 7.9% relatively. Because speaker availability has been partly modeled by fMLLR transforms, the gains achieved by SAT-DNN here are less significant than the gains on filterbank features.

5.4 Extension to BNFs and CNNs

In Section 5.2, we argue that our SAT approach is a general framework. To demonstrate this, we empirically study two natural extensions of SAT-DNNs. All the experiments are based on the same setup as in the previous section.

Table 5.4: WERs(%) of BMMI GMM models when the features are MFCCs, DBNF and SAT-DBNF.

Front-end	WER(%)	Rel. Imp. (%)
MFCCs	24.1	—
DBNF	17.7	—
SAT-DBNF	16.2	8.5

5.4.1 Extension to BNFs

We have studied the application of SAT-DNNs as hybrid models. In tandem systems, DNNs can also be used as discriminative feature extractors. A widely employed manner is to place a bottleneck layer in the DNN architecture. Outputs from the bottleneck layer are taken as new features on top of which GMM models are further built. In this work, we turn to the DBNF [24, 25] approach for bottleneck feature extraction. DBNF is characterized by the asymmetric arrangement of the layers and multiple hidden layers before the bottleneck layer. The DBNF network has 6 hidden layers in which the 5th hidden layer is a bottleneck layer. This bottleneck layer has 42 units, whereas each of the other hidden layers has 1024 nodes. The parameters of the 4 prior-to-bottleneck layers are initialized with SdA pre-training [115]. Inputs to the DBNF network are 11 neighbouring frames of 40-dimensional filterbank features. After this network is trained, we build a LDA+MLLT GMM model following the standard Kaldi pipeline [88]. Specifically, at each frame, the 42-dimensional bottleneck features are further normalized with mean subtraction. Then, 7 consecutive BNF frames are spliced and then projected to 40 dimensions using LDA. On top of the LDA+MLLT model, 4 iterations of discriminative training is performed with the BMMI objective [86].

Applying SAT to bottleneck feature extraction is straightforward. We simply replace the DNN model in Figure 5.2 with the DBNF network. When training is finished, we obtain the features $\mathbf{z}_t$ with the adaptation network and i-vectors. Speaker-adapted bottleneck features are generated by feeding $\mathbf{z}_t$ to the SAT-DBNF network. Table 5.4 compares the performance of the final BMMI GMM models when the bottleneck features are extracted from the DBNF and SAT-DBNF networks respectively. The last column shows the relative improvement of SAT-DBNF over the standard DBNF. We observe that the GMM model with bottleneck features largely outperforms (17.7% vs 24.1%) the GMM model with MFCCs. Extracting bottleneck features from SAT-DBNF further reduces the WER to 16.2%, i.e., 32.8% and 8.5% relative improvement over the MFCC and the standard DBNF front-end respectively. This demonstrates the effectiveness of the SAT technique in improving the quality of bottleneck features.

Table 5.5: Configurations (filter and pooling size) of the two convolution layers in our CNN architecture.

	#1 conv layer		#2 conv layer	
	frequency	time	frequency	time
Filter	9	9	4	3
Pooling	3	1	1	1

5.4.2 Extension to CNNs

Section 5.2 claims the applicability of our SAT technique to CNNs, and we experimentally confirm this in this subsection. Our CNN architecture follows [107], consisting of 2 convolution and 4 fully-connected layers. We apply 2-dimensional convolution over both time and frequency. The CNN inputs are 40-dimensional filterbank features with their $\triangle$ and $\triangle\triangle$ features, with a temporal context of 11 frames. The configuration of the CNN layers is shown in Table 5.5. The first convolution layer filters the inputs using 256x3 kernels each of which has the size of 9x9. This is followed by a max-pooling layer only along the frequency axis, with the pooling size of 3. The second convolution layer takes as inputs the outputs from the pooling layer and filters them with 256x256 kernels with the size of 4x3. All the convolution operations are overlapping with the stride of 1 whereas the pooling is always non-overlapping. Outputs from the convolution layers are 256 feature maps, each of which has the size of 8x1. We then place 4 fully-connected hidden layers and finally the softmax layer on top of the convolution layers. Both the convolution and the fully-connected layers use the logistic sigmoid activation function. Detailed configurations of the convolution layers are listed in Table 5.5.

As with SAT-DNNs, training of the SAT-CNN model starts from a well-trained SI-CNN model. The i-vectors are extracted with bottleneck features as described in Section 5.3.3. We learn the adaptation network that has the output dimension of 3x11x40. The features shifts are added to the original features and the CNN model is updated in the new feature space. Table 5.6 presents the performance of the SI-CNN and SAT-CNN models, with the last column showing the relative improvement of the SAT-CNN over the SI-CNN. We can see that our SI-CNN model outperforms both the SI-DNN model with filterbanks and the DNN model with fMLLRs, revealing that the SI-CNN poses a strong baseline. In comparison with this strong baseline, the SAT-CNN model truly gives a better WER 16.8%. The 7.2% relative improvement achieved by SAT-CNN over SI-CNN is less than the improvement achieved by SAT-DNN over SI-DNN. This is partly because CNNs normalize speech features more effectively than DNNs, which decreases the efficacy of the application of SAT.

Table 5.6: WERs(%) of the SI-CNN and SAT-CNN models.

Model	WER(%)	Rel. Imp. (%)
SI-CNN (baseline)	18.1	—
SAT-CNN	16.8	7.2

5.5 SAT and Speaker Adaptation

In this section, we focus on the comparisons and combinations of SAT and speaker adaptation for DNN models. The performance of SAT is firstly compared against two competitive speaker adaptation methods. Then, we show that model-space adaptation can further improve the recognition accuracy of SAT-DNNs.

5.5.1 Comparing SAT and Speaker Adaptation

Past work [99] closely related with this chapter performs speaker adaptation of DNNs by incorporating i-vectors. Specifically, at each frame t, the original DNN input vector $\mathbf{o}_t$ is concatenated with the corresponding speaker i-vector $\mathbf{i}_s$. The combined feature vector [$\mathbf{o}_t$, $\mathbf{i}_s$] is treated as the new DNN inputs, in both training and testing. For convenience of formulation, this method is referred to as DNN+I-vector. Recently a model-space adaptation method, Learning Hidden Unit Contributions (LHUC), was proposed in [111]. In this method, the SI-DNN model is adapted to a specific speaker s with a SD parameter vector $\mathbf{r}_s^i$ at each hidden layer. The outputs of the i-th hidden layer in the SD model now become:

$$\mathbf{y}_t^i = \psi(\mathbf{r}_s^i) \odot \sigma(\mathbf{W}_i \mathbf{x}_t^i + \mathbf{b}_i) \tag{5.12}$$

where $\odot$ represents element-wise multiplication, ψ is a function to constrain the range of the parameter vector and is set to $\psi(x) = 2/(1 + exp(-x))$ in [111]. During adaptation, only the SD parameters [$\mathbf{r}_s^i, 1 \leq i < I$] are estimated on the adaptation data with error back-propagation, whereas the SI-DNN parameters remain unchanged.

Our implementation of these two methods follows [99] and [111]. For DNN+I-vector, a 100-dimensional i-vector is extracted for each training or testing speaker. All the other settings are consistent with the settings of the SI-DNN model in Section 5.3. For LHUC, the elements of $\mathbf{r}_s^i$ are initialized uniformly to 0. A first pass of decoding with the SI-DNN is done to get the frame-level supervision. Training of the SD parameters goes through 3 epochs with the constant learning rate of 0.8. Table 5.7 shows the results of the DNN models adapted with

Table 5.7: Performance comparisons between SAT-DNN and speaker adaptation methods.

Model/Adaptation Methods	WER(%)	Rel. Imp. (%)
SI-DNN (baseline)	20.0	—
SAT-DNN	17.3	13.5
DNN+I-vector[99]	19.5	2.5
DNN+LHUC[111]	18.3	8.5

Table 5.8: Performance of adapted DNNs using LHUC and different sets of first-pass hypotheses.

1stPass System	WER(%) of Hypotheses	WER(%) of Adapted DNN
SI-GMM	28.1	19.3
SAT-GMM	23.3	18.4
SI-DNN	20.0	18.3

these two methods. The last column shows the relative improvement over the SI-DNN baseline. We observe that compared with the un-adapted SI-DNN, DNN+I-vector brings 2.5% relative improvement and DNN+LHUC gives 8.5%. However, both methods are outperformed by the SAT-DNN model, which justifies the necessity of conducting complete SAT for DNN models.

For more complete comparison, we experiment with LHUC when different sets of supervision hypotheses are used. In particular, we perform decoding with the SI-GMM (LDA+MLLT), SAT-GMM and SI-DNN models respectively, and employ the resulting hypotheses as supervision for LHUC-based adaptation. As shown in Table 5.8, these models have the WERs of 28.1%, 23.3% and 20.0%, simulating first-pass hypotheses of different quality levels. We observe that when using LHUC, the performance of the adapted DNN gets worse as the WER of the supervision hypotheses deteriorates. This degradation results from fine-tuning on the adaptation data where the erroneous supervision is taken as true class labels. On the contrary, the adaptation of SAT-DNN requires no fine-tuning on the adaptation data and therefore its performance becomes robust to supervision errors. It is more suitable for unsupervised adaptation, especially when the first-pass decoding has high WERs.

5.5.2 Combining SAT and Model-space Adaptation

The adaptation of the SAT-DNN model is in fact feature-space adaptation because it normalizes the DNN inputs. After getting the speaker-adapted features ($\mathbf{z}_t$ in Equation (8)), the SAT-DNN model can be further adapted in this new feature space, using any of the model-space adaptation methods. We turn to LHUC for model-space adaptation. Table 5.9 provides a summary of the

Table 5.9: A summary of the performance of SAT-DNNs using i-vectors extracted from MFCCs and BNFs respectively.

Model	WER(%)	Rel. Impr.(%)
SI-DNN	20.0	—
I-vectors from MFCCs		
SAT-DNN	17.7	11.5
+LHUC	16.7	16.5
I-vectors from BNFs		
SAT-DNN	17.3	13.5
+LHUC	16.5	17.5

complete results of the SI-DNN and SAT-DNN models when the input features are filterbanks. In this table, "+LHUC" means that the LHUC-based adaptation is applied over SAT-DNNs. The last column shows the relative improvement over the SI-DNN. Applying LHUC on top of SAT-DNN produces nice gains in terms of WERs. Combining SAT-DNN and LHUC together finally gives us a WER of 16.5% on the test set, which is 17.5% relative improvement compared with the SI-DNN model.

5.6 Acceleration of SAT-DNN training

A major concern for building SAT-DNN models is the additional cost it imposes to the training pipeline. We investigate a simple but effective data reduction strategy to speed up the training of SAT-DNNs. The intuition is that the adaptation network models mapping from the i-vectors to the feature shifts. As a result, during training of the adaptation network, the number of distinct i-vectors plays a more important role than the number of speech frames. Moreover, because of taking the SI-DNN model for initialization, updating of the DNN supposedly needs less data compared with training from scratch. Based on these intuitions, we select one example out of every n speech frames for training of SAT-DNNs. That is, after picking a training example, we skip the following $n - 1$ frames to pick the next one. By doing this, we keep the number of training speakers (i-vectors) unchanged. However, the training set is shrunk to $1/n$ of its original size, which speeds up the SAT-DNN training by n times.

In Table 5.10, we present the results of the resulting SAT-DNN models when n ranges from 1 to 4. The numbers are obtained on the LVCSR task of transcribing TED talks, as described in the thesis. The data size is measured by the ratio of the reduced size to the original size. With n = 2, we accelerate the training process by $2\times$ while getting 1.7% relative WER degradation (17.6%

Table 5.10: Performance of SAT-DNN models when the training set is shrunk by frame skipping ("Frame") and speaker reduction ("Speaker") respectively. The data size is measured by the ratio of the reduced size to the original size.

Data size	Speed up	WER(%) Frame	WER(%) Speaker
1	1×	17.3	
1/2	2×	17.6	18.5

vs 17.3%). A direct comparison is to shrink the training set by reducing the number of speakers, without skipping any speech frames. As shown in Table 5.10, the results of SAT-DNNs using this speaker reduction strategy become notably worse, e.g., 7.0% relative degradation (18.5% vs 17.3%) when n is set to 2. This reveals that data reduction based on frame skipping is an effective strategy to speed up SAT-DNN training without causing significant loss on WERs.

5.7 Application to the Switchboard Dataset

We finally evaluate our SAT-DNN framework on the Switchboard conversational telephone transcription task. We use Switchboard-1 Release 2 (LDC97S62) as the training set which contains over 300 hours of speech. For fast turnarounds, we also select 110 hours from the training set and create a lighter setup. Our test set is the Hub500 (LDC2002S09) set which consists of 20 conversations from Switchboard and 20 conversations from CallHome English. To be consistent with previous work [99, 101], we report results only on the Switchboard part. For decoding, a trigram language model (LM) is trained on the training transcripts. This LM is then interpolated with another trigram LM trained on the Fisher English Part1 transcripts (LDC2004T19).

