Low latency modeling of temporal contexts for speech recognition

by

Vijayaditya Peddinti

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

October, 2017

Abstract

This thesis focuses on the development of neural network acoustic models for large vocabulary continuous speech recognition (LVCSR) to satisfy the design goals of low latency and low computational complexity. Low latency enables online speech recognition; and low computational complexity helps reduce the computational cost both during training and inference.

Long span sequential dependencies and sequential distortions in the input vector sequence are a major challenge in acoustic modeling. Recurrent neural networks have been shown to effectively model these dependencies. Specifically, bidirectional long short term memory (BLSTM) networks, provide state-of-the-art performance across several LVCSR tasks. However the deployment of bidirectional models for online LVCSR is non-trivial due to their large latency; and unidirectional LSTM models are typically preferred.

In this thesis we explore the use of hierarchical temporal convolution to model long span temporal dependencies. We propose a sub-sampled variant of these temporal convolution neural networks, termed time-delay neural networks (TDNNs). These sub-sampled TDNNs reduce the computation complexity by $\sim 5x$, compared to TDNNs, during frame randomized pre-training. These models are shown to be effective in modeling long-span temporal contexts, however there is a performance gap compared to (B)LSTMs.

As recent advancements in acoustic model training have eliminated the need for frame randomized pre-training we modify the TDNN architecture to use higher sampling rates, as the increased computation can be amortized over the sequence. These variants of sub-sampled TDNNs provide performance superior to unidirectional LSTM networks, while also affording a lower real time factor (RTF) during inference. However we show that the BLSTM models outperform both the TDNN and LSTM models.

We propose a hybrid architecture interleaving temporal convolution and LSTM layers which is shown to outperform the BLSTM models. Further we improve these BLSTM models by using higher frame rates at lower layers and show that the proposed TDNN-LSTM model performs similar to these superior BLSTM models, while reducing the overall latency to 200 ms.

Finally we describe an online system for reverberation robust ASR, using the above described models in conjunction with other data augmentation techniques like reverberation simulation, which simulates far-field environments, and volume perturbation, which helps tackle volume variation even without gain normalization.

Readers: Daniel Povey and Sanjeev Khudanpur

Acknowledgments

My advisors

I would like to thank my advisors Dan Povey and Sanjeev Khudanpur, for giving me the freedom to explore topics of my interest, helping me understand problems of significance in the research community and having the patience to answer my naive questions. Dan's hand-held guidance during ASR system building and the oppurtunity to observe him in close quarters when he made system design decisions helped me understand the importance of various practical aspects of ASR systems. I am also thankful to Sanjeev for keeping alive the spirit of scientific inquiry during my investigation, and for reminding me the importance of long standing impact of research contributions.

I would also like thank Hynek Hermansky for his guidance during the initial years of my PhD and sharing his passion for and knowledge of bio-inspired speech processing techniques.

Kaldi community

I would like to thank the Kaldi developer and user communities for ensuring that *reproducible research* is within the grasps of the speech recognition community; and for helping in the quick dissemination of the techniques proposed in this thesis. I would like to thank Dan, once again, and Jan Trmal for keeping these communities alive and thriving with their support. The understanding of the importance of *open source software* that Kaldi has instilled in me will never be forgotten.

My peers

The power of collaboration was exemplified by my peer group at the Center for Language and Speech Processing. While Kaldi provided a common framework for us to collaborate, it was the spirit of sharing and team work among this group which helped us compete with industrial labs both during the *ASpIRE* challenge and *Babel* program. I would like to thank Guoguo Chen, Pegah Ghahremani, Vimal Manohar, Xiaohui Zhang, David Snyder, Matthew Wiesner, Matthew Maciejewski, Chunxi Liu, Minhua Wu and Hainan Xu.

I would also thank the folks at Brno University of Technology for making my research exchange trips to Czech Republic fun and educational. In particular I would like to thank Lukas Burget for his insightful comments about my experiments and Karel Vesely for his discussions. I would also like to thank Martin Karafiat, Olda Plchot and Jan “Honza” Cernocky for making me feel at home.

I would like to thank Thomas Schatz and Emmanuel Dupoux for their valuable collaboration, and making me feel welcome during my visit to Ecole Normale Superieure, Paris.

I would like to thank Jan Trmal and Najim Dehak, for their wonderful company which helped me keep my spirits up. I would like thank Harish Mallidi, Pascal Clark, Keith Levin, ACKNOWLEDGMENTS

Phani Shankar and Thomas Janu for the long drawn technical and social discussions in the company of beer.

I would also like to thank Samuel Thomas, Sriram Ganapathy, Sivaram Garimella and Keith Kintzley for their guidance during the initial years of my PhD.

A big thanks to Guoguo Chen for his wonderful company throughout my PhD.

