Incorporating Context Information into Deep Neural Network Acoustic Models

Yajie Miao

April 2015

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. Also, we study a parallelization mechanism to speed up training of LUFEs over multiple GPUs.

As with GMMs, the performance of DNNs degrades when the mismatch between acoustic models and testing speakers exists. For GMMs, speaker adaptive training (SAT) trains the acoustic models in a normalized space so that the resulting models generalize better to unseen testing speakers. In this thesis, we present a novel framework to perform feature-space SAT for DNN models. We leverage i-vectors as speaker representations to project 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. Our SAT-DNN approach is a general framework that can be naturally extended to other deep learning models such as CNNs. Also, the recognition results of SAT-DNNs can be further improved by applying model-space DNN adaptation atop of SAT-DNNs during decoding.

Environment variability such as noise and reverberation poses challenges to ASR. In real-world applications, a critical part of the variability comes from the varying distance between the speakers and microphones. In the final part of this thesis, we improve the robustness of DNN models by incorporating this distance information. The distance descriptors are derived from a DNN model which is trained on a meeting corpus. With these descriptors, our distance-aware DNNs capture the speaker-microphone distance dynamically at the frame level. Furthermore, on the task of transcribing videos, visual features from the video stream provide additional indication about the acoustic environment. For instance, images from the videos may indicate the scenes (offices, cars, etc.) where the speech data have been recorded. We examine the utility of visual features as additional environment descriptors, and study DNN architectures that can fuse context information from heterogeneous sources.

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 [11, 29, 78]. 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 [19, 20]. In addition to DNNs, other deep structures, such as convolutional neural networks (CNNs) [3, 4, 72, 73, 84] and recurrent neural networks (RNNs) [23, 51, 74, 75], have also been exploited as acoustic models. More details about these architectures will be presented in Chapter 2.

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 [33]. This thesis focuses on alleviating the effects of the following three challenges.

- **Data Scarcity.** DNN models differ from the earlier ANN-HMM systems [17] in that there are more hidden layers in the DNN architecture. Therefore, DNN models tend to have much more parameters than GMM models. For example, in [95], 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 [33] 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 [34, 49] has reported the state-of-the-art WER of around 40% on transcribing YouTube videos.

1.2 Proposal Statement

This thesis proposes to solve these challenges by incorporating additional context information into DNN acoustic models. In comparison to GMMs, 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 up easily to 500. This nice property enables DNNs to combine information from different sources. The simplest form of this combination is to concatenate distinct features types (e.g., MFCCs and filterbanks) as DNN inputs. Corresponding to the aforementioned three challenges, our research work can be described as the following three aspects.

1.2.1 Cross-Language DNNs with Language-Universal Feature Extractors

As discussed in the previous section, 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 [27, 32, 55] 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 [32]. 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 [32].

- 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 [32] 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) [22]. Our previous work [57] 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 Adaptive Training of DNN Models using I-vectors

For GMM models, an effective procedure to alleviate the effect of speaker mismatch is speaker adaptation [18, 43]. 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 [45, 93], SI-DNN models are augmented with additional speaker-specific layers which are trained on the adaptation data. Also, [69, 78] achieve adaptation of DNNs by training DNNs using speaker-adaptive features, and [48] 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 [35, 69, 78] or additional speaker-specific information [26, 76, 80] can be treated as a form of SAT. In [92], 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 [64], 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 [92] 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 general 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 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 atop of SAT-DNN models for further improved recognition results.

1.2.3 Robust Speech Recognition with Distance-Aware and Video-Aware DNNs

Environment variability such as noise and reverberation poses special challenges for ASR systems. For robust ASR, previous work [79] has attempted to incorporate noise information into DNN models. In real-world applications, another critical type of variability comes from the varying distance between speakers and microphones. For example, on amateur videos, the distance between the speakers and microphones may vary frequently, even within a single utterance. In this thesis, we enhance the robustness of DNN models by explicitly incorporating the speaker-microphone distance information. Extraction of the distance information relies on a distance-discriminative DNN (DD-DNN) that is trained on an external meeting corpus, with distance types (e.g., distant, close-talking, etc.) of speech files as labels. This DD-DNN can be transferred to our target dataset. At each speech frame, outputs from the DD-DNN's hidden layers are taken as descriptors of distance types. We build distance-aware DNN (DA-DNN) models by appending these distance descriptors to the original input features. By doing this, DA-DNNs capture the distance information dynamically at the frame level.

On the task of transcribing video data, the video stream provides additional indication about the acoustic environment. For instance, images from the videos indicate the scenes in which the speech data have been recorded. Moreover, actions (running, lifting, walking, etc.) performed by the speakers correlate to speaking rates and styles. This thesis investigates the incorporation of different types of visual features into DNN acoustic models. Traditional audio-visual ASR [16, 24] has successfully combined audio and visual features (e.g., lip contours, mouth shapes, etc.) for robust ASR. However, the applicability of these methods is limited by the availability of the mouth-region features, especially on open-domain videos (e.g., YouTube videos). Another limitation of the traditional audio-visual ASR is that the alignment between the speech and video frames is required. In this thesis, we explore open-domain audio-visual ASR by employing video/segment-level visual features that can be extracted readily from real-world videos. Extrac-

tion of the visual features is achieved with models trained on external datasets. We also study approaches to fusing context information from different sources with a single DNN architecture.

1.3 Proposal Organization

The remainder of this proposal is organized as follows. Chapters 3, 4 and 5 correspond to the three proposal statements in Section 1.2 respectively. In each chapter, we present the work we have completed, and the proposed work we plan to work on for the next step.

- Chapter 2 reviews acoustic modeling with DNNs. In addition to hybrid models, we also review how DNNs are used for BNF extraction. Other deep learning architectures CNNs and RNNs are also explained.
- Chapter 3 presents our work on cross-language DNNs with language-universal feature extractors.
- Chapter 4 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 5 improves the robustness of DNN models by incorporating speaker-microphone distance information. On the task of video transcribing, Chapter 5 also describes fusion of visual features into DNN acoustic modeling.
- Chapter 6 shows a time line for the proposed work.