5.7.1 Experiments on the 110-Hour Setup

We first report experiments on the 110-hour setup. The HMM/GMM systems are built with the standard Kaldi Switchboard recipe [88]. We train the initial ML model based on 39-dimensional MFCCs (plus deltas and double deltas) features with per-speaker mean normalization. Then 7 frames of MFCCs are spliced and projected to 40 dimensions with linear discriminant analysis (LDA). A MLLT is estimated on the LDA features and generates the LDA+MLLT model. Over the LDA+MLLT model, SAT is performed with one fMLLR transform per speaker. The final SAT-GMM model has 4287 CD states (senones).

The baseline DNN model has 5 hidden layers each of which contains 1200 neurons. The parameters of this DNN model are randomly initialized. DNN fine-tuning optimizes the cross-

Table 5.11: Comparisons of DNN and SAT-DNN on the Switchboard corpus. WERs are reported on the Switchboard part of the Hub5'00 (eval2000) testing set.

Model	WER(%)	Rel. Impr.
DNN (baseline)	21.7	—
DNN+I-vector[99]	21.1	2.8%
SAT-DNN	19.3	11.1%

entropy (CE) objective using an exponentially decaying newbob learning rate schedule. Specifically, the learning rate starts from 0.08 and remains unchanged until the increase of the frame accuracy on a cross-validation set between two consecutive epochs falls below 0.2%. Then the learning rate is decayed by a factor of 0.5 at each of the subsequent epochs. The whole learning process terminates when the frame accuracy fails to improve by 0.2% between two successive epochs. A mini-batch size of 256 is adopted for stochastic gradient descent (SGD).

This baseline DNN is taken as the initial model during building of SAT-DNN models. When we apply the SAT-DNN framework, the adaptation network contains 3 hidden layers, each of which has 512 hidden units and uses the logistic sigmoid activation function. The output layer has 440 (the dimension of the DNN input features) units and uses the identify activation function. Parameters of the adaptation network are randomly initialized. Moreover, following the configurations in Section 5.3.1, i-vector extraction generates a 100-dimensional i-vector for each training and testing speaker.

Our results are shown in Figure 5.11. The baseline DNN achieves the WER of 21.7%, while the SAT-DNN model gives the WER of 19.3%. That is, the SAT-DNN gets 11.1% relative improvement over the baseline DNN. In comparison, the DNN+I-vector approach, which simply concatenates i-vectors with the original acoustic features, is also applied, and results in the WER of 21.1%. This translates to only 2.8% relative improvement over the baseline DNN, performing worse than our SAT-DNN approach.

5.7.2 Experiments on the Complete 300-Hour Setup

Our SAT-DNN approach is finally evaluated with the complete 300 hours of training set. We follow the same procedures as described in Section 5.7.1 to build the GMM and DNN models. The number of CD states in the GMM model increases from 4287 to 8929. The DNN model has 6 hidden layers each of which contains 2048 neurons. The DNN is initialized with SdAs that are pretrained in a greedy layerwise fashion. When we apply the SAT-DNN framework, the adaptation network contains 4 hidden layers, each of which has 1024 hidden units. The

Table 5.12: Comparisons of DNN and SAT-DNN on the Switchboard corpus. WERs are reported on the Switchboard part of the Hub5'00 (eval2000) testing set.

Model	WER(%)	Rel. Impr.
DNN (baseline)	16.6	—
DNN+I-vector[99]	15.2	8.4%
SAT-DNN	14.6	12.0%

experimental results are shown in Table 5.12.

With the complete training set, the baseline DNN achieves the WER of 16.6%, which is comparable to numbers reported in previous work [71]. Applying SAT-DNN reduces the WER to 14.6%, i.e., 12.0% relative improvement compared to the baseline DNN. This improvement reveals that our SAT-DNN framework performs effectively on larger datasets such as Switchboard.

We also build the DNN+I-vector model on this setup. DNN+I-vector finally obtains the WER of 15.2%, i.e., 8.4% relative improvement over the baseline DNN. A comparison between the 300-hour and 110-hour results indicates that the DNN+I-vector approach gives noticeable gains under large amounts of training data. These gains are largely diminished when the amount of training data decreases. This is partly because DNN+I-vector employs a simple linear concatenation to incorporate the speaker i-vectors. Since the power of the linear concatenation is relatively limited, the mapping from speaker i-vectors to SA feature space can only be achieved when large amounts of training speech (i.e., large numbers of training speakers) become available. On the contrary, our SAT-DNN framework relies on the adaptation network to serve as the mapping function, and thus adds great flexibility to the learning of the complex mapping. Because of this, our SAT-DNN approach achieves consistent relative improvement across different levels of data availability.

To further prove the advantage of SAT-DNN under limited training data, we create two other training sets: 64 hours and 200 hours. In Figure 5.4, we plot the WERs achieved by various models across different amounts of training speech. It can be clearly seen that with smaller training sets, DNN+I-vector, in which we concatenate i-vectors with acoustic features, achieves only minor improvement over the baseline DNN model. This improvement becomes more significant as the amount of training speech increases. On the contrary, the SAT-DNN model shows consistent gains across different levels of data availability, compared to the baseline DNN.

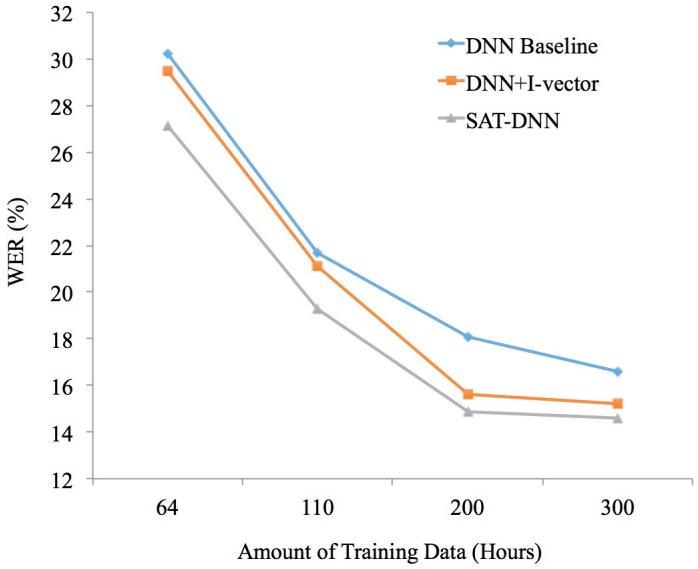


Figure 5.4: The WERs of the baseline DNN, DNN+I-vector and SAT-DNN models when different amounts of training speech are available.

5.8 Summary

In this chapter, we have proposed a framework to perform SAT for DNN acoustic models. The SAT approach relies on i-vectors and the adaptation neural network to realize feature normalization. This work explores the application of SAT-DNNs to LVCSR tasks. We determine the optimal configurations (e.g., features for i-vector extraction) for building of SAT-DNN models. The SAT approach is proved to be a general method in that it can be extended easily to different feature and model types. Furthermore, we study the comparisons and combinations of SAT and speaker adaptation in the context of DNNs. A data reduction strategy, frame skipping, is also employed to accelerate the training process of SAT-DNNs. Our experiments show that compared with the DNN baseline, our SAT-DNN model achieves 13.5% relative improvement in terms of WER. This improvement is enlarged to 17.5% when the LHUC-based model adaptation is further applied atop of SAT-DNNs. Furthermore, we evaluate our SAT-DNN approach on the larger Switchboard dataset and study better strategies to accelerate training of SAT-DNN models.

Chapter 6

Robust Speech Recognition with Distance-Aware DNNs

6.1 Background and Motivation

Robustness requires ASR systems to perform reasonably well on noisy, farfield speech and under unseen environment. Robustness has been a long-standing problem for acoustic modeling [56]. In recent years, DNN models have dramatically advanced the recognition accuracy on clean, close-talking speech. However, robustness still remains to be a challenge for DNNs. It is revealed in [43] that as with GMMs, the performance of DNNs drops significantly as the SNR decreases. Various attempts [57, 62, 102, 118] have been made to build noise-robust DNN models. For example, [102] proposes different methods, including multi-condition training and dropout training, to improve DNN models under low SNRs. In [62, 118], RNNs are employed for this purpose, either as a noise-reduction autoencoder or as the hybrid model directly.

Apart from noise, another critical type of variability is the distance between speakers and the microphones. Performance gap exists when we port ASR systems from close-talking to distant speech [112]. A number of techniques have been presented for improved DNN models on farfield speech. For instance, in [60, 112], DNN models are improved by combining speech signals from multiple distant microphones via concatenation or beamforming [63]. Also for DNN models, [122] evaluates the existing environmental robustness methods on a distant-microphone meeting transcription task. A robust front-end is derived by integrating different enhancement approaches and feature types. Although showing nice gains, these distant speech recognition (DSR) techniques have the limitation that most of them deal with constantly distant speech. That

is, the speaker-microphone distance (SMD) is assumed to be unchanged throughout the course of recording. However, in many real-world scenarios, the SMD can be quite dynamic. For example, a speaker is likely to walk around when talking to a far-field microphone located at a fixed position. In this case, the SMD varies a lot even within a single utterance, which poses special difficulty to acoustic modeling. In this case, the distance between the speaker and the microphone varies a lot, not only within the same video but also within the same utterance. In this thesis, we solve this problem by proposing the distance-aware DNN (DA-DNN) models. DA-DNNs capture the SMD information dynamically at the frame level.

Another line of work has focused on improving robustness of acoustic models with audio-visual ASR. The process of speech perception is bi-modal in nature. This has motivated researchers to combine audio and visual features in acoustic modeling [21, 31]. Previous work has generally adopted visual features extracted from the speaker's mouth region, including lip contours and mouth shapes. Although available in highly constrained videos, these features are not always obtainable from open-domain videos (e.g., YouTube videos). For example, in a large portion of the YouTube videos, the speakers do not appear in the video frames at all. Another limitation of the traditional audio-visual ASR is that the alignment between the speech and video frames is required. Since the speech and video streams have different sampling rates, aligning them may introduce inaccurate visual features into acoustic modeling. In this thesis, we explore open-domain audio-visual ASR by employing video/segment-level visual features. These visual feature can be readily obtained from real-world videos by models trained on external datasets.

This thesis aims to build DNN models robust to rich SMD variability. Our solution takes advantage of DNNs' flexibility to integrate heterogeneous features under the same optimization objective. We propose to construct distance-aware DNNs by explicitly incorporating the SMD information into DNN training. Past work [31, 60, 102] has attempted to incorporate various types of context information (e.g., noise estimate [102], speaker identity [60] and visual clues [31]) for robust DNN acoustic modeling. However, to our knowledge, no previous work has dealt with the explicit incorporation of dynamic distance information.

In this work, we achieve the incorporation of the SMD information by learning a universal SMD extractor. Such an extractor is a DNN with a bottleneck layer in the architecture. It is trained on a comprehensive meeting corpus, using SMD types as the classification targets. Then, we apply this extractor to our target acoustic modeling task. Specifically, each speech frame is fed into the extractor and activations of the bottleneck layer are taken as SMD descriptors. Distance-aware DNNs are built by appending these descriptors to the acoustic features (e.g., filterbanks) as the DNN inputs. Based on this basic framework, we make further optimizations

from two aspects.

- Besides DNNs, we study two alternative architectures, convolutional neural networks (CNNs) [4, 96, 107] and recurrent neural networks (RNNs) [30, 62, 97, 116], as the SMD extractor.
- A distance adaptive training (DAT) approach is proposed to utilize the SMD information more effectively.

6.2 Extraction of SMD Descriptors

We formulate the extraction of the SMD information as a problem of bottleneck-feature (BNF) generation. Three architectures are investigated as the building block of the extractor.

6.2.1 SMD Extraction with DNNs

Our first SMD extractor is a DNN that has a bottleneck layer significantly narrower than the other hidden layers. This bottleneck layer squeezes the discriminative information into a low-dimensional space. The network is trained on a dataset where the SMD type (close-talking, distant, etc.) of each acoustic frame is known. For example, in the ICSI meeting corpus [45], each meeting session is recorded with multiple microphones, and details regarding the locations of the microphones are provided. Training of the network is to classify speech frames to the SMD types. After network training, activations from the bottleneck layer are treated as SMD descriptors that capture the SMD variability dynamically at the frame level.

Our previous work [25] has established the deep BNF (DBNF) architecture for better BNF extraction. DBNF differs from the previous BNF approaches [32, 123] in that the hidden layers are arranged in an asymmetric manner around the bottleneck layer. In particular, we insert multiple hidden layers prior to the bottleneck layer, and only one hidden layer between the bottleneck and the softmax layers. As discovered in [125], activations from higher layers of DNNs are more invariant to acoustic distortions. In this work, we apply this DBNF structure as the SMD extractor.