My mentors

I would like to thank Mike Seltzer, Microsoft Research, and Tara Sainath, Google Research, for their wonderful guidance during my internships. These internships helped me shape my thesis to a great extent.

My family

I would like to thank my mom and dad, the people who provided their support throughout my life and kept me motivated to work towards my PhD. I would like to thank my brother for his guidance and support during the most critical dilemmas. Finally I would like to thank my wife for pushing me out of the bed on weekends and ensuring that I write my thesis and to my puppy Oreo for keeping me energized with his kisses.

Dedication

This thesis is dedicated to my lovely puppy Oreo, and beloved wife Keerthi.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automatic speech recognition is popularly formulated as a sequence recognition problem. In this formulation the speech signal, represented as a sequence of pressure variations recorded at the microphone $\mathbf{X} = x[1]...x[n]$, is translated to a sequence of words $\mathbf{W} = w[1]....w[m]$. In the statistical formulation this is represented as

$$\hat{\mathbf{W}} = arg\ \underset{\mathbf{W}}{max}\ P(\mathbf{W}|\mathbf{X}) \tag{1.1}$$

which requires the estimation of the posterior probability distribution $P(\mathbf{W}|\mathbf{X})$. The Bayes' rule can be used to decompose the probability distribution as

$$P(\mathbf{W}|\mathbf{X}) = \frac{P(\mathbf{W})P(\mathbf{X}|\mathbf{W})}{P(\mathbf{X})} \tag{1.2}$$

Acoustic models estimate the probability distribution $P(\mathbf{X}|\mathbf{W})$. These models use feature representations of the signal $x[n]$ which enhance modeling performance. Let f represent a mapping which transforms the speech signal to a desirable feature representation.

$$f : \mathbb{R}^N \rightarrow \mathbb{C}^M \tag{1.3}$$

$$\mathbf{x}[t] \mapsto f(\mathbf{x}[t]) \tag{1.4}$$

where $\mathbf{x}[t]$ represents a vector of length N around time t in the original scalar sequence $\mathbf{X}$. The short notation $\mathbf{f}_t$ will be used to represent $f(\mathbf{x}[t])$, and $\mathbf{F} = \mathbf{f}_1..\mathbf{f}_n$. Thus the acoustic modeling problem is modified as $P(\mathbf{F}|\mathbf{W})$.

Hidden Markov Models (HMMs) are popularly used to estimate this probability. As it is difficult to estimate the parameters of the word sequence HMM models due to data sparsity in any practical scenario, the word sequences are decomposed into context-dependent phone sequences. Each of these context-dependent phones is represented using a HMM, with typically three states. As the parameters of these context-dependent phone HMMs can be estimated using data pooled across all the words, the sparsity problem is significantly alleviated.

The parameters of a HMM, composed of states $\{s_i\}_{i=1}^N \in \mathcal{S}$, are its transition probabilities $P(s_i|s_j)$ and emission probabilities $P(\mathbf{f}|s_i)$. We address the problem of estimating $P(\mathbf{f}|s_i)$ for all the states of the context-dependent phone HMMs henceforth referred to as the context-dependent (CD) states.

Gaussian mixture models (GMMs) can be used to estimate the likelihoods $P(\mathbf{f}|s_i)$.

However the GMMs make the strong assumption of statistical independence among the vectors $\mathbf{f}_t$ and in some cases even an assumption of uncorrelatedness among the elements of the vector $\mathbf{f}_t$. These are violated by the real data and this model error has been shown to be detrimental to the performance [1].

1.1 Neural network acoustic models

To overcome the limitations of HMM architecture stated above a number of variants which allow temporally correlated observations have been proposed, as summarized in Chapter $3.6.2$ of [2]. Of these the neural network based variant called the HMM-DNN (Hidden Markov Model Deep Neural Network) hybrid recognizer is widely adopted in the current state-of-the-art systems.

Neural networks when trained with a classification objective have been shown to estimate Bayesian *a posteriori* probabilities [3]. They do not make any assumptions of statistical independence or uncorrelatedness among the input feature vectors in the sequence or even among the individual features in the feature vector. Neural network based posterior estimators could thus be used to estimate the posterior probability distribution $P(s_i|\mathbf{f}_t)$ by training with the objective of predicting state labels s_i given the feature vector $\mathbf{f}_t$. The emission probabilities $P(\mathbf{f}_t|s_i)$ can be estimated from the $aposteriori$ probabilities using the Bayes' rule *i.e.,*

$$P(\mathbf{f}_t|s_i) = \frac{P(s_i|\mathbf{f}_t) * P(\mathbf{f}_t)}{P(s_i)} \tag{1.5}$$

However the marginal probability $P(\mathbf{f}_t)$ is same for all states s_i and does not change the classification performance. Thus practical systems use the estimate

$$P(\mathbf{f}_t|s_i) \approx \frac{P(s_i|\mathbf{f}_t)}{P(s_i)} \tag{1.6}$$

Neural networks have been shown to exploit the dependencies of vectors in the sequence $\mathbf{F}$ to generate more reliable estimates [4]. Further the correlations among the features in the input vectors $\mathbf{f}_t$ have been shown to further improve the performance when used with networks with multiple layers [5]. Thus the state probability estimators, used in practice, are actually of the form $P(s_i^t|\{\mathbf{f}_n\}_{n=t-l}^{t+r})$ where the l and r denote the left and right context of the input vector sequence used to estimate the state occupation probabilities at time t [4].