Chapter 2

Review of DNN Acoustic Models

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.

2.1 DNN Models

The architecture of the DNN we use is shown in Figure 2.1. 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 2.1 can be computed as:

$$\mathbf{a}_t^i = \mathbf{W}_i \mathbf{x}_t^i + \mathbf{b}_i \qquad \mathbf{y}_t^i = \sigma(\mathbf{a}_t^i) \qquad 1 \leq i \leq L \tag{2.1}$$

where L is the total number of layers, the weight matrix $\mathbf{W}_i$ connects the *i*-1-th and *i*-th layers, and $\mathbf{b}_i$ is the bias vector of the *i*-th layer. The inputs to the *i*-th layer $\mathbf{x}_t^i$ can be formulated as:

$$\mathbf{x}_t^i = \begin{cases} \mathbf{o}_t & i = 1 \\ \mathbf{y}_t^{i-1} & 1 < i \leq L \end{cases} \tag{2.2}$$

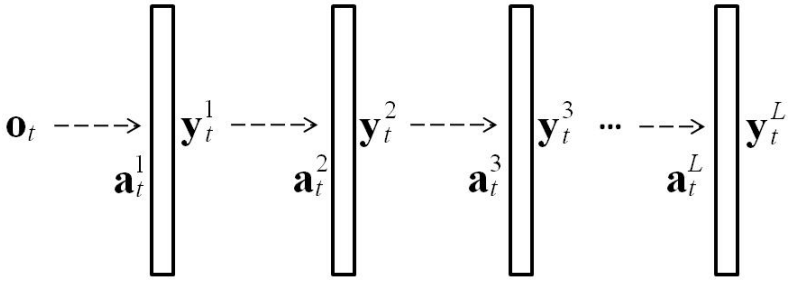

Figure 2.1: Architecture of the DNN model.

where L is the total number of layers, $\mathbf{W}_i$ is the weight matrix connecting 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 $i = L$, $\sigma(x)$ takes the form of the logistic sigmoid and softmax functions respectively.

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

$$\mathbb{L} = -\sum_{t=1}^{T}\sum_{s=1}^{S} \mathbf{g}_t(s) \log \mathbf{y}_t^L(s) \tag{2.3}$$

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:

$$\boldsymbol{\epsilon}_t^L = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_t^L} = \mathbf{y}_t^L - \mathbf{g}_t \tag{2.4}$$

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

$$\boldsymbol{\epsilon}_t^i = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_t^i} = \mathbf{W}_{i+1}^T \boldsymbol{\epsilon}_t^{i+1} \odot \mathbf{y}_t^i \odot (1 - \mathbf{y}_t^i) \tag{2.5}$$

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:

$$p(\mathbf{o}_t|s) \propto \mathbf{y}_t^L(s)/p(s) \tag{2.6}$$

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 [28], 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, [71] 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 [19, 20, 25, 94] has attempted to train the DNN

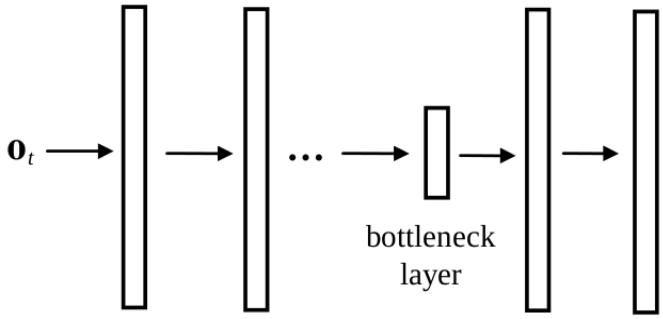

Figure 2.2: Architecture of the DBNF network.

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 [20], we have established the deep BNF (DBNF) architecture for more effective BNF extraction. Our method differs from the previous BNF proposals [25, 94] 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 [96], 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 2.2 depicts the architecture of our DBNF network.

2.2 CNN Models

CNNs have been applied widely in the areas of image processing and computer vision [41]. In the time-delay neural networks (TDNNs) for phoneme recognition [89], 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, 72, 73, 84]. 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 2.3 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

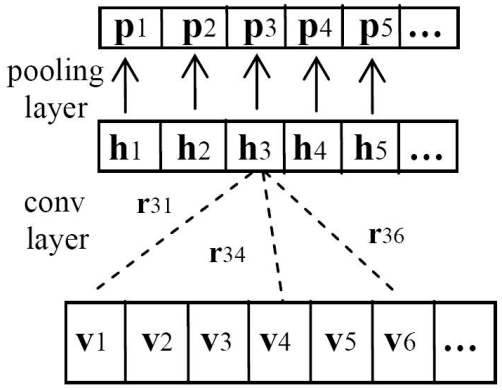

Figure 2.3: Convolution and max-pooling layers in the CNN architecture.

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

$$\mathbf{h}_j = \sigma(\sum_{i=1}^{N} \mathbf{r}_{ji} * \mathbf{v}_i + \mathbf{b}_j) \tag{2.7}$$

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-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.

2.3 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 [23, 51, 74, 75] has studied the application of RNNs to acoustic modeling. In Section 5.4, we propose to perform the extraction of the speaker-microphone distance information with RNNs. Therefore, we also give a brief review of RNNs here.

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

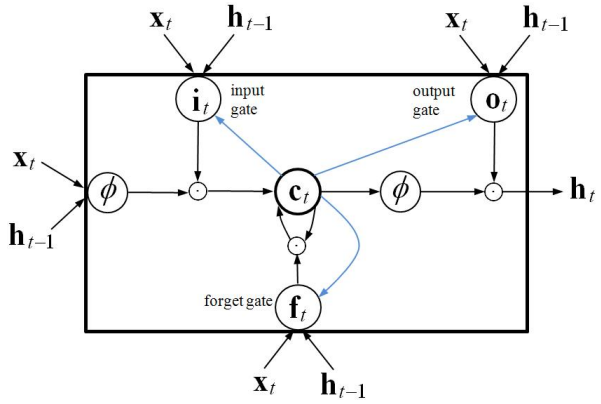

Figure 2.4: A memory block of LSTM.