6.2.2 SMD Extraction with CNNs

Instead of fully-connected (FC) weight matrices, CNNs are characterized by local filters which capture locality along the frequency bands. On top of the convolution layers, max-pooling layers are usually added to improve the shift invariance in the frequency domain. Because of these

configurations, CNNs have been shown to outperform DNNs on acoustic modeling tasks [4, 69, 96, 107]. In highly challenging acoustic conditions, the quality of the SMD descriptors might be undermined by spectral distortions such as noise and reverberation. Applying CNNs as the SMD extractor enables us to obtain SMD descriptors robust to these distortions. Following [107], our CNN-based extractor contains two convolution layers, on top of which multiple FC layers are added. One of the FC layers is a bottleneck layer for SMD descriptors generation. More details about our CNN settings are presented in Section 6.4.2.

6.2.3 SMD Extraction with RNNs

Both DNNs and CNNs can only model limited temporal dependency within the fixed-size context window. To resolve this limitation, previous work [30, 97, 98] has studied the application of RNNs to acoustic modeling. Unlike the standard feedforward networks, the RNN has self-connections on its hidden layers. These connections allow temporal information to be propagated through many time steps. We use the more complicated Long Short-Term Memory (LSTM) [40], which has been reviewed in Section 3.4, to construct RNNs. LSTM is a special recurrent layer that exploits memory cells to store temporal information and purpose-built gates to control the information flow. These modifications make RNNs particularly suited for modeling long-range temporal dynamics. In this work, we use a deep LSTM-RNN architecture, which stacks multiple LSTM layers, as the SMD extractor. Within each LSTM layer, following [97], we add a linear projection layer that transforms the memory cell activations into lower-dimensional outputs. In [97], adding this projection layer is found to reduce model parameters and generate better recognition accuracy. Interested readers can refer to [97] for more details.

6.3 Distance-Aware DNNs

The trained SMD extractor is transferred to our target acoustic modeling task on which distance-aware DNNs are then built. Note that the inputs to the SMD extractor and the inputs to the DNN acoustic model are not constrained to be identical. They may be trained with different feature types, e.g., MFCCs and filterbanks. Even with the same feature type, the SMD extractor and the DNN model may require different normalization on their front-end. At the t-th frame, the input vector of the DNN model is denoted as $\mathbf{o}_t$, and the inputs of the SMD extractor as $\mathbf{r}_t$. By feeding $\mathbf{r}_t$ to the extractor, we obtain its bottleneck layer activations $\mathbf{d}_t$ as the SMD descriptors. Two methods are investigated to incorporate these descriptors into DNN training.

6.3.1 Simple Concatenation

A simple way is to append these SMD descriptors to the original DNN inputs. Then, the DNN model is built over the augmented feature vectors $[\mathbf{o}_t, \mathbf{d}_t]$. During fine-tuning, the bottom layers are trained to fuse the SMD information and the acoustic features with non-linear transformations. The activations from these bottom layers become more invariant across SMD conditions, and thus benefit the CD states classification performed by the upper layers.

6.3.2 Distance Adaptive Training

In our previous work [75, 77], we have presented a framework to perform speaker adaptive training (SAT) for DNN models. This approach requires an i-vector [17] to be extracted for each speaker. Based on the well-trained speaker-independent (SI) DNN, a separate *adaptation neural network* is learned to convert i-vectors into speaker-specific linear feature shifts. Adding these shifts to the original DNN inputs (i.e., $\mathbf{o}_t$) produces a speaker-normalized feature space. Parameters of the SI-DNN are re-updated in this new space, which finally generates the SAT-DNN model.

This idea can be naturally ported to the SMD descriptors, and we conduct distance adaptive training (DAT) for DNNs. Specifically, we replace the i-vector representations with the SMD descriptors. A distance-normalized feature space is similarly derived by learning the adaptation network. Note that in this case, the linear feature shifts generated by the adaptation network are frame-specific rather than speaker-specific. Re-updating the parameters of the DNN in the normalized feature space gives us the adaptively trained DAT-DNN model. This DAT-DNN model becomes independent of specific SMD conditions, and thus generalizes better to unseen distance variability.

6.4 Experiments

6.4.1 Experimental Setup

Following [75], our target acoustic modeling task is to transcribe real-world *instructional videos*. To create the dataset, we download a collection of English videos from online video archives. These videos are uploaded by social media users to share expertise on specific tasks (e.g., oil change, sandwich making, etc.). Unlike broadcast news videos that are produced professionally, these amateur videos are shot using various types of portable devices (e.g., cameras, cellphones,

Table 6.1: Accuracy (%) of various SMD extractors on the ICSI meeting training data, where the classes are the 2311 combinations of speakers and distance types. The accuracy is measured on the frame level.

SMD Extractor	Training Accuracy (%)	Validation Accuracy (%)
DNN	43.46	39.71
CNN	51.86	46.78
LSTM	65.07	48.77

etc.) with far-field microphones. Also, in many of the videos, the speakers frequently change their locations relative to the microphones. For example, in a video about soccer skills, the speaker has to move around in order to perform various actions. All these factors make the SMD vary a lot, not only within the same video but also within a single utterance. This SMD variability poses special challenges to acoustic modeling on this task. On average, the downloaded videos have the duration of 90 seconds. For each video, the manual transcripts have been provided by the uploading user. We take several steps to convert the collected data into an ASR corpus. These steps include transcripts normalization, utterance filtering, audio downsampling, and lexicon expansion. We finally get 94 hours of speech, out of which 90 hours are selected for training and 4 hours for testing. A video is taken as a speaker. For decoding, a trigram language model (LM) is trained on the training transcripts. This LM is then interpolated with another trigram LM trained on an additional set of 300 hours of instructional-video transcripts.

We train the SMD extractor on the ICSI meeting corpus [45]. In this corpus, each meeting session has been recorded with microphones laid out at different distances from the speakers. However, the total number of channels is not constant across meeting sessions. Moreover, not enough details regarding the channels are provided in the corpus that enables us to align channels across meetings. For instance, the channels marked with "#1" in two different meetings are not necessarily referring to the same microphone. Therefore, instead of only distance types, the combinations of speakers and distance types are taken as the labels, which totally results in 2311 classes. A SMD extractor can be learned by taking these classes as the targets using the standard CE objective. In Table 6.1, we show the frame-level accuracy achieved by SMD extractors with different architectures.

Our GMM models are built with the open-source Kaldi toolkit [88]. We first train the initial MLE model using 39-dimensional MFCCs+$\triangle$+$\triangle\triangle$. Then 7 frames of MFCCs are spliced together and projected down to 40 dimensions with linear discriminant analysis (LDA). A MLLT is applied on the LDA features and generates the LDA+MLLT model. Discriminative training with the boosted maximum mutual information (BMMI) objective [86] is finally performed over

Table 6.2: Results (% WER) of the BMMI-GMM and baseline DNN models on the testing set.

Models	WER(%)
BMMI-GMM	26.1
DNN	23.4

the LDA+MLLT model. The GMM model has 3819 tied triphone states and an average of 16 Gaussian components per state.

We construct DNN models using our PDNN framework [67]. DNN inputs include 11 neighbouring frames (5 on each side of the center frame) of 40-dimensional log-scale filterbank coefficients, with per-video mean and variance normalization. The DNN model has 6 hidden layers each of which contains 1024 neurons. Parameters of the network are initialized randomly. DNN fine-tuning uses frame labels generated through forced alignment with the LDA+MLLT GMM model. The cross-entropy (CE) objective is optimized based on a decaying "newbob" learning rate schedule. Specifically, the learning rate starts from 0.08 and remains unchanged until the increase of the frame accuracy on a cross-validation set between two consecutive epochs falls below 0.2%. Then the learning rate is decayed by a factor of 0.5 at each of the subsequent epochs. The whole learning process terminates when the frame accuracy fails to improve by 0.2% between two successive epochs. We adopt the mini-batch size of 256 and the momentum of 0.5 for stochastic gradient descent (SGD). Table 6.2 shows the WERs of the BMMI-GMM and DNN models on the 4-hour testing set.

6.4.2 Experiments of SMD Extractors

In this section, we firstly investigate the optimal configurations for the SMD extractor. Training of the extractor uses filterbanks as the features. However, instead of per-video normalization, we apply global mean and variance normalization to preserve channel and speaker variation across videos. The trained SMD extractor is applied to our video-transcribing dataset, generating the SMD descriptors from its bottleneck layer. We employ the simple concatenation method for incorporating the SMD information. When formulated as a DNN, the extractor contains 6 hidden layers in which the 5-th layer is the bottleneck layer. All the non-bottleneck hidden layers have 1024 units. Table 6.3 shows the WERs of the resulting distance-aware DNNs when the SMD descriptors have different dimensions. In the best case, we place 100 units at the bottleneck layer and have the WER of 22.1%. As a comparison, we train the DNN-based SMD extractor using the 468 speakers (instead of speaker-channel combinations) as the targets. In this case,

Table 6.3: Results (% WER) of the SMD extractors with different configurations and architectures. "SMD Dim" refers to the dimension of the SMD descriptors, i.e., the size of the bottleneck layer in the SMD extractor.

SMD Extractor	SMD Dim	WER(%)
DNN	50	22.4
DNN	100	22.1
DNN	150	22.3
CNN	100	22.1
LSTM-RNN	100	21.8

the distance-aware DNN gets a much worse WER 23.0%. This verifies the necessity of adding channel (distance) targets in SMD extractor training.

Section 6.2 presents three architectures to instantiate the SMD extractor. We compare the performance of these architectures in Table 6.3. The CNN architecture follows [75, 107], consisting of 2 convolution layers. The convolution operation is applied over both time and frequency. Atop of the convolution layres, 4 FC hidden layers and finally the softmax layer are placed. The 5-th hidden layer, i.e., the 3-rd FC layer, is the bottleneck layer which has 100 neurons. In our LSTM-RNN architecture, we have 2 LSTM layers, each of which contains 800 memory cells and 512 output units. On top of these two LSTM layers, we have a bottleneck layer with 100 units which is followed by the final softmax layer. Unlike DNNs and CNNs, the LSTM-RNN takes single frames of filterbanks as its inputs, without any context splicing.

From Table 6.3, we observe that applying the CNN as the SMD extractor gives no improvement over the DNN extractor. This is partly because although showing rich SMD variability, our dataset is relatively clean, without much noise and reverberation. The advantage of CNNs in normalizing spectral distortions cannot be manifested under this condition. In comparison, the application of the LSTM-RNN results in more obvious gains (0.3% absolute). This demonstrates the ability of LSTM-RNNs to learn more accurate SMD descriptors by exploiting long-term temporal dependency. To this end, the distance-aware DNN achieves the WER of 21.8%, which translates to 6.8% relative improvement over the DNN baseline (23.4%).

6.4.3 Results of DAT-DNNs

In addition to the simple concatenation, Section 6.3.2 proposes DAT for incorporation of the SMD descriptors. In this section, we experimentally study the performance of DAT-DNN models. As with SAT of DNNs [75, 77], building of DAT-DNN models starts from the DNN baseline

Table 6.4: Results (% WER) of the adaptively trained DNN models. "Feat" means the type of additional descriptors used in adaptive training, e.g., SMD descriptors for DAT.

Adaptive Model	Feat	WER(%)
DAT-DNN	SMD descriptors	21.5
SAT-DNN	I-vectors	22.0
SAT+DAT-DNN	SMD descriptors + I-vectors	21.3

which has been well trained in Section 6.4.1. An adaptation network is learned to convert the SMD descriptors into frame-level linear features shifts. This adaptation network contains 3 hidden layers, each of which has 512 units and uses the sigmoid activation function. Its output layer has 440 (the dimension of the DNN input features) units and uses the identity function $f(x) = x$. After adding the shifts to the original DNN inputs, we obtain the distance-normalized feature space, in which the DNN model is re-updated. This gives us the DAT-DNN model. Training of the adaptation network and updating of the DNN use the standard back-propagation.

During decoding, we adapt the DAT-DNN model simply by extracting the SMD descriptors on the testing speech frames, feedforwarding the descriptors through the adaptation network, and adding the feature shifts to the original filterbank features. The DAT-DNN model is finally decoded in the normalized feature space. We use the SMD descriptors generated by the LSTM-RNN. Table 6.4 shows the results of the DAT-DNN model. Due to more complicated integration of the SMD information, our DAT approach outperforms the simple concatenation.