1.2 Problem

The problem of interest in this thesis can be stated as learning an estimate of the function $P(s_i|\mathbf{f}_t)$ using neural networks *i.e.,*

$$\hat{P}(s_i|\mathbf{f}_t) = \Phi(\mathbf{F}, t; \Theta) \tag{1.7}$$

where Θ are the parameters of the neural network. Note that we have used the entire feature vector sequence ($\mathbf{F}$) as input to the neural network to accommodate networks which have varied input contexts (e.g. bidirectional recurrent neural networks utilize the entire input sequence [6]).

Two major challenges in sequence modeling are **long-span sequential dependencies** and **sequential distortions**.

1.2.1 Long-span sequential dependencies

Due to these dependencies the probability estimate at a given instant t is dependent on the input vector $\mathbf{f}_{t\pm q}$ where q can be quite large. Training neural networks to model these long span temporal dependencies is non-trivial. Acoustic models which effectively tackle these dependencies typically have considerable latency as they require significant amount of future context.

1.2.2 Sequential distortions

Changes in speaker, channel and environment can lead to changes in the feature vector sequence $\mathbf{F}$. These changes are modeled as distortions of the input $\mathbf{X}$. In this thesis we are interested in the sequential distortions which create longer span temporal dependencies in the $\mathbf{X}$, which translate to dependencies in $\mathbf{F}$.

Let $d_x(.)$ represent a distortion which operates on the original speech signal $\mathbf{X}$ and $d_f(.)$ represent a distortion which operates on the feature vector sequence $\mathbf{F}$.

- In the far-field recognition scenario $d_x(.)$ is usually modeled as convolution with the room impulse response ($\mathbf{R} = \{r_i\}_{i=0}^{M}$) where M can be quite large (few seconds). Thus the distorted sequence $d_x(\mathbf{X}) = \mathbf{R} * \mathbf{X}$ has longer range dependencies than the original sequence $\mathbf{X}$.

- In scenarios such as radio transmission the distortions are non-linear [7] and also sequential.
- Speaking rate changes could be modeled by distortions which introduce non-linear temporal warp of the input sequence X [8].

All these distortions when composed with the feature mapping function f, which is usually non-linear, lead to more complicated interactions in the feature vector sequence $\mathbf{F}$.

In more adversarial scenarios there exist distortions $d_f(.)$ on the feature vector sequence $\mathbf{F}$ which permute the scalars in the vector $\mathbf{f}_t$. Examples of such scenarios are the spectral inversion problem seen in radio transmission or frequency scrambling done for encryption.

Additional distortions are usually introduced by the presence of other audio sources in the environment. Let $\mathbf{N}$ represent a sequence generated by another audio source. In this case the distorted sequence can have the form $d_1(\mathbf{X}) + d_2(\mathbf{N})$ where $d_1(.)$ & $d_2(.)$ are two different distortions e.g. in the case of far field recognition these would correspond to two different room impulse responses determined by the position of the speaker and noise source *w.r.t.* microphone.

A typical distorted feature vector sequence has the form

$$\psi(\mathbf{F}) = f(d_1(d_2(\mathbf{X}) + \sum_{i=1}^{M} d_{2+i}(\mathbf{N}_i))) \tag{1.8}$$

due to channel distortion ($d_1(.)$), reverberation ($d_2(.), ..d_M(.)$) and various noise sources in the environment ($\mathbf{N}_i$). The function class of distortions is rich and usually includes distortions which may not have been seen in the training data.

It is desirable to learn posterior estimators which are robust to these distortions $i.e.$,

$$\Phi(\mathbf{F}, t; \Theta) \approx \Phi(\psi(\mathbf{F}), t; \Theta) \tag{1.9}$$

1.3 Outline

Chapter 2 discusses existing approaches to tackle long span sequential dependencies. Based on their primary focus these approaches can be categorized as

- ***feature based approaches***, where feature representations are designed to capture long span information in the instantaneous representation provided to the model.

- ***model based approaches*** where neural network architectures are designed to learn long span dependencies from the short span representations.

Typical speech recognition systems use a combination of these two approaches. This thesis focuses on model based approaches.