$\mathbf{H} = (\mathbf{h}_1, ..., \mathbf{h}_T)$ via the following equations:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2.8}$$

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 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 [31] 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 [21]. 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:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{2.9a}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{2.9b}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \phi(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.9c}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_{t-1} + \mathbf{b}_o) \tag{2.9d}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \phi(\mathbf{c}_t) \tag{2.9e}$$

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.

Chapter 3

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 [57], 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 [55, 56, 57, 61]

3.1 Related Work

Previous work has proposed various methods to improve DNNs under low-resource conditions. A potential solution is to build sparse DNNs [95], 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 [30]. Random dropout is observed to perform effectively on phone recognition [12] and LVCSR [55, 99], displaying special benefits when language resources become highly limited. In [57], 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 [77]. For GMM models, the subspace Gaussian mixture models (SGMMs) [66] 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 [8]. Then, these subspace parameters are transferred to a low-resource target language on which only the non-subspace parameters need to be estimated [50]. 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 [59] have been proposed to increase the flexibility of the multilingually-trained subspace parameters on the target language.

In [96], 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 [96] 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, [32, 55] 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 [32], 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 [32], 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.

3.2 Cross-Language DNNs with LUFEs

Our framework is illustrated in Figure 3.1. On the left of Figure 3.1, 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

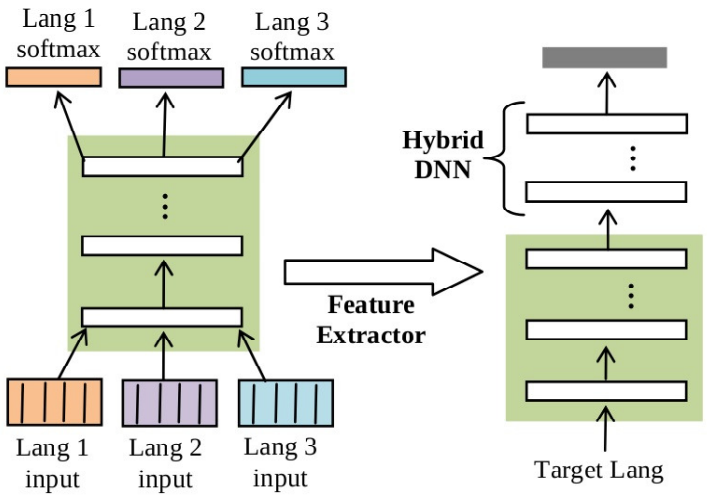

Figure 3.1: Cross-language DNNs with the LUFE.

that we have a new language as shown on the right of Figure 3.1. 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 [32] in that [32] estimates a softmax layer on the new 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.

3.3 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.

3.3.1 LUFEs with Convolutional Networks

As discussed in Section 2.2, 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, 72, 73, 84]. This motivates us to apply CNNs as the building block of LUFEs. Our CNN architecture used in LUFE training follows the previous studies [3, 72, 73, 84]. 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 2.2.

3.3.2 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) [42, 82, 83] 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) [57]. Maxout networks [22] are originally presented as an alternative to DNNs for object recognition. In [57], 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 [100].

Figure 3.2 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 $\mathbf{x}_t$, 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 2.1, we compute the maxout-layer outputs as:

$$\mathbf{y}_t^i(u) = \max_j(\mathbf{a}_t^i(j)) \qquad (u-1) \times g + 1 \leq j \leq u \times g \tag{3.1}$$

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 3.2. 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 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* [63] for each feature type. If the t-th frame has the feature vector $\mathbf{f}_t$, then population sparsity

$$pSparsity = \| \frac{\mathbf{f}_t}{\|\mathbf{f}_t\|_2} \|_1 \tag{3.2}$$

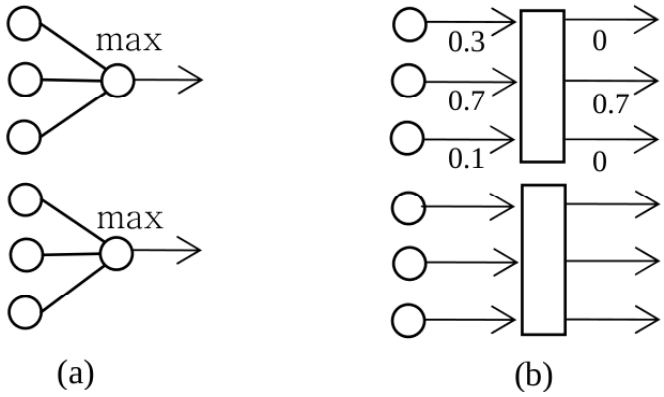

Figure 3.2: An example for (a) maxout layer and (b) non-maximum masking.

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 3.3.3, we experimentally show that this combined feature extractor ends up to be the best LUFE studied in this work.

3.3.3 Experiments

Experimental Setup

Our experiments follow the setup in [57], using the multilingual corpus collected under the IARPA BABEL research program [9, 10, 53]. We aim at improving ASR on the Tagalog (TG, IARPA-babel106-v0.2f) *limited language pack* (LimitedLP). This is a low-resource condition 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 [19, 20, 57] 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 3.1.

On each language, we build the GMM models with the same recipe. An initial maximum

Table 3.1: 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

Table 3.2: WER(%) of monolingual DNN and CNN on the target language.

Models	WER%
Monolingual DNN	70.8
Monolingual CNN	68.2

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 3.1. Training of the DNN and CNN models is performed with the PDNN framework [54].

Inputs of DNNs and CNNs include 11 consecutive frames of 30-dimensional log-scale filter-banks 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) [88] 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 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 2.2 and Figure 2.3. We adopt one-dimensional convolution only along the frequency axis. The size of the filter vectors ($\mathbf{r}_{ji}$ in Equation (2.7)) 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 3.2 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.

Table 3.3: WER(%) of various LUFEs on the target language.