For complete evaluation, we also build the SAT-DNN model by extracting a 100-dimensional i-vector for each speaker. From Table 6.4, we observe that the improvement (22.0% vs 23.4%) of the SAT-DNN over the baseline DNN is not as large as the improvement reported in [75]. This is because the video-level i-vectors are insufficient to capture the rich SMD variability within videos/utterances. In contrast, DAT can model the SMD dynamics by taking advantage of the frame-level SMD descriptors. As a result, the DAT-DNN model ends up to achieve better results than the SAT-DNN. Finally, we concatenate the i-vectors and SMD descriptors into an expanded representation, which encodes both the video-level speaker characteristics and the frame-level SMD dynamics. Adaptive training is similarly carried out over these new representations. Table 6.4 shows that the resulting *SAT+DAT-DNN* model obtains a better WER than both the SAT-DNN and the DAT-DNN. Compared with the DNN baseline, the SAT+DAT-DNN results in 9.0% relative improvement (21.3% vs 23.4%).

Table 6.5: Analysis of the correlation between DA-DNN WER improvement and the SMD variability. "WER Improvement" refers to the relative improvement of the DA-DNN over the baseline DNN. "Average SMD Distance" means the average Euclidean distance of the SMD descriptors (vectors) across the frames.

WER Improvement	Average SMD Distance
greater than 15%	0.392
between 0% and 15%	0.334
equal to or smaller than 0%	0.285

6.4.4 More Analysis

So far, our experiments have shown that incorporating the SMD descriptors into DNNs brings notable gains. Here we conduct quantitative analysis to verify the correlation between WER gains and the SMD variability. The two models used in our analysis are: the baseline DNN model with the WER% of 23.4%, and the DA-DNN using the simple concatenation and with the WER% of 21.8%. On our decoding data, we break their WERs down to the 156 speakers (i.e., videos). The WER relative improvement is then computed on the per-video basis.

Meanwhile, we quantify the SMD variability by measuring the change of the extracted SMD descriptors. Specifically, for a specific video, we obtain the SMD vector for each of the frames from the LSTM-RNN extractor. Then we compute the Euclidean distance between every pair of neighbouring frames, and get the average distance across the frames as the overall indicator for the video. As shown by Table 6.5, on 18 testing videos, the DA-DNN achieves more than 15% relative improvement than the baseline DNN. On these 18 videos, the average SMD distance is 0.392. For a comparison, we randomly select 18 videos on which the improvement of the DA-DNN is between 0% and 15%, and the average distance is 0.334. The distance further reduces to 0.285 when we examine the videos on which no improvement is achieved. The overall observation from these comparisons is that the DA-DNN model gets more gains when the frame-wise SMD distance gets larger. This confirms our intuition towards distance-aware DNNs: the acoustic model incorporates the distance information dynamically and performs particularly well on videos with high SMD variability.

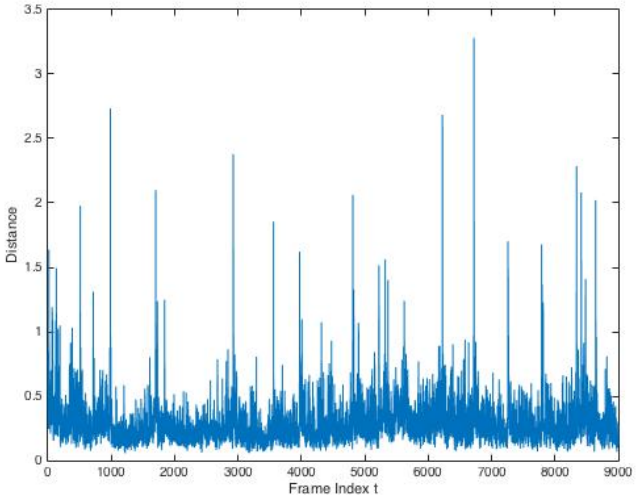
For further qualitative analysis, we select two videos on which incorporating the SMD descriptors gives significant (20% relative in this case) and no WER improvements respectively. For each video, we characterize its *SMD variability* by calculating the Euclidean distance of the SMD vectors between every pair of neighboring frames. More formally, given a video that has T speech frames, we obtain SMD descriptors after feeding the T frames into the SMD extractor.

The resulting SMD descriptors are a sequence of T vectors $[\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_T]$. Then the SMD variability is represented by a $T-1$ dimensional vector $[sv_1, sv_2, ..., sv_{T-1}]$, where each element sv_t is the Euclidean distance between the two vectors $\mathbf{d}_t$ and $\mathbf{d}_{t+1}$.

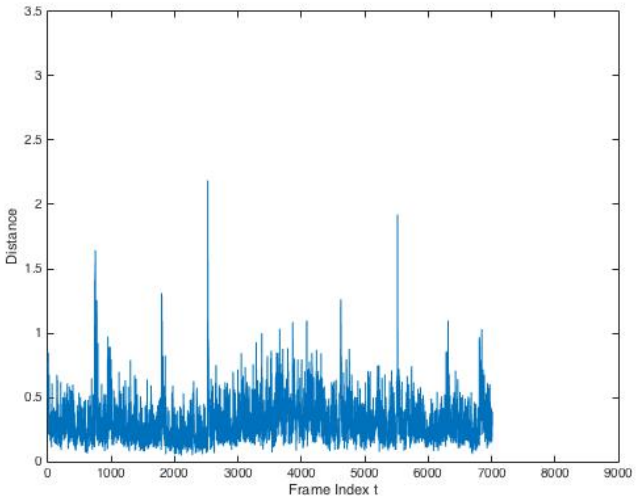
For each video, we plot the SMD variability vector sequentially along the time axis (speech frames). The visualization for the two selected videos is presented in Figure 6.1. We observe that on the video where the WER is improved by 20%, the SMD descriptors displays more variability than on the video where the WER sees no improvement. This demonstrates that the SMD descriptors we extract truly characterize the speaker-microphone information, and eventually enhance the robustness of acoustic models.

6.5 Summary

In this chapter, we have investigated the impact of dynamic SMD on the performance of DNN acoustic models. To alleviate the effect of SMD, we build distance-aware DNN models by explicitly incorporating SMD descriptors into DNN training. Extraction of the SMD information is achieved by a universal extractor that is learned on a meeting corpus. We study the effectiveness of different deep learning architectures acting as the SMD extractor. Also, two approaches, feature concatenation and adaptive training, are employed to incorporate the SMD descriptors into DNN model training. Our experiments show that distance-aware DNNs achieve 8.1% relative WER improvement over the DNN baseline. The SMD descriptors can be combined with speaker i-vectors into a more comprehensive representation which results in further WER reduction. In addition to the quantitative evaluations, we also conduct qualitative analysis, revealing the reason why incorporating the SMD information improves recognition accuracy.



(a) On the video where the WER is improved by 20% relatively with SMD descriptors incorporated.



(b) On the video where the WER has no improvement when SMD descriptors are incorporated.

Figure 6.1: Visualization of the SMD variability vectors along the time axis. The variability is quantified as the Euclidean distance of the SMD descriptors vectors for every pair of neighboring frames. The time axis is represented by the frame index.

Chapter 7

Open-Domain Audio-Visual Speech Recognition using Deep Learning

The pervasive deployment of speech interfaces requires ASR systems to handle various types of variability. In general, DNNs display superior recognition accuracy than the traditional GMMs models. However, robustness remains to be a challenge for DNN models [56]. For example, in [43], it is revealed that the performance of DNNs degrades significantly as the SNR drops. As revealed in Chapter 6, an effective strategy to deal with variability is to incorporate additional knowledge explicitly into DNN models.

When transcribing video data, in addition to the audio stream, we can also take advantage of the video stream, which provides additional information potentially helpful for acoustic modeling. For instance, images extracted from a specific video may indicate the scene/environment in which the speech data have been recorded. Also, actions (running, lifting, walking, etc.) performed by the speaker may correlate with the speaker's speaking rate or style. This section investigates the incorporation of various types of visual features into DNN acoustic models. Unlike previous work on audio-visual ASR [21, 31], we study visual features that can be extracted at the video or segment level, and thus enable open-domain audio-visual ASR on real-world video data.

7.1 Related Work on Audio-Visual ASR

ASR on video data naturally has access to two modalities: audio and video. The bimodal nature of speech perception motivates researchers to work on audio-visual ASR. The goal of this field

is to explore the visual modality to improve the performance of ASR, especially under noisy acoustic conditions with low SNRs. There exist two key problems for audio-visual ASR. The first problem is the extraction of the visual features that can potentially improve ASR. Previous work [8, 10, 21, 31, 49] has generally exploited visual features that are extracted from the speaker's mouth region. For example, in automatic lip-reading literatures [8, 19], areas of interest (AOI) centered around the lips are extracted to form the image features. Facilitating lip-reading in online recognition requires the deployment of face tracking and lip locating modules [20]. In [21], 24 coefficients of lip shape and intensity, together with their temporal dynamics, are generated as the visual descriptors. A more straightforward feature type is the gray-scale pixel values of the (downsampled) image covering the speaker's mouth [127]. In [49], visual features are obtained from pixel color using raster scan, i.e., 30-dimensional RGB features with 10 dimensions from each color. In [65], a real-world system is described to automatically track and extract lip regions, and thus alleviate the difficulty of obtaining these features. To alleviate the difficulty of obtaining visual features surrounding lips, the second problem lies in the fusion of the audio and visual modalities into the bimodal system. In practice, this fusion can be conducted either on the feature level [10, 49, 85] where a bimodal front-end is constructed with the two feature streams, or on the decision level [8, 21, 113] where outputs from classifiers are combined during the recognition stage. A major challenge for the feature-level fusion is the asynchrony of the audio and video streams. To solve this issue, attempts have been made to combine the classifier outputs (e.g., states likelihoods) at a coarser level, for example the phone or even the word level [21, 113]. Another useful tool to deal with this asynchrony is Dynamic Bayesian Networks (DBNs) which allow for different levels of information fusion and have shown effectiveness in audio-visual ASR [22, 29].

Despite of these advancement, audio-visual ASR still has limitations that prevent its deployment to real-world video data. For example, mouth/lip related visual features are not always available in open-domain videos. In this thesis, we aim to further remove these constraints and thus achieve truly open-domain audio-visual ASR.

7.2 Speaker Attributes and Actions

Different speakers have different acoustic characteristics which are highly correlated with speech realization. The information revealing the speaker's attributes (e.g., age, gender, race, etc.) is valuable to ASR tasks. Also, actions performed by the speaker may affect the speaker's speaking

rate or style. This section focuses on the extraction and incorporation of speaker attributes and actions for acoustic modeling.

7.2.1 Speaker Attributes

We first study speaker attributes that can be derived automatically from video frames. In the instructional video dataset we have constructed, we observe that the (principal) speaker tends to appear at the beginning for a brief introduction. Based on this observation, we extract only the frame at the position which is immediately after the first utterance starts. Then, this image, which is assumed to show the speaker, is submitted to the Face++ API (www.faceplusplus.com), a platform built by Megvii Inc. for compact, powerful, and cross-platform vision services. With Face++'s face recognition functionality, we obtain the classification results of 3 attributes: age, gender, and race. The returned results are processed in the following manner.

- *Age*. The returned value of age is continuous. We categorize the age value into 6 bins: $<$20, 20-30, 30-40, 40-50, 50-60, $>$60. These bins are represented by a 6-dimensional vector. Each of the 6 elements is a binary variable indicating whether the speakers age falls into the corresponding bin. For example, if the speaker is classified to be aged at 58, then the age attribute is denoted by the vector [0 0 0 0 1 0]. For some videos, no speaker attributes can be generated due to image resolution, illumination condition or timing of the speakers show-ups. In this case, we set the elements in the age vector uniformly, i.e., [0.16 0.16 0.16 0.16 0.16 0.16].
- *Gender*. The gender attribute has two categorical values: male and female. The gender classification result is converted into a 2-dimensional vector whose binary elements denote male and female respectively, i.e., [1 0] stands for male and [0 1] for female. If no value is returned, then the gender attribute is represented by the vector [0.5 0.5].
- *Race*. The race attribute has three categorical values: White, Black, and Asian. A 3-dimensional vector is employed to represent the 3 possible values of race, that is, [1 0 0], [0 1 0] and [0 0 1] represent White, Black and Asian respectively. With the missing value, the race attribute is encoded by [0.33 0.33 0.33].

The final attribute vector is assembled by concatenating these three sub-vectors. For example, the attribute vector for a 58-year-old, male and white speaker is [0 0 0 0 1 0 | 1 0 | 1 0 0]. An example is shown in Figure 7.1.

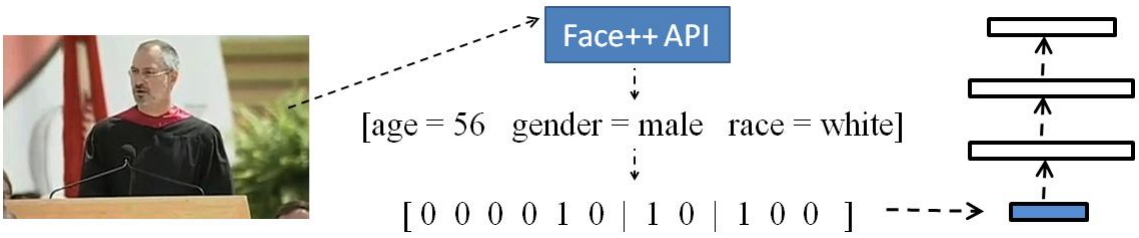


Figure 7.1: Incorporation of speaker attributes into DNN.