In Chapter 2 we survey prior work, in Chapter 3 we describe the experimental setup, and in Chapters 4 and 5 we describe the proposed models. In Chapter 6 we demonstrate the applicability of the proposed models in the context of far-field speech recognition. Finally we present the conclusions and scope for future work in Chapter 7.

Chapter 2

Prior Work

Modeling of long-span temporal dependencies in speech signal has received focus in acoustic modeling research due to a variety of motivations. Information theoretic analyses ([9], [10]) measuring mutual information between spectro-temporal loci show discriminative dependence over a span of 100s of milliseconds. Studies on cortical processing of speech ([11]) provide evidence for integration of information at larger time scales (150-200 ms).

A trivial solution to model long-span temporal information in the acoustic model is to use a wider context of feature vectors (*i.e.*, $\{\mathbf{f}_i\}_{i=t-l}^{t+r}$) as an input to the neural network. In a standard feed-forward DNN an affine transform is learnt over the entire input context provided. However due to the temporal distortions in the speech input which lead to non-linear temporal warping, the space of possible input variations increases [12]. This in turn leads to a corresponding increase in the amount of data necessary to learn transforms invariant to these distortions.

A standard approach to reduce the possible variations in wider context inputs is to ensure stability of the feature representation to temporal warping. Stability is defined as *Lipschitz* continuity of the representation *i.e.*,

$$||\phi(x) - \phi(x_\tau)|| \leq C \underset{t}{sup} |\tau'(t)| \, ||x|| \tag{2.1}$$

where x represents the speech input of required context, x_τ represents the temporally warped input, $\phi(.)$ represents the feature function [1] and $\tau(t)$ represents the warping function [13]. Thus to ensure stability it is sufficient to make the feature function less sensitive to the absolute position of various acoustic events within the input context. This can be accomplished by using representations with low temporal resolution. However such coarse representations lose information necessary for classification. A variety of techniques have been proposed to preserve information while ensuring stability of the feature representation. These are discussed in Section 2.1.

In neural network based approaches the network is constrained to learn temporally local transforms, while ensuring that wider context information is integrated before predicting the output. Recurrent neural networks and convolutional neural networks are two different topologies which accomplish this. As the transforms are temporally local the impact of input distortions is reduced. These approaches are discussed in detail in Section 2.2.

[1] If the input was just stacked instantaneous features $\phi(x) = \{\mathbf{f}_i\}_{i=t-l}^{t+r}$

2.1 Feature based approaches

The spectro-temporal envelope of the speech signal $H(t, f)$ is the fundamental element of speech representations used in ASR systems. A variety of signal processing techniques are used to derive reliable estimates of the spectro-temporal envelope ([14], [15], [16], [17], [18], [13]). In speech recognition applications the spectral axis is warped ($\omega = w(f)$) to ensure stability to local temporal distortions ([13], [14]). This warped spectro-temporal envelope $H(t, \omega)$ is used to derive a variety of representations.

2.1.1 Multi-scale representations

The distortions in the speech signal due to changes in speaker, environment and speaking style translate to corresponding distortions in the spectro-temporal envelope $H(t, \omega)$. Local spectro-temporal distortions of the envelope, such as those induced by changes in vocal tract length, are tackled by using local smoothing operators (~ 25 ms windows). Mel filter-bank coefficients represent one such output.

To remain stable to distortions which produce longer span effects smoothing over a larger window is necessary. Typical examples of such distortions are speaking rate changes which can non-linearly warp the vowel durations, or reverberation. However such smoothing leads to loss of information which could be useful for classification. Hence these smoother long term representations are combined with short term representations [19]. Multi-scale features which represent the information in the spectro-temporal envelope at several different resolutions are adopted to tackle this issue. The coarser resolution features are stable to distortions while the finer resolution features preserve the detail necessary for better classification. Filter-bank design to generate these multi-scale features has been done using a variety of techniques. Data-driven filter design [20], gabor filter based design [21] and neuro-physiology inspired design [22] are some examples. The multi-scale features are frequently combined with multi-stream models [23] where features of each resolution are used in a separate stream of information, which are finally fused.

2.1.2 Long term temporal features

Long term temporal trajectories of feature vectors are frequently characterized by Fourier transform [24], Discrete Cosine Transform (DCT, [25]) or auto correlation coefficients. These representations are commonly termed modulation spectra. However these modulation spectra suffer from instability to time warping deformations [13]. Information from long term temporal trajectories can also be integrated using carefully designed modulation filters which enhance speech specific modulations (e.g. RASTA filter [26]). In some cases these modulation filters are derived form linear discriminant analysis of the data [27]. Other approaches such as Temporal Patterns (TRAPs, [28]) and Hidden Activation TRAPs (HATs, [29]) estimate the feature transforms using neural networks.