LUFE Models	WER%	pSparsity
DNN-LUFE	69.6	21.3
Method in [32]	70.1	21.3
CNN-LUFE	67.1	20.4
DMN-LUFE	67.5	17.7
CNN-DMN-LUFE	65.9	16.6

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 3.3 shows the results of the hybrid models when various LUFEs are applied. We can see from Table 3.2 that cross-language models based on LUFEs give consistently better results than the monolingual DNN. On the same setup, we also show the WER obtained by the method described in [32], 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 [57], during training of DMNs, the dropout technique [30] is applied in order to prevent overfitting. We impose dropout on each hidden layer by following the implementation described in [55]. 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 3.3 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 3.3 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.

3.4 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 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.

3.4.1 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 [52, 100]. 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 3.3, 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 3.2. 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 [13, 27], 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 examples on each GPU. Because of this, delay resulting from model averaging ends up to be a tiny fraction of the entire training time.

3.4.2 Experiments

As with Section 3.3.3, 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

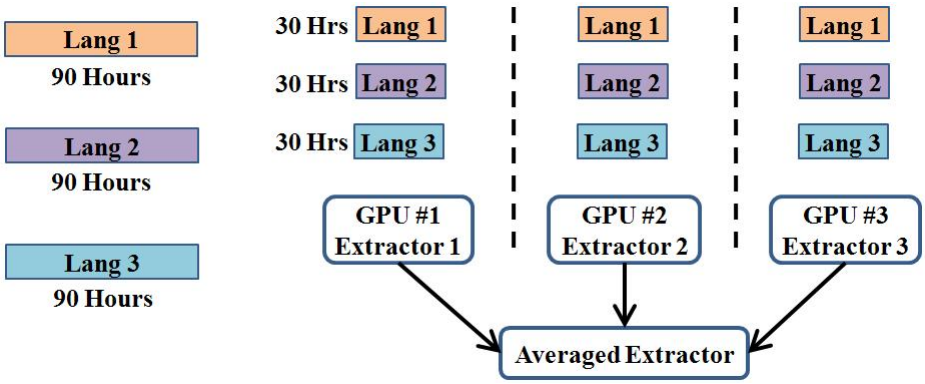

Figure 3.3: The DistModel distributed learning strategy for LUFEs.

using the same recipes as described in Section 3.3.3. 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 3.3.3. 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 3.4 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 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 3.5 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 3.4. This reveals that our proposed DistModel strategy is particularly suited for the task of LUFE learning.

Table 3.4: 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 3.5: 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

3.5 Proposed Work

Our future work for this chapter mainly focuses on more complete and conclusive experiments. Here are two aspects we will work on for the next step.

- **Consistent Experimental Setup.** Our experiments in this chapter have relied on inconsistent setups. For example, in Section 3.3, the source languages are the LimitedLP (10-hour) sets of three languages. In comparison, Section 3.4 uses the FullLP (around 80-hour) sets of the same languages as the source languages. Also, even with the same target language (Tagalog LimitedLP), we have used different scoring setups which make our numbers in the two sections incomparable. Our future work will unify these setups, and present results/analysis on a consistent basis.
- **Complete Evaluation.** In this chapter, our experiments have taken the LimitedLP set of Tagalog as the target language. In our future work, we will evaluate our cross-language DNN framework by taking the FullLP set as the target language. This will give us insight on how our framework performs under various levels of data availability on the target language.

3.6 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. Our next step focuses on converging to a unified experimental setup for more complete and conclusive experiments.

Chapter 4

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 [18, 43, 58]. Recently, a large amount of work has been dedicated to speaker adaptation of DNNs [48, 86, 93]. 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 [56, 62]

4.1 Related Work

This section reviews previous work that is related to our work. These previous work is divided into two aspects: speaker adaptation of DNNs and extraction of i-vectors.

4.1.1 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 [45, 78], 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 [45]. Yao *et al.* [93] 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.* [91] perform adaptation by learning SD matrices that are inserted between the decomposed weight matrices. In the recently proposed learning hidden unit contributions (LHUC) approach [86], 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.

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 [69, 78]. Another way of normalizing speaker variability is to augment the input features with additional speaker-specific information. Speaker i-vectors [14, 15] have been explored for this purpose. For example, Saon *et al.* [76] append i-vectors to the original features at each speech frame. The effectiveness of incorporating i-vectors is further verified in [26, 80]. I-vector based adaptation is further developed in [38, 47] 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, [49] 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.* [97] 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.* [68] 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 [81], 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 [35, 69, 78] or additional speaker-specific information [26, 76, 80] can be treated as a form of SAT. In [92], 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 [64], 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 [92] 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.

4.1.2 I-Vector Extraction

Recently, the application of the i-vector paradigm has resulted in significant advancement in the speaker verification community [14, 15]. The i-vector approach differs from the earlier JFA [40] 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:

$$\mathbf{o}_t \sim \sum_{k=1}^{K} r_t(k) N(\boldsymbol{\mu}_k + \mathbf{T}_k \mathbf{w}_s, \boldsymbol{\Sigma}_k) \tag{4.1}$$

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 [14, 15] for more details.

Given a collection of speech segments, the UBM model can be trained using the standard maximum likelihood estimation (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 [26, 37, 76, 80] 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).

4.2 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.

4.2.1 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 4.1. Suppose that an i-vector has been extracted for each speaker as described in Section 4.1.2 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 4.1) 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 4.1) is learned to take i-vectors as inputs and generate speaker-specific linear shifts to the original DNN feature vectors. In Figure 4.1, 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 values attached to the adaptation network can be computed as:

$$\mathbf{a}_s^i = \mathbf{W}_i \mathbf{x}_s^i + \mathbf{b}_i \qquad \mathbf{y}_s^i = \phi(\mathbf{a}_s^i) \qquad -I \leq i \leq -1 \tag{4.2}$$

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:

$$\mathbf{x}_s^i = \begin{cases} \mathbf{i}_s & i = -I \\ \mathbf{y}_s^{i-1} & -I < i \leq -1 \end{cases} \tag{4.3}$$

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*:

$$\mathbf{z}_t = \mathbf{o}_t \oplus \mathbf{y}_s^{-1} \tag{4.4}$$

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.