7.2.2 Speaker Actions

The second type of visual features is the actions performed by the speaker. To obtain the action information, in our corpus, we extract the video segments each of which corresponds to a speech utterance. Then, each of the video segments is fed to an action recognition system. This system has been trained with the UCF101 dataset [108]. UCF101 consists of realistic action videos collected from YouTube, having 101 action categories (e.g., baby crawling, surfing, applying eye makeup, sky diving). More details regarding our action recognition system can be retrieved in [46]. After performing action recognition, on each video segment, we get a 101-dimensional vector containing the probabilities over the 101 action classes. The dimension of these action features are further reduced to 50 through PCA. The resulting 50-dimensional vector is taken as the additional action information at the utterance level.

7.2.3 Experiments and Results

Following Section 6.4, our experiments are conducted on the video transcribing task where we attempt to recognize real-world instructional videos. After feature extraction, the resulting speaker attributes vector is on the video (speaker) level, and the actions vector is on the utterance level. These features are incorporated into DNN acoustic models via simple feature concatenation. Specifically, for the speaker attributes, the 11-dimensional attribute vector is appended to each speech frame belonging to this speaker. For the speaker actions, the 50-dimensional vector is appended to each frame coming from the utterance on which the action information has been generated. The DNN acoustic models are then built over the expanded feature vectors. Table 7.1 presents the performance of the resulting DNN models when the speaker attributes and speaker actions are incorporated. From Table 7.1, we can see that incorporating both types of visual features results in consistent improvement. When using speaker attributes, we get the WER of 22.8% which translates to 2.6% relative improvement (22.8% vs 23.4%). Incorporating speaker actions gives the WER of 22.9%, i.e., 2.1% relative improvement (22.9% vs 23.4%).

Table 7.1: Performance of DNN models when speaker attributes and actions are incorporated. The incorporation is achieved via simple feature concatenation.

Models	Additional Features	WER(%)
DNN (baseline)	—	23.4
DNN	11-dim speaker attributes	22.8
DNN	50-dim speaker actions	22.9

7.3 Deep Image Features

So far, we have studied two types of visual features, speaker attributes and speaker actions. Their effectiveness in helping acoustic modeling is verified on our video transcribing task. However, the application of these visual features still has limitations. For example, these features cannot be extracted accurately from videos where the speakers make no appearances at all. To further remove these limitations, we propose to exploit visual features that are extracted directly from the raw images (frames of the videos).

In this section, we aim to relax these constraints and extend audio-visual ASR to truly open-domain videos. Our approach to achieving this is based on deep learning, and can be split into two parts:

- We extract the visual features using deep architectures, i.e., deep convolutional neural networks (CNNs). Depending on the type of visual features we attempt to model, these deep architectures are trained on object recognition or scene labeling tasks. This enables us to obtain informative visual features from the raw pixels of open-domain videos. Also, with these deep models, we generate visual features on the segment (speech utterance) level, rather than on the frame level, which removes the need for time synchronization of the audio and video streams.
- We investigate audio-visual ASR in the context of DNN-based acoustic models. In addition to simple feature concatenation, we also apply an adaptive training framework to incorporate visual features in a more flexible way. Together with our previous work [70, 77, 79], we prove the generality of this adaptive training framework in fusing different types of additional information.

7.3.1 Extraction of Visual Features

Suppose we are dealing with an utterance u which has the acoustic features $\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T\}$, where T is the total number of speech frames. On a video transcribing task, there always exists

a video segment corresponding to u, with the same start and end time. This video segment is represented as $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N\}$, where N is the number of video frames in the video segment. From this video segment, we select a video frame $\mathbf{v}_n$ which serves as the image representation for the speech utterance. Two types of image features are extracted from $\mathbf{v}_n$.

Object Features. The huge success of deep learning in computer vision started from the object recognition task. Our first type of visual information is derived from object features. The intuition is that object features may encapsulate information regarding the acoustic environment/condition of the speech. For example, classifying an image to the classes "computer keyboard" and "monitor" may indicate that the speech segment has been recorded in an office. Distilling this knowledge during model training can improve the robustness of the acoustic models. We extract this object information using a deep CNN model which has been trained on a comprehensive object recognition dataset, e.g., ImageNet [18]. The resulting CNN model is referred to as *object-CNN*. Then the video frame $\mathbf{v}_n$ is fed into the CNN model, from which we get the distribution (posterior probabilities) over the object classes. These probabilities encode the object-related information that are incorporated into DNN acoustic models. Due to the high dimension of the probabilities, we may need to perform dimension reduction. More details regarding training of the CNN networks can be found in Section 7.3.3.

Place Features. The utility of the object features comes from the "place" information that is implicitly encoded by the object classification results. It is then natural to utilize place features in a more explicit way. To achieve this, we train a deep CNN model meant for the scene labeling task. Given a video frame, the classification outputs from this *place-CNN* encode the place information, which is then incorporated into acoustic models. For convenience of formulation, the resulting visual feature vector for this utterance u is represented as $\mathbf{f}_u$.

7.3.2 Incorporation of Visual Features

After obtaining the visual features, we adopt two methods to incorporate the visual features into DNN acoustic models.

Feature Concatenation

A simple way is to append the visual features to the original DNN inputs. Then, the DNN model is built over the augmented feature vectors. Within the utterance u, the augmented feature vector on the t-th frame now becomes [$\mathbf{o}_t$, $\mathbf{f}_u$]. During fine-tuning, the bottom layers in the DNN are trained to fuse the visual information and the acoustic features with non-linear transformations.

The activations from these bottom layers become more robust to environment/place variability, and thus benefit the CD states classification performed by the upper layers.

Visual Adaptive Training

In Section 5, we have presented a framework to perform SAT for DNN models. This approach requires an i-vector [17] to be extracted for each speaker. Based on the well-trained speaker-independent (SI) DNN, a separate *adaptation neural network* is learned to convert i-vectors into speaker-specific linear feature shifts. Adding these shifts to the original DNN inputs produces a speaker-normalized feature space. Parameters of the SI-DNN are re-updated in this new space, which finally generates the SAT-DNN model. This framework has also been applied successfully to descriptors of speaker-microphone distance, and thus achieves distance adaptive training (DAT) for DNNs [70].

In this thesis, we port this idea to the visual features, which enables us to conduct visual adaptive training (VAT) for DNNs. Specifically, in the SAT framework, we replace the i-vector representations with the visual features. An adaptation network is learned by taking the visual features as inputs, and then generates an adaptive feature space with respect to the visual descriptors. Note that in this case, the linear feature shifts generated by the adaptation network are utterance-specific rather than speaker-specific. Re-updating the parameters of the DNN in the normalized feature space gives us the adaptively trained VAT-DNN model. This VAT-DNN model takes advantage of extracted visual features as additional knowledge, and thus generalizes better to unseen variability.

7.3.3 Experimental Setup

Extraction of Object Recognition Features

In our experiments, we train the object-CNN model and extract the object recognition features by following the following setup.

- *CNN Architecture*. We turn to a deep CNN model for image feature extraction. This CNN model follows the standard AlexNet architecture [51]. The networks contains 5 convolution layers which use the rectifier non-linearity (ReLU) [27] as the activation function. On top of the convolution layers, we have 3 fully-connected (FC) layers. The last FC layer has the number of neurons equal to the number of classes, which is finally followed by a softmax layer. More details regarding the architecture can be found in Appendix A.

- *Data and Model Training*. We train our CNN-based feature extractor towards a object recognition task. Our training data come from ImageNet [18], a dataset containing over 15 million labeled images belonging to around 22,000 categories. We use a training set from the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) held in 2012. This ILSVRC training set is a subset of ImageNet, amounting to 1.2 million images covering 1000 classes. Each image from the training data is resized to the size of 256x256, which is then normalized with mean and variance normalization. From this resized image, we crop the 227x227 block from the center of the image, and feed this block as the CNN inputs. Therefore, the inputs into the CNN model have the size of 3x227x227, where 3 is the number of channels (RGB). Model training optimizes the standard cross-entropy (CE) objective. After training, we use a testing set from the ILSVRC 2012 challenge to briefly evaluate the performance of the CNN model. This testing set contains 50,000 images, i.e., 50 images for each class. The resulting CNN model achieves 20% top-5 error rate on the testing set.
- *Feature Extraction*. After obtaining the object-CNN model for object recognition, we can transfer this model to our acoustic modeling task. In our video transcribing task, each speech utterance corresponds to a video segment. From this video segment, we randomly select an image frame as the image representation for this speech utterance. Feeding this image to the trained CNN model gives us the classification results over the classes, that is, a 1000-dimensional probabilities vector. The same image preprocessing, which has been used in model training, is applied to the image: we resize the image to the size of 256x256, impose mean and variance normalization, and crop the 227x227 block from the image center. The generated probabilities vectors are taken as high-level image representations which encode semantic information regarding the image content. Due to the high dimension of these image representations, their dimension is reduced down to 100 via PCA. Finally, we get a 100-dimensional image feature for each video segment and its corresponding speech utterance. These 100-dimensional image features are leveraged as the additional visual information.

Extraction of Place Features

The place-CNN model is trained by adopting the following configurations.

- *Data*. We train the place-CNN on the MIT Places dataset [131] which contains 2.5 million images belonging to 205 scene categories. Examples of the scenes include “dinning room”,

Table 7.2: Performance of DNN models when object recognition features are incorporated. Two approaches are adopted for the incorporation: feature concatenation and adaptive training.

Models	Additional Features	WER(%)
DNN (baseline)	—	23.4
Feature Concatenation	100-dim object features	23.0
Adaptive Training	100-dim object features	22.5

"coast", "conference center", "courtyard", etc. We use the complete set of 2.5 million images to train the place-CNN model, and follow the same image preprocessing as used on ImageNet. Each image is resized to the size of 256x256, which is then normalized with mean and variance normalization. The center 227x227 block cropped from the image is fed as the network inputs. The resulting place-CNN model achieves the classification accuracy of 50% on a testing set from the same Places205 database.

- *CNN Architecture*. The architecture of place-CNN is almost the same as that of the object-CNN model. The only difference is that in the final FC layer, place-CNN has 205 neurons corresponding to the 205 scene classes, whereas object-CNN contains 1000 classes.
- *Feature Extraction*. After training the place-CNN model, we can apply this model to our acoustic modeling task. Again, for each speech utterance, we select an image frame as its image representation, and feed this image to the place-CNN model. From this model, we get the scene classification results (e.g., a 205-dimensional probabilities vector) whose dimension is further reduced via PCA. The resulting features are incorporated as additional image features into DNN acoustic modeling.

7.3.4 Results

Results of Object Features

Table 7.2 shows the results of the final DNN models on our video transcribing task. When we perform VAT, the adaptation network takes the 100-dimensional utterance-level visual features as the inputs. The adaptation network has 3 hidden layers each of which has 512 hidden units. From Table 7.2, we can see that incorporating object features results in moderate but consistent improvement. In particular, when applying the VAT approach, we obtain 0.9% absolute improvement over the baseline DNN (22.5% vs 23.4%), which translates to 3.8% relative improvement.

Table 7.3: Performance of DNN models when place features are incorporated. Two approaches are adopted for the incorporation: feature concatenation and adaptive training.

Models	Additional Features	WER(%)
DNN (baseline)	—	23.4
Feature Concatenation	50-dim place features	23.0
Feature Concatenation	100-dim place features	22.7
Feature Concatenation	150-dim place features	22.9
Adaptive Training	100-dim place features	22.5

Results of Place Features

Table 7.3 shows the results of the final DNN models with scene features incorporated. For feature concatenation, different configurations for the dimension of scene features are evaluated. We can see that using 100-dimensional scene features results in the best WER 22.7%. This number is better than the result of feature concatenation with the object recognition features extracted by the object-CNN. This reveals that the major benefit of using image features lies in the scene/place information encoded by the image features. Switching to the place-CNN enables us to eliminate the information generated by the object-CNN that is unrelated to scenes/places. That is why we are able to get superior recognition performance.

The 100-dimensional scene features are further applied to adaptive training. By doing this, we continue to get gains, bringing the WER down to 22.5%. Compared to the baseline DNN, we obtain 0.9% absolute improvement (i.e., 3.8% relative improvement) to this end. Moreover, the adaptive training model with the place features gets the same WER as the adaptive training model with the object recognition features. This means that although containing noise, the object recognition features can be transformed into a more effective representation with the adaptation network in the adaptive training framework.