2.1.3 Permutation invariant representations

In the feature functions discussed in Sections 2.1.1 and 2.1.2 the representations derived preserve the sequence information i.e., they preserve information about the ordering of acoustic events in the feature vector sequence. In contrast to these approaches one could also estimate *permutation invariant* features to represent the long term information. A function is *permutation invariant* if the value of the function does not change irrespective of the ordering of the input sequence. Permutation invariant feature functions are used to summarize long term information typically over a span of several seconds. In neural network acoustic models they are used to perform *adaptive training* or *instantaneous adaptation*.

iVectors [30] which capture both speaker and environment specific information have been shown to be useful for instantaneous and discriminative adaptation of the neural networks ([31, 32, 33]). Noise mean estimates computed from all the noise frames in the utterance are also used as auxiliary information in neural network training [34].

2.2 Model based approaches

In model based approaches neural network architectures capable of modeling long term temporal dependencies from instantaneous feature representations are designed. Convolutional and recurrent architectures have been shown to effectively model long term dependencies in speech data.

In convolution architectures (*e.g.*, time-delay neural networks) the lower convolution layers learn temporally local transforms and the context of the network is gradually increased with depth of the network using hierarchical convolution. In the recurrent architectures output at a given instant is computed using the temporally local transform of the input and the state of the network, which summarizes the longer context information. The critical design element in both these architectures is the use of temporally local input transforms. Use of temporally local transforms reduces the variance in the input context of the transform. Further assuming the non-linear warping functions can be approximated with piece-wise linear warping functions the input in the context of the transform can be assumed to be linearly warped.

2.2.1 Recurrent neural networks

Recurrent neural networks (RNNs) which use a dynamically changing contextual window over all of the sequence history rather than a fixed context window have been shown to achieve state-of-art performance on LVCSR tasks [6, 35].

A simple recurrent neural network is represented by the equation 2.2. From the equation it is clear that the input transform $\mathbf{W_f}$ is temporally local. Standard representations (e.g. log mel filter-banks and MFCCs [36]) in speech tasks are stable to local ($\sim 25ms$) temporal distortions as they use a temporal smoothing filter. Thus $\mathbf{W_f}$ is stable to local temporal distortions. The longer context information necessary is captured in the state vector $\mathbf{y}(t-1)$. Thus the network is both stable to input distortions and also able to exploit long term temporal information in estimating the current output.

$$\mathbf{y}(t) = \sigma(\mathbf{W_f}\mathbf{f}(t) + \mathbf{W_y}\mathbf{y}(t-1)) \tag{2.2}$$

The temporally local transform $\mathbf{W_f}$ is shared across time steps of the input feature vector sequence. Further the transform $\mathbf{W_y}$ can be insensitive to the absolute position of the acoustic event within the input context of the network [2], as the entire past context is summarized as a fixed length state vector $\mathbf{y}(t-1)$. Hence both these transforms are learnt with shared statistical strength explicitly or implicitly across all time steps [37].

As the fixed length representation is learnt using a task dependent loss function, it can preserve the necessary detail from the context.

2.2.1.1 Variants of recurrent neural network architectures

RNNs are affected by a variety of learning issues of which vanishing and exploding gradients are prominent [38, 39, 40, 41]. Further they have also been shown to have limitations in the amount of memory [42]. To tackle these issues architectural variants have been proposed. Of these the Long Short Term Memory (LSTM) architecture has been shown to achieve state-of-art performance across a variety of sequence recognition tasks [43, 44, 6, 35]. In exhaustive and controlled empirical evaluations LSTM has been shown to be outperform or match the performance of other recurrent models [45]. LSTM relies on a fairly sophisticated gated structure with an ability to control flow of information within the

[2]the input context of a recurrent neural network is all the input until the current time step

network. This allows the network to potentially remember information for longer periods [46]. There has been a major focus on distilling the essence of the LSTMs.

The comparatively complex structure of LSTMs has motivated in proposal of several variants which tackle the longer term memory problem using different techniques. Gated recurrent units (GRU) [47] use a single gating unit to simultaneously control the forgetting factor and the decision to update the state unit. Clockwork RNNs (CW-RNN) [48] divide the hidden layer of the CW-RNN into separate modules, each running at a different clock speed. This allows CW-RNNs to learn information at different time scales. Even simple RNNs have been shown to learn long term dependencies when a part of the recurrent matrix is constrained to have a diagonal structure [46]. These diagonal units are shown to capture information at a longer range. Even when initialized with a diagonal structure [49] RNNs are shown to outperform LSTMs on some bench mark sequence learning problems. However these "simpler" RNNs have had limited or no success in acoustic modeling applications.