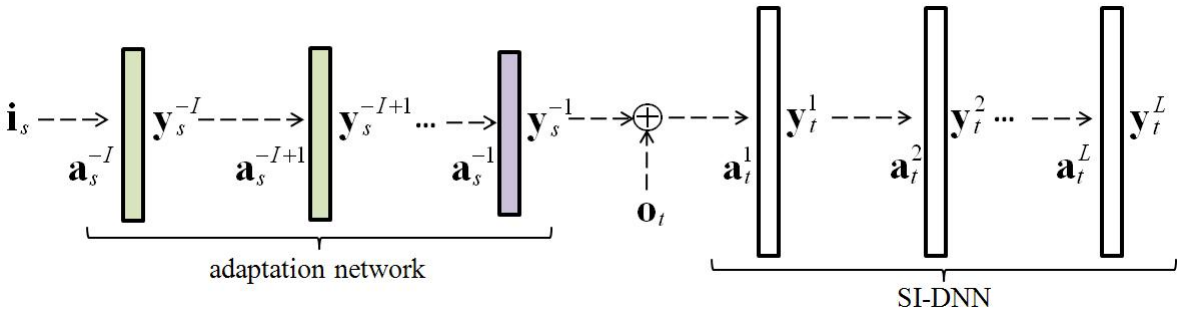

Figure 4.1: Architecture of the SAT-DNN model.

Second, the number of units at the output layer (i.e., the dimension of $\mathbf{y}_s^{-1}$) has to equal the dimension 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.

4.2.2 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 **??**, i.e., the negative 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:

$$\frac{\partial \mathbb{L}}{\partial \mathbf{z}_t} = \mathbf{W}_1^T \boldsymbol{\epsilon}_t^1 \tag{4.5}$$

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:

$$\boldsymbol{\epsilon}_s^{-1} = \frac{\partial \mathbb{L}}{\partial \mathbf{a}_s^{-1}} = \frac{\partial \mathbb{L}}{\partial \mathbf{y}_s^{-1}} = \frac{\partial \mathbb{L}}{\partial \mathbf{z}_t} \tag{4.6}$$

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.

4.2.3 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 negative 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, 72, 73, 84] and RNNs [23, 51, 74, 75]. In our experiments, we will demonstrate the extension of SAT-DNN to various model and feature types.

4.2.4 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 $\mathbf{z}_t$.

As reviewed in Section 4.1.1, most of the existing speaker adaptation methods require multiple passes of decoding for unsupervised adaptation. The first-pass decoding generates initial hypotheses from which we derive the frame-level supervision. In comparison, because i-vector extraction uses only the speech data, adaptation of SAT-DNN models requires one single pass of

decoding. Furthermore, with no fine-tuning on the adaptation data, adaptation of SAT-DNNs is insensitive to hypotheses errors. This is especially an advantage when the first-pass hypotheses have high WERs. Therefore, using SAT-DNN model, we can achieve both efficient and robust unsupervised adaptation.

4.3 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.

4.3.1 Experimental Setup

Dataset

Our experiments use the benchmark TEDLIUM dataset [70] 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 MFCC+$\triangle$+$\triangle\triangle$. Then 7 frames of MFCCs are spliced together and projected down to 40 dimensions with 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 [65] 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 [54]. 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.

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

Table 4.1: 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

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 filterbank 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 [60, 62].

4.3.2 Basic Results

Table 4.1 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 4.1 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.

Table 4.2: 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

4.3.3 Bridging I-vector Extraction with DNN Training

I-vector extraction in Section 4.1.2 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 [44] has studied approaches to leveraging DNN models in i-vector extraction. In [44], 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 [44].

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.

Following our DBNF architecture [19, 20], 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 4.2, 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.

4.3.4 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. 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

Table 4.3: 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

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 4.3 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.

4.4 Extension to BNFs and CNNs

In Section 4.2, 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.

4.4.1 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 [19, 20] 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 [88]. 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 [67]. 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 [65].

Applying SAT to bottleneck feature extraction is straightforward. We simply replace the DNN model in Figure 4.1 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 4.4 compares the performance of the

Table 4.4: 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

Table 4.5: 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

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.

4.4.2 Extension to CNNs

Section 4.2 claims the applicability of our SAT technique to CNNs, and we experimentally confirm this in this subsection. Our CNN architecture follows [84], 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 4.5. 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 4.5.

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 4.3.3. 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

Table 4.6: WERs(%) of the SI-CNN and SAT-CNN models.

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

4.6 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.

4.5 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.

4.5.1 Comparing SAT and Speaker Adaptation

Past work [76] 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 [86]. 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:

$$\mathbf{y}_t^i = \psi(\mathbf{r}_s^i) \odot \sigma(\mathbf{W}_i \mathbf{x}_t^i + \mathbf{b}_i) \tag{4.7}$$

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 [86]. 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 [76] and [86]. 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 4.3. 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

Table 4.7: 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[76]	19.5	2.5
DNN+LHUC[86]	18.3	8.5

Table 4.8: 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

frame-level supervision. Training of the SD parameters goes through 3 epochs with the constant learning rate of 0.8. Table 4.7 shows the results of the DNN models adapted with 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.

4.5.2 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 4.8 provides a summary of the 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.

4.6 Proposed Work

For this chapter, we plan to work on the following two aspects for the next step.

- **Evaluations on larger datasets.** A major competitor of our SAT approach is the DNN+I-vector [76] speaker adaptation method based on i-vectors. In [76], the authors use the Switchboard dataset (300 hours) and report over 10% relative improvement achieved by DNN+I-vector. In comparison, our experiments, which use a smaller dataset (118 hours), report only 2.5% relative improvement from DNN+I-vector. Therefore, we plan to test the SAT-DNN approach on the complete Switchboard dataset. This enables us to compare SAT-DNN and DNN+I-vector more completely. Moreover, we can analyze how SAT-DNN performs with respect to different levels of data availability.
- **Acceleration of SAT-DNN training.** Although showing promising improvement, application of SAT-DNNs increases the training cost dramatically. This is because building of SAT-DNNs is an additional step that has to be based on the training of SI-DNN models. Therefore, it is worthwhile to study how to accelerate the training of SAT-DNN models. Our focus will be on designing data selection strategies for the SAT-DNN training. This is because 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. In our future work, we will explore acceleration (data selection) strategies which give us good speed-up while minimizing WER loss.