Qualitative Analysis

In addition to the overall recognition accuracy, we also want to analyse the effects of incorporating the scene information on individual videos. Specifically, for each video, we compare the WER achieved by the feature concatenation model with the WER achieved by the baseline DNN model. Figure 7.3 exemplifies the videos on which feature concatenation (with scene information incorporated) ends up to have the same WER as the baseline DNN model. In our training data, a large majority of the videos have been filmed in indoor environments (e.g., offices). This data distribution enables the indoor environments to be sufficiently modeled by the acoustic model.

(a) Kitchen, WER 30.95% → 27.38%

(b) Baseball field, WER 18.70% → 15.65%

(c) Airport apron, WER 34.12% → 28.24%

(d) Train, WER 44.71% → 38.24%

Figure 7.2: Examples of videos on which incorporating the scene information gives over 10% relative improvement. For each video, we show an image frame extracted from it. "A% → B%" means incorporating the scene information reduces the WER **on this video** from A% to B%.

This explains why in Figure 7.3, adding the additional scene information results in either no or minor gains on indoor videos.

In comparison, Figure 7.2 exemplifies the videos on which feature concatenation outperforms the baseline DNN model by at least 10% relatively. We observe that most of these videos are recorded either in outdoor environments (e.g., baseball field, airport apron, street, etc.) , or in non-typical indoor conditions (e.g., kitchen, music studio, etc.) where music/noise may interfere with the actual speech a lot. Adding the scene descriptors helps the DNN model normalize the acoustic characteristics of these rare conditions, and thus benefits the generalization to unseen testing speech.

Finally we go through all the 156 testing videos and annotate each video into two categories: "typical indoor" (e.g., office) and "other". The "other" category can either be outdoor environments (e.g., baseball field, airport apron, street, etc.) or non-typical indoor conditions (e.g., kitchen, music studio, etc.). This annotation process gives us 111 typical indoor videos and 45

(a) Indoor, WER 27.55% → 27.55%

(b) Indoor, WER 22.27% → 22.27%

(c) Indoor, WER 16.42% → 16.42%

(d) Indoor, WER 26.44% → 26.44%

Figure 7.3: Examples of videos on which incorporating the scene information gives no improvement. For each video, we show an image frame extracted from it. "A% → B%" means incorporating the scene information changes the WER **on this video** from A% to B%.

Table 7.4: WER break down onto the two video categories: "typical indoor" and "other", for both the baseline DNN and also the DNN trained with the place features. The feature concatenation approach is adopted for image feature incorporation.

Video Category	WER% of the baseline DNN	WER% of the DNN with place features
typical indoor	22.1	21.7
other	27.6	25.7

other videos. WERs are computed on these two types of videos, with and without the place information incorporated. Table 7.4 shows that on both video types, we are able to get gains by leveraging the additional place features extracted from images. However, the improvement obtained on "other" is more significant than the improvement on "typical indoor". This shows that by taking advantage of the place features, DNN models can smooth out the variability of acoustic environment, and benefit recognition on the "other" category which is not sufficiently covered in training.

7.4 Fusion of Visual Features

So far, we have investigated three types of visual features: speaker attributes, speaker actions and image features. This final subsection attempts to fuse these three feature types together into a comprehensive representation. Specifically, we concatenate the 11-dimensional speaker attributes, 50-dimensional speaker actions, and 100-dimensional scene features into a combined representation. Then these combined features are incorporated into DNN models with feature concatenation or adaptive training. The results are shown in Table 7.5.

When using the 161-dimensional combined visual features, the feature concatenation approach gets the same WER as using the 100-dimensional place features. This is because with feature concatenation, the visual features are simply concatenated with the original acoustic features. With a larger dimension, the visual features are likely to outweigh the acoustic features, which undermines the utility of visual feature incorporation. On the contrary, due to its flexibility of feature learning, the adaptive training approach continues to get gains, reducing the WER down to 22.3%. Compared to the baseline DNN, this translates to 4.7% relative improvement.

Finally, we eliminate the speaker actions which have given the least improvement among the three visual feature types. The combined features then amount to 111 dimensions, consisting of the speaker attributes and the place features. From Table 7.5, we observe that shrinking the visual features gives us worse WERs, for both the feature concatenation and adaptive train-

Table 7.5: Performance of DNN models when we incorporate the combined visual features. Tow approaches, feature concatenation and adaptive training, are adopted for visual feature incorporation.

Models	Additional Features	WER(%)
DNN (baseline)	—	23.4
Feature Concatenation	161-dim combined features	22.7
Adaptive Training	161-dim combined features	22.3
Feature Concatenation	111-dim speaker attributes + place features	22.9
Adaptive Training	111-dim speaker attributes + place features	22.5

ing methods. This indicates that the three visual feature types are complimentary, collectively forming a comprehensive visual representation for speech signals.

7.5 Fusion of Speaker I-Vectors and Visual Features

Finally, we combine the visual features with speaker i-vectors. Specifically, we concatenate the 161-dimensional visual features and the 100-dimensional speaker i-vectors into a combined representation. These combined features are utilized for DNN models with feature concatenation or adaptive training. The results are shown in Table 7.6.

When simple concatenation is applied for feature incorporation, the 261-dimensional combined features outperform the 100-dimensional i-vectors. However, they performed worse than when only the 161-dimensional visual features are used. This is mainly because the additional information becomes too high-dimensional, and thus the relative weight of the original acoustic features is reduced inappropriately in the final front-end. In contrast, when we apply the adaptive training approach, the WER is reduced to 21.5% with the 261-dimensional combined features. This number outperforms both the speaker i-vectors (22.0%) and the 161-dimensional visual features (22.3%). This again demonstrates the advantage of the adaptive training framework in transforming context information and selecting the salient elements from additional features. To this end, compared to the baseline DNN model, our adaptive training approach gets 8.1%, when the visual features and speaker i-vectors are fused together as additional context information.

Table 7.6: Performance of DNN models when we incorporate the combined i-vector and visual features. Tow approaches, feature concatenation and adaptive training, are adopted for feature incorporation.

Models	Additional Features	WER(%)
DNN (baseline)	—	23.4
Feature Concatenation	161-dim visual features	22.7
Feature Concatenation	100-dim speaker i-vectors	23.0
Feature Concatenation	261-dim combined features	22.8
Adaptive Training	161-dim visual features	22.3
Adaptive Training	100-dim speaker i-vectors	22.0
Adaptive Training	261-dim combined features	21.5

7.6 Summary

In this chapter, our goal is to extend the existing audio-visual ASR idea to open-domain videos. We achieve this by taking advantage of video- or segment-level visual features, and adopting deep learning methods. We first investigate two speaker-related visual features, the video-level speaker attributes and the segment-level speaker actions. For DNNs acoustic models, employing these two types of visual features gives moderate but consistent improvements. To further remove the constraints imposed by these speaker-related features, we exploit visual features extracted directly from video frames (images). A unified deep learning framework is presented to accomplish this. First, the visual features are extracted using deep learning architectures which have been pre-trained on computer vision tasks, e.g., object recognition and scene labeling. Second, the visual features are incorporated into ASR under deep learning based acoustic modeling. In addition to simple feature concatenation, we also apply an adaptive training framework to incorporate visual features in a more flexible way. On our video transcribing task, audio-visual ASR using deep image features gets 3.8% relative improvement in terms of WER, compared to ASR merely using speech features. The image features can be readily combined with the speaker-related features, which increases the improvement of the audio-visual model to 4.7% relative. Fusing the various types of visual features with speaker i-vectors results in further WER reduction.

Chapter 8

Conclusions and Future Work

The goal of this thesis is to improve speech recognition by incorporating additional context information into DNN acoustic models. Bearing the challenges faced by modern speech recognition in mind, this thesis has answered two principal questions. First, what additional information/features are potentially useful for ASR? In response to this question, we have conducted a systematic study about various types of additional information in enhancing ASR accuracy. Depending on their sources, these information are categorized into four types: language, speaker, distance and video. Second, how can we incorporate these additional information into DNNs acoustic models? We have presented a novel adaptive training approach to achieving this feature incorporation effectively. This approach is applicable to different types of additional information, and consistently outperforms the simple feature concatenation method. The major conclusions are summarized as follows for each part of the thesis, and the future work is presented in the final section.

8.1 Cross-Language DNNs with Language-Universal Feature Extractors

Due to large model size, DNN acoustic models suffer from data scarcity. Under low-resource languages, we perform cross-language DNN acoustic modeling by borrowing knowledge from other languages. Knowledge transfer across languages is achieved via language-universal feature extractors (LUFEs) trained over a group of source languages. The contributions of this thesis are summarized as follows.

- We have proposed a framework to perform cross-language DNN acoustic modeling with

language-universal feature extractors (LUFEs). The LUFE is trained over a group of source languages in the form of multi-task learning, and then transferred to the target language.

- The quality of the LUFEs is improved by applying CNNs and DMNs as the building block for LUFEs. These two architectures have the advantage of generating invariant and sparse feature representations respectively. Combining these two architectures gives us the best LUFE signified by the lowest WER on the target language.
- Training LUFEs is inherently expensive, especially with large amounts of speech from the source languages. We have proposed a distributed learning strategy DistModel to speed up training of LUFEs. DistModel accelerates LUFE learning significantly, while causing negligible WER loss on the target language.

8.2 Speaker Adaptive Training of DNN Models using I-vectors

The performance of ASR systems degrades when a mismatch exists between the training and testing speakers/conditions. An effective approach to alleviating speaker mismatch is to perform SAT during training and speaker adaptation during testing. Meanwhile, the acoustic characteristics of speakers can be encapsulated by compact vector representations such as i-vectors. The contributions of this thesis are summarized as follows.

- We have proposed a framework to perform SAT for DNN acoustic models. The SAT approach relies on i-vectors and the adaptation neural network to realize feature normalization.
- Detailed empirical evidence has been presented to determine the optimal configurations for building the SAT-DNN models.
- We have investigated the further combination of SAT and model-space speaker adaptation during the recognition stage.
- A data reduction strategy, frame skipping, has also been employed to accelerate the training process of SAT-DNNs.
- The value of the proposed SAT approach is that it is a general approach. It can be extended easily to different network inputs and model architectures, and shows consistent gains. Also, as demonstrated by Chapter 6 and Chapter 7, this framework can also be applied to different types of additional information, and the fusion of them.

8.3 Robust Speech Recognition with Distance-Aware DNNs

The pervasive deployment of speech interfaces requires ASR systems to handle environmental variability effectively. A critical variability lies in the distance between the speaker and microphone. Distant speech recognition (DSR) remains to be a challenge for DNN models. This thesis has dealt with the problem of dynamic speaker-microphone distance (SMD), and proposed to tackle it by constructing distance-aware DNNs.

- To alleviate the effect of dynamic SMD, we have proposed to build distance-aware DNN models by explicitly incorporating frame-level SMD descriptors into DNN training.
- Extraction of the SMD information has been achieved by a universal extractor that is learned on a comprehensive meeting corpus. We have studied the effectiveness of different deep learning architectures (i.e., DNNs, CNNs, and RNNs) acting as the SMD extractor.
- In addition to simple feature concatenation, a distance adaptive training (DAT) approach has been proposed to utilize the SMD information more effectively. Within the adaptive training framework, the SMD descriptors can be combined with speaker i-vectors into a more comprehensive representation which results in further WER reduction.
- In addition to the quantitative evaluations, we also conduct qualitative analysis, revealing the reason why incorporating the SMD information improves recognition accuracy.

8.4 Open-Domain Audio-Visual Speech Recognition using Deep Learning

To further improve the robustness of ASR systems, this thesis has attempted to extend the existing audio-visual ASR idea to open-domain videos. We achieve this by taking advantage of video- or segment-level visual features, and adopting deep learning based fusion methods. The contributions of this thesis are as follows.

- We have investigated two speaker-related visual features, the video-level speaker attributes and the segment-level speaker actions. Incorporating these two features consistently improves the performance of DNN models.
- To further remove the constraints of speaker-related features, we have proposed to take advantage of visual features extracted directly from raw images.
- Two types of deep image features, object and place features, have been examined as the

additional information. These features are generated by deep architectures that have been trained for object recognition and scene classification tasks.

- In addition to simple feature concatenation, we have applied the adaptive training framework to incorporate visual features in a more flexible way.
- Combination of the visual features, and fusion of the visual features and speaker i-vectors have been empirically studied. Also, qualitative analysis has been presented to reveal the impact of the visual features in the ASR performance.

8.5 Future Work

In this thesis, our acoustic modeling has been based on DNNs and CNNs models. In the future, we would like to extend the incorporation of additional context information to RNNs based acoustic models. We would like to study how to combine the context information with single-frame RNNs inputs. Also, how to apply the adaptive training approach to RNNs models is a question worth examining.