2.2.1.2 Bidirectional recurrent neural networks

Among the RNN acoustic models and their variants, e.g. LSTMs, the bidirectional versions have been shown to outperform the unidirectional versions by a large margin [50, 51]. However the latency of the bidirectional models is significantly larger, making them unsuitable for online speech recognition. To overcome this limitation chunk based training and decoding schemes [50, 52, 53, 54, 55] have been previously investigated to make them amenable for online decoding. A common characteristic of these methods is the use of frame chunks in place of the entire utterance, and they differ in the way the recurrent states are initialized when processing these chunks.

Chen *et al.*, [53] proposed the use of context-sensitive chunks (CSC), where a fixed context of frames to the left and right of the chunk is used to intialize the recurrent states of the network. Zhang *et al.*, [51] carried over the recurrent states for the forward LSTM from previous chunks reducing the computation on the left context. Xue *et al.*, [55] proposed the use of a feed-forward DNN to estimate the initial state of the backward LSTMs, for a given chunk. They also proposed the use of a simple RNN in place of an LSTM for the backward direction. Zeyer *et al.*, [50] proposed the use of overlapping chunks, without additional chunk context, and combining the posterior estimates from overlapping chunks. In all these *online* variants inference is restricted to chunk-level increments to amortize the computation cost of backward LSTMs, which significantly increases the model latency.

2.2.2 Convolution Neural Networks

As discussed in Section 2.2.1.1 learning in RNN architectures is non-trivial. Hence feed-forward alternatives to RNNs are of interest. In the previous section it was argued that a critical feature of the RNN is that it has a temporally local input transform which is shared across time steps. This same feature is also shared by *time-delay neural networks* (TDNN)[56, 57, 58]. A TDNN integrates longer term information by hierarchical filtering. Each layer of convolution learns a temporally local transform. The deeper convolution layers process inputs from wider contexts. However they operate on lower resolution output of the preceding layer, which ensures stability of the representation being processed by the convolution layer. Hence the deeper layers have the ability to learn wider temporal relationships, while remaining stable to distortions at the corresponding temporal width. This architecture has striking similarities to the multi-scale representations discussed in Section 2.1.1. The multi-scale representations are generated by the network at each layer by gradually processing inputs of wider context as we go deeper in the network. Further due to the sharing of temporal transforms across time steps there is once again sharing of statistical strength in learning the transforms i.e., during back-propagation, due to tying, the lower layers of the network are updated by a gradient accumulated over all the time steps of the input temporal context. Another advantage of this tying is that the convolution layers of the network are forced to learn translation invariant feature transforms, of the corresponding scale.

Though the TDNNs are claimed to operate on multi-scale representations [59], these representations are derived from transforms learnt from data. The filter responses of these transforms are not guaranteed to be low-pass. Thus the claim of multi-scale representations is not guaranteed. The representations at deeper layers can be explicitly made coarser through the use of pooling [12] like *max* or *average* pooling. The output of a layer is thus invariant (max pooling) or stable (average pooling) to the shifts of the input events within the pooling window.

2.2.2.1 Spectro-temporal convolution

In addition to temporal convolution described above, spectro-temporal convolution has also been widely explored in acoustic modeling. Spectral convolution followed by pooling is used to learn invariants to vocal tract length differences which manifest as warping along the spectral axis.

A variety of efforts have shown convolutional neural networks to be effective for both close-talk and far-field speech recognition, and robust speech recognition in general [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77].

Further recent efforts focusing on very deep convolutional architectures have been shown to perform competitively or better than LSTM acoustic models.

2.2.3 Convolutional and recurrent architectures

Combining convolution with recurrent layers has been previously shown to be helpful for acoustic modeling in [78, 79]. These hybrid architectures have been shown to perform better than either of these individual architectures. Specifically Sainath *et al.,* [78] had proposed prepending spatio-temporal convolutional layers and recurrent layers. Temporal convolution was also used by Amodei *et al.,* [79] above or below the recurrent layer stack, for better look-ahead. In these networks convolutional operations are typically used to model dependencies along spectral axis or for modeling temporal dependencies in the future context.

There have also been efforts to replace convolution operation with a recurrence relation for modeling spectral dependencies. *e.g.* recF-LSTMs [80], TF-LSTMs [81], grid LSTMs [82] or ReNet LSTMs [83] have been shown to effectively model both temporal and frequency dependencies and provide state-of-the-art performance [84].

In addition to this the convolution operation has also been used to replace the matrix multiplications in LSTM cells [85].

Chapter 3

Experimental Setup

In this chapter we detail the experimental setup. The goal of *online* speech recognition motivates the choices made in the design of this experimental setup. These decisions will be highlighted when we describe the individual modules.