4.7 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. For our future work, we propose to evaluate our SAT-DNN approach on the larger Switchboard dataset and study better strategies to accelerate training of SAT-DNN models.

Chapter 5

Robust Speech Recognition with Distance-Aware and Video-Aware DNNs

5.1 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 [46]. 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 [33] that as with GMMs, the performance of DNNs drops significantly as the SNR decreases. Various attempts [47, 51, 79, 90] have been made to build noise-robust DNN models. For example, [79] proposes different methods, including multi-condition training and dropout training, to improve DNN models under low SNRs. In [51, 90], 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 [87]. A number of techniques have been presented for improved DNN models on farfield speech. Although showing nice gains, these methods have the limitation that they only deal with constantly distant speech. That is, the speaker-microphone distance keeps unchanged during the course of recording. However, in many real-world scenarios, this distance is quite dynamic. For instance, on amateur videos, it is common to see that the speaker walks around when talking to a farfield microphone. 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 speaker-microphone distance 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 [16, 24]. 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.

5.2 Distance-Aware DNNs

Building of DA-DNNs starts with the modeling of speaker-microphone distance at the frame level. We perform the extraction of the distance information with an additional distance-discriminative DNN (DD-DNN). A DD-DNN is effectively a DNN with a narrow bottleneck layer in it. The DD-DNN is trained on a dataset where the distance type (close-talking, distant, etc.) of each speech file is known. Training of the DD-DNN is to classify each speech frame to the distance types, instead of CD phonetic states. After the training is finished, the outputs from the bottleneck layer of the DD-DNN is inherently discriminative with respect to distance types of speech frames. Then, this DD-DNN can be transferred to our target dataset. The feature vector corresponding to each speech frame is fed into the DD-DNN. Outputs from the DD-DNN's bottleneck layer are treated as speaker-microphone distance descriptors, and are appended to the original DNN input features. By doing this, the DNN model on the target domain captures the distance information dynamically at the frame level.

In our experiments, the DD-DNN is trained on the ICSI meeting corpus [36]. 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 different 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 purely distance types, the combinations of speakers and distance types are taken as the labels, which totally gives us 2311 classes. Our DD-DNN has 5 hidden layers in which the fourth layer is the bottleneck layer with 100 units. Each of the other hidden layers has 1024 units.

Our target domain is a video transcribing task. We download a collection of around 4k English videos from online archives such as Youku.com, Tudou.com, YouTube.com and CreativeCommons.org [98]. These videos are intended for expertise sharing on specific tasks (e.g., oil change and sandwich making), and have an average duration of 90 seconds. For each video, the manual transcriptions have been provided by the uploading user. We take several steps to convert the collected data into an applicable ASR training corpus. These steps include cleaning and normalizing transcripts, down-sampling the audio track, adding new words into the dictionary, etc. Time markers for each utterance are obtained via forced alignment with the raw closed captions and our existing broadcast news recognizer. This finally gives us 94 hours of speech data, out of which 90 hours are selected for training and 4 hours for testing.

On this video-transcribing corpus, we build the GMM and normal DNN models by following the similar procedures as described in Section 4.3. For GMMs, we build the SI model with the LDA+MLLT front-end. The SAT model is constructed which has 3891 triphone states. The DNN

Table 5.1: Performance of DNN and DA-DNN for video transcribing.

Models	WER(%)
DNN (baseline)	23.4
DA-DNN	22.1

model has 6 hidden layers each of which has 1024 units. Inputs of the DNN model consists of 11 neighbouring frames of log-scale filterbanks. The labels for fine-tuning the DNN are generated from the SAT model. To rule out the impact of pre-training on our performance comparisons, the parameters of the DNN model are randomly initialized. Table 5.1 shows the performance of the DNN and DA-DNN on the 4-hour testing set. Compared with the baseline DNN, we can see that the DA-DNN model achieves 1.3% absolute WER improvement, that is, 5.6% relatively.

5.3 Video-Aware DNNs

In addition to the audio stream, the video stream provides additional indication about the acoustic environment. For instance, images from the videos indicate the scenes in which the speech data have been recorded. Moreover, actions (running, lifting, walking, etc.) performed by the speakers correlate to speaking rates and styles. This thesis investigates the incorporation of different types of visual features into DNN acoustic models. Unlike previous work on audio-visual ASR [16, 24], we study video/segment-level visual features that can be obtained from real-world videos.

We firstly study speaker attributes that can be deduced 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) that returns 3 attributes: age, gender, and race. The value of age is continuous, while gender and race have categorical values. 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. The gender classification result is converted into a 2-dimensional vector whose binary elements denote male and female respectively. Similarly, a 3-dimensional vector is employed to represent the 3 possible values of race: White, Black and Asian. 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]. 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 each of the sub-vectors uniformly, e.g., [0.5, 0.5] for gender and [0.33 0.33 0.33] for race.

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 [85]. UCF101 consists of realistic action

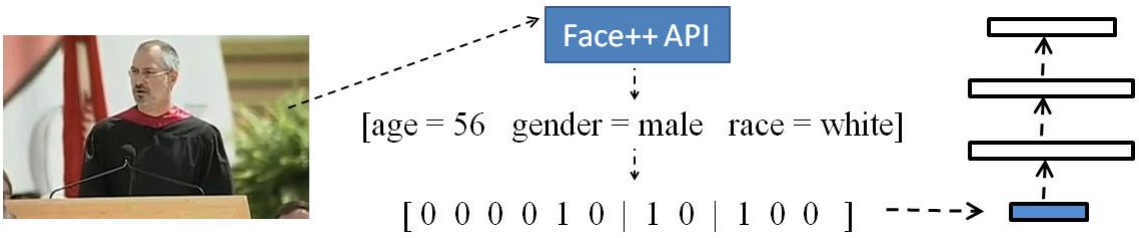

Figure 5.1: Incorporation of speaker attributes into DNN.

Table 5.2: Performance of DNN and DA-DNN for video transcribing.