Another direction we would like to pursue is the incorporation of context information into end-to-end ASR pipelines. Our work in this thesis is constrained to the conventional ASR pipeline, as reviewed in Chapter 3. Recently end-to-end paradigms have shown encouraging results for speech recognition [78]. It is interesting to investigate the utility of the additional information in improving end-to-end speech recognition. Also, end-to-end ASR systems generally employ LSTMs as the acoustic models. This complicated recurrent architecture opens up new opportunities for feature fusion. For example, the LSTM structure can be enriched with new gates which regulate the memory cells states based on the context information.

Lastly, we would like to study the adaptation of language models using the visual features. This thesis has proposed to fuse the object- and place-related visual features into acoustic models. These visual features are potentially useful also for language modeling. For example, objects recognized from the video stream are correlated with words (presumably nouns) appearing in the transcripts. A straightforward approach would be to train RNNs based language models and attach the visual features to the RNNs inputs. It is also interesting to discover how the vision-enhanced acoustic models and vision-enhanced language models interact and complement with each other.

Appendix A

Architecture of the Object-CNN Network

The Object-CNN network for extraction of object recognition features used in Section 7.3.3 has the configuration listed in Table A.1. The network contains 5 convolution layers which use the rectifier non-linearity (ReLU) [27] as the activation function. In the first and second convolution layers, a local response normalization (LRN) layer is added after the ReLU activation, and a max pooling layer follows the LRN layer. In the third and fourth convolution layers, we do not apply the LRN and pooling layers following the ReLU non-linearity. In the fifth convolution layer, we only apply the max pooling layer, without LRN applied. On top of the convolution layers, we have 3 fully-connected (FC) layers. The first and second FC layers have 4096 neurons, on top of which we apply the ReLU activation function and dropout regularization (with the dropout factor of 0.5). The third (last) FC layer has the number of neurons equal to the number of classes, which is finally followed by a softmax layer.

Table A.1: Configurations of the convolution, LRN and max pooling layers in the AlexNet architecture used in our experiments. The layers are listed by the order, from bottom to top, in the architecture. "# Feature Maps", "Kernel Size" and "Conv Stride" represent the number of feature maps, the size of the kernels, and the stride of the convolution operations in the convolution layers. "LRN Size" represents the size of the neighboring region over which LRN is performed. "Pooling Size" and "Pooling Stride" represent the size and stride for the max pooling layer.

Layers	# Feature Maps	Kernel Size	Conv Stride	LRN Size	Pooling Size	Pooling Stride
conv1	96	11	4	—	—	—
relu1	—	—	—	—	—	—
norm1	—	—	—	5	—	—
pool1	—	—	—	—	3	2
conv2	256	5	1	—	—	—
relu2	—	—	—	—	—	—
norm2	—	—	—	5	—	—
pool2	—	—	—	—	3	2
conv3	384	3	1	—	—	—
relu3	—	—	—	—	—	—
conv4	384	3	1	—	—	—
relu4	—	—	—	—	—	—
conv5	256	3	1	—	—	—
relu5	—	—	—	—	—	—
pool5	—	—	—	—	3	2

Appendix B

List of Abbreviations

ASR – Automatic Speech Recognition
BMMI – Boosted Maximum Mutual Information
BNF – Bottleneck Feature
BPTT – Back-Propagation through Time
CNN – Convolutional Neural Network
CD – Context Dependent
CE – Cross Entropy
CI – Context Independent
DA-DNN – Distance-Aware DNN
DAT – Distance Adaptive Training
DBN – Dynamic Bayesian Network
DBNF – Deep Bottleneck Feature
DMN – Deep Maxout Network
DNN – Deep Neural Network
DSR – Distant Speech Recognition
DT – Discriminative Training
EM – Expectation Maximization
FSA – Finite-State Acceptor
fMLLR – Feature-space MLLR
GMM – Gaussian Mixture Model
HMM – Hidden Markov Model
ILSVRC – ImageNet Large-Scale Visual Recognition Challenge
LDA – Linear Discriminant Analysis

LM – Language Model

LSTM – Long Short-Term Memory

LUFE – Language-Universal Feature Extractor

LVCSR – Large Vocabulary Continuous Speech Recognition

MFCC – Mel Frequency Cepstral Coefficient

MLE – Maximum Likelihood Estimation

MLLR – Maximum Likelihood Linear Regression

MLLT – Maximum Likelihood Linear Transform

MAP – Maximum A Posterior

MLP – Multilayer Perceptron

PCA – Principal Component Analysis

PLP – Perceptual Linear Prediction

RNN – Recurrent Neural Network

SAT – Speaker Adaptive Training

SDA – Stacked Denoising Autoencoder

SGD – Stochastic Gradient Descent

SGMM – Subspace Gaussian Mixture Model

SI – Speaker Independent

SMD – Speaker-Microphone Distance

SVD – Singular Value Decomposition

TDNN – Time-Delay Neural Network

WER – Word Error Rate

WFST – Weighted Finite-State Transducer

UBM – Universal Background Model

VTLN – Vocal Tract Length Normalization

Bibliography

[1] Ossama Abdel-Hamid and Hui Jiang. Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7942–7946. IEEE, 2013. 1.2.2, 5.1.2

[2] Ossama Abdel-Hamid and Hui Jiang. Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition. In *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1248–1252. ISCA, 2013. 1.2.2, 5.1.2

[3] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 3366–3370. ISCA, 2013. 1, 3.3, 4.3.1, 5.2.3

[4] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(10):1533–1545, 2014. 1, 3.3, 4.3.1, 5.2.3, 6.1, 6.2.2

[5] Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul. A compact model for speaker-adaptive training. In *Fourth International Conference on Spoken Language (ICSLP*, volume 2, pages 1137–1140. IEEE, 1996. 1.2.2, 5, 5.1.1

[6] Tasos Anastasakos, John McDonough, and John Makhoul. Speaker adaptive training: a maximum likelihood approach to speaker normalization. In *Acoustics, Speech and Signal Processing (ICASSP), 1997 IEEE International Conference on*, pages 1043–1046. IEEE, 1997. 1.2.2, 5, 5.1.1

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies

with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994. 3.4

[8] Christoph Bregler, Hermunn Hild, Stefan Manke, and Alex Waibel. Improving connected letter recognition by lipreading. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 1, pages 557–560. IEEE, 1993. 7.1

[9] Lukas Burget, Petr Schwarz, Mohit Agarwal, Pinar Akyazi, Kai Feng, Arnab Ghoshal, Ondrej Glembek, Nagendra Goel, Martin Karafiát, Daniel Povey, et al. Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4334–4337. IEEE, 2010. 4.1

[10] Tsuhan Chen. Audiovisual speech processing. *Signal Processing Magazine, IEEE*, 18(1): 9–21, 2001. 1.2.4, 7.1

[11] Justin Chiu and Alexander Rudnicky. Using conversational word bursts in spoken term detection. In *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2013. 4.3.3

[12] Justin Chiu, Yun Wang, Jan Trmal, Daniel Povey, Guoguo Chen, and Alexander Rudnicky. Combination of fst and cn search in spoken term detection. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 4.3.3

[13] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012. 1

[14] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013. 4.1

[15] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012. 4.4.1

[16] Najim Dehak, Reda Dehak, Patrick Kenny, Niko Brümmer, Pierre Ouellet, and Pierre Dumouchel. Support vector machines versus fast scoring in the low-dimensional total

variability space for speaker verification. In *Tenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1559–1562. ISCA, 2009. 5.1.2, 5.1.3, 5.1.3

[17] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(4):788–798, 2011. 5.1.2, 5.1.3, 5.1.3, 6.3.2, 7.3.2

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 7.3.1, 7.3.3

[19] Paul Duchnowski, Uwe Meier, and Alex Waibel. See me, hear me: integrating automatic speech recognition and lip-reading. In *ICSLP*, volume 94, pages 547–550, 1994. 7.1

[20] Paul Duchnowski, Martin Hunke, Dietrich Büsching, Uwe Meier, and Alex Waibel. Toward movement-invariant automatic lip-reading and speech recognition. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 109–112. IEEE, 1995. 7.1

[21] Stéphane Dupont and Juergen Luettin. Audio-visual speech modeling for continuous speech recognition. *Multimedia, IEEE Transactions on*, 2(3):141–151, 2000. 1.2.4, 6.1, 7, 7.1

[22] Virginia Estellers and Jean-Philippe Thiran. Overcoming asynchrony in audio-visual speech recognition. In *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, pages 466–471. IEEE, 2010. 7.1

[23] Mark JF Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998. 1.2.2, 2.1.6, 5, 5.1.1

[24] Jonas Gehring, Wonkyum Lee, Kevin Kilgour, Ian Lane, Yajie Miao, and Alex Waibel. Modular combination of deep neural networks for acoustic modeling. In *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2013. 1, 3.2, 4.3.3, 5.3.3, 5.4.1

[25] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3377–3381. IEEE, 2013. 1, 3.2, 4.3.3, 5.3.3, 5.4.1, 6.2.1

[26] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *The Journal of Machine Learning Research*, 3:115–143, 2003. 3.4

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323, 2011. 7.3.3, A

[28] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327, 2013. 1.2.1, 4.3.2

[29] John N Gowdy, Amarnag Subramanya, Chris Bartels, and Jeff Bilmes. Dbn based multi-stream models for audio-visual speech recognition. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. IEEE, 2004. 7.1

[30] Alex 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*, pages 6645–6649. IEEE, 2013. 1, 3.4, 5.2.3, 6.1, 6.2.3

[31] Guillaume Gravier, Gerasimos Potamianos, and Chalapathy Neti. Asynchrony modeling for audio-visual speech recognition. In *Proceedings of the second international conference on Human Language Technology Research*, pages 1–6. Morgan Kaufmann Publishers Inc., 2002. 1.2.4, 6.1, 7, 7.1

[32] Frantisek Grezl and Petr Fousek. Optimizing bottle-neck features for LVCSR. In *Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on*, pages 4729–4732. IEEE, 2008. 3.2, 6.2.1

[33] Vishwa Gupta, Patrick Kenny, Pierre Ouellet, and Themos Stafylakis. I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6334–6338. IEEE, 2014. 1.2.2, 5.1.2, 5.1.3

[34] Reinhold Haeb-Umbach and Hermann Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 13–16. IEEE, 1992. 2.1.6

[35] John B Hampshire, Alex H Waibel, et al. The meta-pi network: Connectionist rapid adaptation for high-performance multi-speaker phoneme recognition. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 165–168. IEEE, 1990. 5.1.2

[36] Georg Heigold, Vincent Vanhoucke, Andrew Senior, Patrick Nguyen, M Ranzato, Matthieu Devin, and Jeffrey Dean. Multilingual acoustic models using distributed deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8619–8623. IEEE, 2013. 1.2.1, 4.4.1

[37] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional HMM systems. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1635–1638. IEEE, 2000. 3.2

[38] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012. 1

[39] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 4.1, 4.3.3

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3.4, 6.2.3

[41] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE, 2013. 1.2.1, 4.1, 4.2, 4.3, 4.3.3

[42] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Foreword By-Reddy. *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice Hall PTR, 2001. 2.1.4

[43] Yan Huang, Dong Yu, Chaojun Liu, and Yifan Gong. A comparative analytic study on the Gaussian mixture and context dependent deep neural network hidden Markov models. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 1.1, 6.1, 7

[44] Navdeep Jaitly, Patrick Nguyen, Andrew W Senior, and Vincent Vanhoucke. Applica-

tion of pretrained deep neural networks to large vocabulary speech recognition. In *Thirteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2012. 1.1

[45] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. The icsi meeting corpus. In *Acoustics, Speech and Signal Processing (ICASSP), 2003 IEEE International Conference on*, pages I–364. IEEE, 2003. 6.2.1, 6.4.1

[46] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014. 7.2.2

[47] Martin Karafiát, Lukás Burget, Pavel Matejka, Ondrej Glembek, and J Cernocky. iVector-based discriminative adaptation for automatic speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 152–157. IEEE, 2011. 5.1.3

[48] Penny Karanasou, Yongqiang Wang, Mark JF Gales, and Philip C Woodland. Adaptation of deep neural network acoustic models using factorised i-vectors. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 5.1.2

[49] Yosuke Kashiwagi, Masayuki Suzuki, Nobuaki Minematsu, and Keikichi Hirose. Audio-visual feature integration based on piecewise linear transformation for noise robust automatic speech recognition. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 149–152. IEEE, 2012. 1.2.4, 7.1

[50] Patrick Kenny, Pierre Ouellet, Najim Dehak, Vishwa Gupta, and Pierre Dumouchel. A study of interspeaker variability in speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(5):980–988, 2008. 5.1.3

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 3.3, 7.3.3

[52] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104, 2009. 4.3.2

[53] Christopher J Leggetter and Philip C Woodland. Maximum likelihood linear regression

for speaker adaptation of continuous density hidden Markov models. *Computer Speech & Language*, 9(2):171–185, 1995. 1.2.2, 5, 5.1.1

[54] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1695–1699. IEEE, 2014. 5.3.3

[55] Bo Li and Khe Chai Sim. Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems. In *Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2010. 1.2.2, 5.1.2