As the results presented in this thesis are from a wide variety of large vocabulary continuous speech recognition (LVCSR) tasks with varying amounts of training data (80-1800 hours) and correspond to different conditions *e.g.* telephone speech, close-talk microphone speech and far-field speech, the experimental setup will describe the common aspects of all these systems. The critical difference across these setups is the data augmentation strategy, where robust speech recognition systems use multi-style training (MTR) [86]. However Ko *et al.,* [87] have recently shown that MTR is helpful even for telephone speech recognition systems. Hence the critical difference in acoustic models across these systems is in the amount of training data; which influences the hyper-parameters of the neural network acoustic model (e.g. number of layers, number of parameters per layer, cost function) and the training recipe (e.g. number of workers used during parallel training, number of training epochs).

Section 3.1 details the features, Section 3.2 presents the data augmentation strategy, Section 3.3 details the cost functions, Section 3.4 discusses the baseline models which include the (B)LSTMs, Section 3.5 describes the training procedure and finally Section 3.6 describes the lexicon and language model.

All the experimentation is performed using the Kaldi speech recognition toolkit [88].

3.1 Features

This section briefly describes the feature extraction and normalization procedure. For greater detail readers are recommended to refer to Povey *et al.,* [89].

3.1.1 MFCCs

Mel frequency cepstral coefficients (MFCCs) [14] without cepstral liftering are used as input features. These are equivalent to the log mel filter-bank coefficients popularly used with neural networks [5] but for the reversible discrete cosine transform. MFCCs are chosen as they enable the use of better compression schemes for training data transfer during parallel training [89]. 40 Mel filters are used per frame.

3.1.2 iVectors

iVectors originally developed for speaker identification applications [30] have been shown to be helpful even for speech recognition applications for performing *instantaneous speaker adaptation* [32, 31, 90]. iVector based ASR systems have been shown to be competitive with LVCSR systems which use speaker adapted features (*e.g.* fMLLR [91]) which typically require 2-pass decoding. However iVectors are typically estimated in an offline fashion *i.e.,* the feature vectors from the entire utterance or the set of utterances from the speaker are used to estimate iVectors. We restrict our attention to *online* iVectors where only the frames preceding the current frame are used in the iVector estimation process. In preliminary experiments Dan Povey *et al.,* have shown that online iVector systems perform similar to offline iVector systems in telephone and close-talk microphone tasks. However as we will discuss in Chapter 6 offline iVector systems perform significantly better than online iVector systems in far-field speech recognition tasks. As recommended in other training recipes [31] we use a 100 dimensional iVector.

3.1.3 Data normalization

Mean and variance normalization is typically performed on the input data. However in the current setup the concatenated input vector containing the iVector and the spliced feature vector, over an input window typically $[t-2, t+2]$ *i.e.,* 2 frames context on left and right of the current frame, is processed through a multi-class linear discriminant analysis

(LDA) transform matrix in lieu of mean and variance normalization. The output of this LDA transform is not only invariant to arbitrary affine transforms of the input but also helps avoid issues which are typically observed when iVectors are used in DNN acoustic models e.g. [92].

3.2 Data augmentation

Augmentation of speech data has been shown to be helpful for both low-resource speech recognition and for speech recognition in mismatch scenarios. For the experiments reported in this thesis two different data augmentation schemes have been used to create synthetic speakers or to simulate reverberation. These are detailed below.

3.2.1 Speed perturbation

Typically data augmentation is used in low-resource ASR tasks by creating synthetic speakers, which is accomplished using vocal tract length perturbation (VTLP) [93]. In addition to this tempo perturbation [94] is used to vary the speaking rate [95]. However temporal warping of the speech signal has been recently found to be superior to both the above techniques [96]. This warping is accomplished by scaling the time axis using a constant α *i.e.,* signal $x(t)$ warped by a factor α generates the signal $x(\alpha t)$. This warping not only shifts the frequency coefficients but also modifies the speaking rate. This technique is shown to provide an average relative improvement of 4.3% across several tasks compared to not using an augmentation technique. In the current set of experiments α of $\{0.9, 1.0, 1.1\}$ were used. This technique is termed *speed perturbation* [96].

3.2.2 Reverberation

This section describes the multi-style training recipe for robust far field speech recognition. Readers are recommended to refer to [97, 98, 87] for greater detail.

A major challenge in acoustic modeling is the mismatch between the train and test audio. Hence multi-style training is a widely adopted strategy to train robust acoustic models [86]. However the acquisition of real multi-style training data is non-trivial due to the associated costs; and simulation of training data is seen as a viable alternative. Multi-style training has had significant impact in robust acoustic modeling. For example, in the recent IARPA-ASpIRE far-field recognition challenge [99] which deals with far-field ASR in mismatched environments the best performing systems used data augmentation [98, 100]. Compared to other techniques used in these systems, data augmentation was shown to provide the most significant relative improvement. Further even in the case of products like *Google Home* [101], which require a far-field recognizer, multi-style training with simulated data has been shown to be critical [102].