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

videos collected from YouTube, having 101 action categories. After performing action recognition, on each video segment, we get a 101-dimensional vector containing the probabilities over the 101 action classes. This vector is taken as additional information at the utterance level, and is appended to each of the original feature vectors from this utterance. Table 5.2 presents the performance of video-aware DNN (VA-DNN) models when the speaker attributes and speaker actions are the additional information. From Table 5.2, we can see that incorporating both types of visual features results in consistent improvement, although the improvement is not as significant as that of incorporating the distance information.

5.4 Proposed Work

The work described in this chapter is still ongoing. We plan to extend our work from the following aspects.

5.4.1 Better Extraction of Distance Information

The effectiveness of our DA-DNN models is largely determined by the extraction of distance information, that is, the quality of the DD-DNN model. Currently, this distance information is generated with a DNN model that has a bottleneck layer. For better extraction of the distance information, we propose to investigate the more advanced CNNs and LSTM-RNNs architectures.

- We plan to study the utility of CNNs in the extraction of the distance descriptors. CNNs have shown superior performance than DNNs for acoustic modeling. Compared with DNNs, CNNs have the special advantage of normalizing local variation along the frequency dimension. In highly challenging acoustic conditions, the quality of the distance descriptors might be undermined by spectral variations such as noise and reverberation. Applying the CNN enables us to extract distance descriptors that are robust to variability from spectral distortion.

- Both DNNs and CNNs can only model the limited temporal dependency within the fixed-size context window. However, the variation of the speaker-microphone distance inherently displays complex temporal dynamics. The LSTM-RNNs architecture overcomes some weaknesses of the conventional RNNs, and is particularly suitable for modeling long-range temporal dependency. We propose to apply LSTM-RNNs as our distance-information extractor. By considering the long-term contextual history, such an extractor potentially generates more accurate distance descriptors.

5.4.2 Investigation of other Video Features

Exploiting speaker attributes and actions gives us flexibility to conduct audio-visual ASR on open-domain videos. In our proposed work, we plan to expand the types of visual features that are used in our vidoe-aware DNN models.

- Our preliminary work in Section 5.3 uses the UCF101 dataset for speaker actions. In addition to UCF101, larger-scale action datasets [39] have been collected and released in recent years. In our proposed work, we will study the effectiveness of these additional action datasets in the incorporation of speaker action information. Action classifiers trained on these datasets potentially generate action information that has wider coverage over real-world speaker actions. Moreover, such a study gives us the insight regarding how the choice of the external action datasets impacts the performance of video-aware DNNs.
- The application of speaker attributes and actions still imposes limitations on audio-visual ASR. For example, the underlying assumption for incorporating these two types of features is that the speaker shows up in the video. However, it is not always such a case in real-world videos. To further relax the constraints, we will explore the incorporation of feature representations extracted directly from the images. Specifically, we take the raw images as the inputs and a deep CNN model as the feature extractor. This CNN model is trained on some external image classification (object recognition) datasets such as ImageNet. Outputs from some higher layers of the CNN will be treated as the additional features. In our proposed work, we plan to study the integration of these "wild" image features into the DNN acoustic models.

5.4.3 Effective Fusion of Information

In Section 5.3, we have employed the feature-concatenation for incorporation of various additional features. However, both the distance and the video descriptors are high-level features, whereas the DNN inputs are normally low-level, raw acoustic features (e.g., filterbanks). Moreover, these distance/video descriptors may have different value ranges than the acoustic features. Therefore, the simple concatenation at the front-end may not be an optimal manner for information fusion. In our proposed work, we will investigate more effective strategies for information fusion. In particular, two possible solutions will be studied.

- The first solution is motivated by our SAT-DNN framework presented in Chapter 4. For the case of distance-aware acoustic modeling, we can naturally achieve distance adaptive training (DAT) with the technique from Chapter 4. Specifically, we replace the speaker

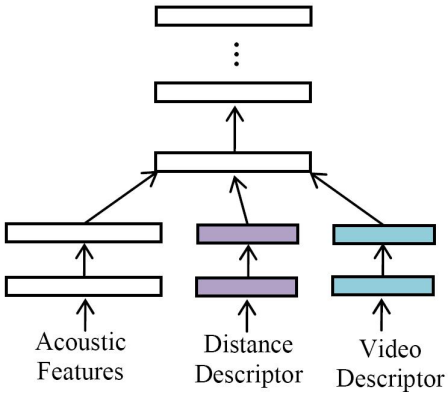

Figure 5.2: A DNN architecture for fusing features from different sources.

i-vectors with the distance descriptors. As with in SAT-DNNs, building of DAT-DNNs learns the adaptation network to derive the distance-normalized feature space. The feature shifts generated by the adaptation network are on the frame level rather than on the speaker level. Finally, parameters of the initial DNN are updated in the normalized space. This adaptively trained DNN becomes independent of specific distance conditions, and thus generalizes better to unseen distance variability. The same technique also applies to video-aware DNNs. The additional visual features are used to replace or enrich the i-vectors representations.

- The second solution is a DNN architecture that has multiple towers at the bottom layers, as depicted in Figure 5.2. At each speech frame, the low-layer towers take individual feature types such as the original acoustic feature, the speaker attributes, the speaker actions, etc. The outputs from these towers are concatenated at a certain higher layer. The non-linear transformations in each tower help reduce the incompatibility across the different feature types. Additinally, this architecture is flexible in that it does not depend on the type of the model that has been built on the original speech features. It can be applied both to DNNs and also to CNNs. When applied to CNNs, the tower that models the original acoustic features will contain convolution layers. In comparison, the simple feature-concatenation method is not applicable to CNNs.

Chapter 6

Time Line

6.1 To-do List

6.1.1 Chapter 3

- **Task 1.** Unify the experimental setups for LUFE learning and present results/analysis on a consistent basis.
- **Task 2.** Evaluate our cross-language DNN framework by taking the FullLP set as the target language.

6.1.2 Chapter 4

- **Task 1.** Test the SAT-DNN approach on the complete Switchboard dataset; analyze how SAT-DNN performs with respect to different levels of data availability.
- **Task 2.** Explore better acceleration (data selection) strategies which speed up training of SAT-DNN models and at the same time minimize WER loss.