[56] Jinyu Li, Li Deng, Yifan Gong, and Reinhold Haeb-Umbach. An overview of noise-robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(4):745–777, 2014. 6.1, 7

[57] Jinyu Li, Jui-Ting Huang, and Yifan Gong. Factorized adaptation for deep neural network. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5537–5541. IEEE, 2014. 5.1.2, 6.1

[58] Hank Liao. Speaker adaptation of context dependent deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7947–7951. IEEE, 2013. 1.2.2, 5

[59] Hank Liao, Erik McDermott, and Andrew Senior. Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 368–373. IEEE, 2013. 1.1, 5.1.2

[60] Yulan Liu, Pengyuan Zhang, and Thomas Hain. Using neural network front-ends on far field multiple microphones based speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5542–5546. IEEE, 2014. 1.2.3, 6.1

[61] Liang Lu, Arnab Ghoshal, and Steve Renals. Regularized subspace Gaussian mixture models for cross-lingual speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 365–370. IEEE, 2011. 4.1

[62] Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. Recurrent neural networks for noise reduction in robust ASR. In *Thir-*

teenth Annual Conference of the International Speech Communication Association (INTERSPEECH). ISCA, 2012. 1, 3.4, 5.2.3, 6.1

[63] Davide Marino and Thomas Hain. An analysis of automatic speech recognition with multiple microphones. In *Twelfth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1281–1284. ISCA, 2011. 1.2.3, 6.1

[64] Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009. 4.4.1

[65] Uwe Meier, Rainer Stiefelhagen, Jie Yang, and Alex Waibel. Towards unrestricted lip reading. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(05): 571–585, 2000. 7.1

[66] Florian Metze, Ankur Gandhe, Yajie Miao, Zaid Sheikh, Yun Wang, Di Xu, Hao Zhang, Jungsuk Kim, Ian Lane, Wonkyum Lee, Sebastian Stuker, and Markus Muller. Semi-supervised training in low-resource ASR and KWS. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4699–4703. IEEE, 2015. 4.3.3

[67] Yajie Miao. Kaldi+PDNN: building DNN-based ASR systems with Kaldi and PDNN. *arXiv preprint arXiv:1401.6984*, 2014. 4.3.3, 5.3.1, 6.4.1

[68] Yajie Miao and Florian Metze. Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training. In *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2237–2241. ISCA, 2013. 1.2.1, 4, 4.1, 4.3.3

[69] Yajie Miao and Florian Metze. Improving language-universal feature extraction with deep maxout and convolutional neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 4, 5, 6.2.2

[70] Yajie Miao and Florian Metze. Distance-aware dnns for robust speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015. 7.3, 7.3.2

[71] Yajie Miao and Florian Metze. On speaker adaptation of long short-term memory recurrent neural networks. In *Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015. 5.7.2

[72] Yajie Miao, Florian Metze, and Shourabh Rawat. Deep maxout networks for low-resource speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 398–403. IEEE, 2013. 1.2.1, 4, 4.1, 4.3.2, 4.3.3, 4.3.3

[73] Yajie Miao, Florian Metze, and Alex Waibel. Learning discriminative basis coefficients for eigenspace mllr unsupervised adaptation. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7927–7931. IEEE, 2013. 5

[74] Yajie Miao, Florian Metze, and Alex Waibel. Subspace mixture model for low-resource speech recognition in cross-lingual settings. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7339–7343. IEEE, 2013. 4.1

[75] Yajie Miao, Lu Jiang, Hao Zhang, and Florian Metze. Improvements to speaker adaptive training of deep neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014. 5.3.1, 6.3.2, 6.4.1, 6.4.2, 6.4.3

[76] Yajie Miao, Hao Zhang, and Florian Metze. Distributed learning of multilingual DNN feature extractors using GPUs. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 4

[77] Yajie Miao, Hao Zhang, and Florian Metze. Towards speaker adaptive training of deep neural network acoustic models. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 5, 5.3.1, 6.3.2, 6.4.3, 7.3

[78] Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015. 8.5

[79] Yajie Miao, Hao Zhang, and Florian Metze. Speaker adaptive training of deep neural network acoustic models using i-vectors. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(11):1938–1949, 2015. 7.3

[80] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002. 2.1.5

[81] Markus Muller and Alex Waibel. Using language adaptive deep neural networks for improved multilingual speech recognition. In *The 21th International Workshop on Spoken Language Translation (IWSLT)*, 2015. 4.1

[82] Markus Muller, Sebastian Stuker, Zaid Sheikh, Florian Metze, and Alex Waibel. Multi-

lingual deep bottle neck features a study on language selection and training techniques. In *The 11th International Workshop on Spoken Language Translation (IWSLT)*, 2014. 4.1

[83] Jiquan Ngiam, Zhenghao Chen, Sonia A Bhaskar, Pang W Koh, and Andrew Y Ng. Sparse filtering. In *Advances in Neural Information Processing Systems*, pages 1125–1133, 2011. 4.3.2

[84] Tsubasa Ochiai, Shigeki Matsuda, Xugang Lu, Chiori Hori, and Shigeru Katagiri. Speaker adaptive training using deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6349–6353. IEEE, 2014. 1.2.2, 5.1.2

[85] Gerasimos Potamianos, Juergen Luettin, and Chalapathy Neti. Hierarchical discriminant features for audio-visual lvcsr. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 165–168. IEEE, 2001. 7.1

[86] Daniel Povey. *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge, 2005. 5.3.1, 5.4.1, 6.4.1

[87] Daniel Povey, Lukáš Burget, Mohit Agarwal, Pinar Akyazi, Feng Kai, Arnab Ghoshal, Ondřej Glembek, Nagendra Goel, Martin Karafiát, Ariya Rastrow, et al. The subspace Gaussian mixture modelA structured model for speech recognition. *Computer Speech & Language*, 25(2):404–439, 2011. 4.1

[88] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. The Kaldi speech recognition toolkit. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 1–4. IEEE, 2011. 5.4.1, 5.7.1, 6.4.1

[89] Daniel Povey, Mirko Hannemann, Gilles Boulianne, Lukáš Burget, Arnab Ghoshal, Miloš Janda, Martin Karafiát, Stefan Kombrink, Petr Motlicek, Yanmin Qian, et al. Generating exact lattices in the wfst framework. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4213–4216. IEEE, 2012. 2.1.5

[90] Ryan Price, ISO Kenichi, and Koichi Shinoda. Speaker adaptation of deep neural networks using a hierarchy of output layers. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014. 5.1.2

[91] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in

speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 2.2.3

[92] Shakti P Rath, Daniel Povey, Karel Veselỳ, and Jan Cernockỳ. Improved feature processing for deep neural networks. In *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 109–113. ISCA, 2013. 1.2.2, 5.1.2

[93] Anthony Rousseau, Paul Deléglise, and Yannick Estève. Ted-lium: an automatic speech recognition dedicated corpus. In *LREC*, pages 125–129, 2012. 5.3.1

[94] Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4153–4156. IEEE, 2012. 3.2

[95] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y Aravkin, and Bhuvana Ramabhadran. Improvements to deep convolutional neural networks for LVCSR. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 315–320. IEEE, 2013. 1, 3.3, 4.3.1, 5.2.3

[96] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for LVCSR. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013. 1, 3.3, 4.3.1, 5.2.3, 6.1, 6.2.2

[97] Hasim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 1, 3.4, 5.2.3, 6.1, 6.2.3

[98] Haşim Sak, Oriol Vinyals, Georg Heigold, Andrew Senior, Erik McDermott, Rajat Monga, and Mark Mao. Sequence discriminative distributed training of long short-term memory recurrent neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014. 1, 3.4, 5.2.3, 6.2.3

[99] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny. Speaker adaptation of neural network acoustic models using i-vectors. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 55–59. IEEE, 2013. 1.2.2, 5.1.2, 5.1.3, 5.5.1, 5.5.1, 5.7, 5.7, 5.11, 5.12

[100] Tanja Schultz and Alex Waibel. Language-independent and language-adaptive acoustic

modeling for speech recognition. *Speech Communication*, 35(1):31–51, 2001. 4.1

[101] Frank Seide, Gang Li, Xie Chen, and Dong Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 24–29. IEEE, 2011. 1, 1.2.2, 5.1.2, 5.7

[102] Michael L Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7398–7402. IEEE, 2013. 6.1

[103] Andrew Senior and Ignacio Lopez-Moreno. Improving DNN speaker independence with i-vector inputs. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 225–229. IEEE, 2014. 1.2.2, 5.1.2, 5.1.3

[104] Sabato Marco Siniscalchi, Jinyu Li, and Chin-Hui Lee. Hermitian polynomial for speaker adaptation of connectionist speech recognition systems. *Audio, Speech, and Language Processing, IEEE Transactions on*, 21(10):2152–2161, 2013. 5.1.2

[105] Garimella SVS Sivaram and Hynek Hermansky. Multilayer perceptron with sparse hidden outputs for phoneme recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5336–5339. IEEE, 2011. 4.3.2

[106] Garimella SVS Sivaram, Sridhar Krishna Nemala, Mounya Elhilali, Trac D Tran, and Hynek Hermansky. Sparse coding for speech recognition. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4346–4349. IEEE, 2010. 4.3.2

[107] Hagen Soltau, George Saon, and Tara N Sainath. Joint training of convolutional and non-convolutional neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5572–5576. IEEE, 2014. 1, 3.3, 4.3.1, 5.2.3, 5.4.2, 6.1, 6.2.2, 6.4.2

[108] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 7.2.2

[109] Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, page 5, 2011. 2.1.4

[110] Pawel Swietojanski and Steve Renais. Sat-lhuc: Speaker adaptive training for learning hidden unit contributions. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5010–5014. IEEE, 2016. 5.1.2

[111] Pawel Swietojanski and Steve Renals. Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014. 5, 5.1.2, 5.5.1, 5.5.1, 5.7

[112] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals. Hybrid acoustic models for distant and multichannel large vocabulary speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 285–290. IEEE, 2013. 1.2.3, 6.1

[113] Michael J Tomlinson, Martin J Russell, and NM Brooke. Integrating audio and visual information to provide highly robust speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 2, pages 821–824. IEEE, 1996. 7.1

[114] Jan Trmal, Jan Zelinka, and Luděk Müller. On speaker adaptive training of artificial neural networks. In *Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2010. 1.2.2, 5.1.2

[115] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11: 3371–3408, 2010. 4.3.3, 5.4.1

[116] Oriol Vinyals, Suman V Ravuri, and Daniel Povey. Revisiting recurrent neural networks for robust asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4085–4088. IEEE, 2012. 6.1

[117] Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3):328–339, 1989. 1, 3.3

[118] Chao Weng, Dong Yu, Shinji Watanabe, and Biing-Hwang Fred Juang. Recurrent deep neural networks for robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5532–5536. IEEE, 2014. 6.1

[119] Jian Xue, Jinyu Li, Dong Yu, Mike Seltzer, and Yifan Gong. Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In

Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 6359–6363. IEEE, 2014. 5.1.2

[120] Shaofei Xue, Ossama Abdel-Hamid, Hui Jiang, Lirong Dai, and Qingfeng Liu. Fast adaptation of deep neural network based on discriminant codes for speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12): 1713–1725, 2014. 1.2.2, 5.1.2

[121] Kaisheng Yao, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012. 1.2.2, 5, 5.1.2

[122] Takuya Yoshioka and Mark JF Gales. Environmentally robust asr front-end for deep neural network acoustic models. *Computer Speech & Language*, 31(1):65–86, 2015. 1.2.3, 6.1

[123] Dong Yu and Michael L Seltzer. Improved bottleneck features using pretrained deep neural networks. In *Twelfth Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 237–240. ISCA, 2011. 3.2, 6.2.1

[124] 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*, pages 4409–4412. IEEE, 2012. 1.1, 4.1

[125] Dong Yu, Michael L Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide. Feature learning in deep neural networks-studies on speech recognition tasks. *arXiv preprint arXiv:1301.3605*, 2013. 3.2, 4.1, 6.2.1

[126] Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7893–7897. IEEE, 2013. 5.1.2

[127] Ben P Yuhas, Moise H Goldstein Jr, and Terrence J Sejnowski. Integration of acoustic and visual speech signals using neural networks. *Communications Magazine, IEEE*, 27(11): 65–71, 1989. 7.1

[128] Puming Zhan and Alex Waibel. Vocal tract length normalization for large vocabulary continuous speech recognition. Technical report, DTIC Document, 1997. 2.1.6

[129] Hao Zhang, Yajie Miao, and Florian Metze. Regularizing DNN acoustic models with

Gaussian stochastic neurons. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4964–4968. IEEE, 2015. 4.1

[130] Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. Improving deep neural network acoustic models using generalized maxout networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 215–219. IEEE, 2014. 4.3.2, 4.4.1

[131] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014. 7.3.3