6.1.3 Chapter 5

- **Task 1.** Investigate the application of CNNs and LSTM RNNs to extracting distance information for distance-aware DNNs.
- **Task 2.** For video-aware DNNs, explore the utility of other types of visual features in assisting acoustic modeling. For example, we plan to incorporate features that are extracted by CNNs from the raw images.
- **Task 3.** Study more powerful models/DNN architectures which can effectively integrate context information from different sources.

Table 6.1: Proposed Time Line.

Tasks	Time
Chapter 3 Task 1 and 2	05/2015
Chapter 4 Task 1 and 2	06/2015 to 07/2015
Chapter 5 Task 1	08/2015 to 09/2015
Chapter 5 Task 2	10/2015 to 11/2015
Chapter 5 Task 3	11/2015 to 02/2016
Thesis Writing	02/2016 to 04/2016
Thesis Defense	04/2016

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, 4.1.1

[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, 4.1.1

[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, 2.2, 3.3.1, 4.2.3

[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, 2.2, 3.3.1, 4.2.3

[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, 4

[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, 4

[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. 2.3

[8] 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. 3.1

[9] 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. 3.3.3

[10] 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. 3.3.3

[11] 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

[12] 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. 3.1

[13] 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. 3.4.1

[14] 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. 4.1.1, 4.1.2, 4.1.2

[15] 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. 4.1.1, 4.1.2, 4.1.2

[16] Stéphane Dupont and Juergen Luettin. Audio-visual speech modeling for continuous speech recognition. *Multimedia, IEEE Transactions on*, 2(3):141–151, 2000. 1.2.3, 5.1, 5.3

[17] Horacio Franco, Michael Cohen, Nelson Morgan, David Rumelhart, and Victor Abrash. Context-dependent connectionist probability estimation in a hybrid hidden markov model-neural net speech recognition system. *Computer Speech & Language*, 8(3):211–222, 1994. 1.1

[18] Mark JF Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer speech & language*, 12(2):75–98, 1998. 1.2.2, 4

[19] 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, 2.1, 3.3.3, 4.3.3, 4.4.1

[20] 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, 2.1,

3.3.3, 4.3.3, 4.4.1

[21] 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. 2.3

[22] 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, 3.3.2

[23] Alex Graves, A-R 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, 2.3, 4.2.3

[24] 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. 1.2.3, 5.1, 5.3

[25] 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. 2.1

[26] 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, 4.1.1, 4.1.2

[27] 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, 3.4.1

[28] 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. 2.1

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

[30] 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. 3.1, 3.3.3

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2.3

[32] 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, 3.1, 3.2, 3.3, 3.3.3

[33] 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, 5.1

[34] Navdeep Jaitly, Patrick Nguyen, Andrew W Senior, and Vincent Vanhoucke. Application 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

[35] Trmal Jan, Zelinka Jan, 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, 4.1.1

[36] 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. 5.2

[37] 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. 4.1.2

[38] 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. 4.1.1

[39] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1725–1732. IEEE, 2014. 5.4.2

[40] 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. 4.1.2

[41] 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. 2.2

[42] 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. 3.3.2

[43] 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, 4

[44] 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. 4.3.3

[45] 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, 4.1.1

[46] 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. 5.1

[47] 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. 4.1.1, 5.1

[48] 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, 4

[49] 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, 4.1.1

[50] 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. 3.1

[51] 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 *Thirteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2012. 1, 2.3, 4.2.3, 5.1

[52] 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. 3.4.1

[53] 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. 3.3.3

[54] Yajie Miao. Kaldi+pdnn: building dnn-based asr systems with kaldi and pdnn. *arXiv preprint arXiv:1401.6984*, 2014. 3.3.3, 4.3.1

[55] 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, 3, 3.1, 3.3.3

[56] 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. 3, 4

[57] 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, 3, 3.1, 3.3.2, 3.3.3, 3.3.3

[58] 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. 4

[59] 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. 3.1

[60] 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. 4.3.1

[61] 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. 3

[62] 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. 4, 4.3.1

[63] 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. 3.3.2

[64] 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, 4.1.1

[65] Daniel Povey. *Discriminative training for large vocabulary speech recognition*. PhD thesis, University of Cambridge, 2005. 4.3.1, 4.4.1

[66] 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. 3.1

[67] 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. 4.4.1

[68] 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. 4.1.1

[69] 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, 4.1.1

[70] Anthony Rousseau, Paul Deléglise, and Yannick Estève. Ted-lium: an automatic speech recognition dedicated corpus. In *LREC*, pages 125–129, 2012. 4.3.1

[71] 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. 2.1

[72] Tara N Sainath, Brian Kingsbury, A-r 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, 2.2, 3.3.1, 4.2.3

[73] Tara N Sainath, A-r 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, 2.2, 3.3.1, 4.2.3

[74] 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, 2.3, 4.2.3

[75] 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, 2.3, 4.2.3

[76] 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, 4.1.1, 4.1.2, 4.5.1, 4.5.1, 4.7, 4.6

[77] Tanja Schultz and Alex Waibel. Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1):31–51, 2001. 3.1

[78] 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, 4.1.1

[79] 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. 1.2.3, 5.1

[80] 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, 4.1.1, 4.1.2

[81] 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. 4.1.1

[82] 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. 3.3.2

[83] 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. 3.3.2

[84] 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, 2.2, 3.3.1, 4.2.3, 4.4.2

[85] 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. 5.3

[86] 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. 4, 4.1.1, 4.5.1, 4.5.1, 4.7

[87] 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. 5.1

[88] 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. 3.3.3, 4.4.1

[89] 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. 2.2

[90] 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. 5.1

[91] 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. 4.1.1

[92] 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, 4.1.1

[93] 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, 4, 4.1.1

[94] 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. 2.1

[95] 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, 3.1

[96] 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. 2.1, 3.1

[97] 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. 4.1.1

[98] Shoou-I Yu, Lu Jiang, and Alexander Hauptmann. Instructional videos for unsupervised harvesting and learning of action examples. In *Proceedings of the ACM International Conference on Multimedia*, pages 825–828. ACM, 2014. 5.2

[99] 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. 3.1

[100] 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. 3.3.2, 3.4.1