Far-field data typically has reverberated speech and point-source noises, in addition to the isotropic noise at the receiver position. Reverberation is typically represented by convolution of the audio signals with a room impulse response (RIR) [103]. Among other things, these RIRs are affected by the room, receiver position and type, speaker position and positions of different obstacles. Assuming availability of RIRs corresponding to different source positions, samples of anechoic, e.g. close-talking speech, and samples of isotropic and point-source noises, we can simulate far-field speech using the following equation

$$x_r[t] = x[t] * h_s[t] + \Sigma_i n_i[t] * h_i[t] + d[t] \tag{3.1}$$

where $x_r[t]$ represents simulated far-field speech, $x[t]$ represents the speech signal, $h_s[t]$ represents the RIR corresponding to the speaker position, $n_i[t]$ represents a point-source noise and $h_i[t]$ represents the corresponding RIR, and $d[t]$ represents other additive noise sources like isotropic noise.

3.2.2.1 RIR databases

A variety of real room impulse response databases are available. In [97], Peddinti *et al.,* used three different RIR databases. These are

- the RWCP sound scene database[1] [104]
- the REVERB challenge database [105]
- the Aachen impulse response database [106]

325 muti-channel recordings of RIRs were selected from the three databases. Isotropic noise recordings were available for only 51 RIRs. The first channel from the multi-channel

[1]We would like to thank Mitsubishi Electric Research Laboratories (MERL), for providing the RWCP database and letting us host it on `openslr.org`

recordings was used for corruption.

3.2.2.2 Simulation of RIRs

From Equation 3.1, it can be seen that even the simulation of far-field speech requires several RIRs, corresponding to each source. Recording real RIRs in a wide variety of environments and at several points in the room is non-trivial; hence simulation of RIRs is a problem of interest. In [87], Ko *et al.,* studied the impact of using simulated RIRs in place of real RIRs. They identified that even with the use of several thousand simulated RIRs, the ASR system trained using the resultant MTR data could not match the performance of the ASR system trained with the real RIRs described in Section 3.2.2.1.

However as simulation allows for estimation of RIRs from several points in the room; inclusion of point source noises is trivial. Ko *et al.,* [87] identified that after the inclusion of point source noises even the system with simulated RIRs outperformed or matched the performance of the system corresponding to real RIRs and isotropic noises.

3.2.2.3 Noise database

The MUSAN corpus of music, speech and noise [107] was used for point source noises when using simulated RIRs. The dataset consists of music from several genres, speech from twelve languages, and a wide assortment of technical and non-technical noises. This database has an additional classification of background and foreground noise sources. It is released under a flexible Creative Commons license.

3.3 Cost functions

3.3.1 Sequence training with cross-entropy pretraining

As discussed in Section 1.1 this thesis focuses on HMM-DNN acoustic models. Acoustic models are typically trained using sequence discriminative cost functions like maximum mutual information (MMI) [108, 109], boosted MMI (bMMI) [110], minimum phone error (MPE) [111] or minimum Bayes risk (MBR) [112, 113, 114]. These cost functions have also been shown to be effective in the context of HMM-DNN models [115, 116, 117, 118, 119, 120]. Vesely *et al.,* [120] compared the cost functions listed above and reported no significant difference in performance. However state-level MBR (sMBR) has been widely adopted in Kaldi due to slight performance improvements.

The MBR cost function is given in Equation 3.2; where $A(\mathbf{W}, \mathbf{W}_u)$ is the raw accuracy between the two word sequences, κ is the acoustic scaling factor and the cost is being computed over a minibatch of utterances u.

$$\begin{aligned}\mathcal{L}_{MBR} &= \Sigma_u P(\mathbf{W}|\mathbf{F}_u) A(\mathbf{W}, \mathbf{W}_u) \\ &= \Sigma_u \frac{\Sigma_W p(\mathbf{F}_u|\mathcal{S})^\kappa P(\mathbf{W}) A(\mathbf{W}, \mathbf{W}_u)}{\Sigma_{W'} p(\mathbf{F}_u|\mathcal{S})^\kappa P(\mathbf{W}')}\end{aligned} \tag{3.2}$$

It is clear that the denominator can be significantly very expensive to compute as we have to sum over all possible word sequences *w.r.t* a given language model. To efficiently compute the denominator probabilities lattices which represent the most probable sequences are used [121, 122, 116].

Empirical studies have shown that the quality of the denominator lattice can significantly affect the performance [122]; which is in turn affected by the quality of the acoustic model used to generate these lattices. Further initialization of the neural networks is also shown to significantly impact the performance. Hence neural networks are typically pre-trained with a frame-level cost function like cross-entropy [116, 119]. The resultant neural networks are then used to both generate the lattices and initialize the model. The standard training recipe for HMM-DNN acoustic models is shown in Figure 3.1.

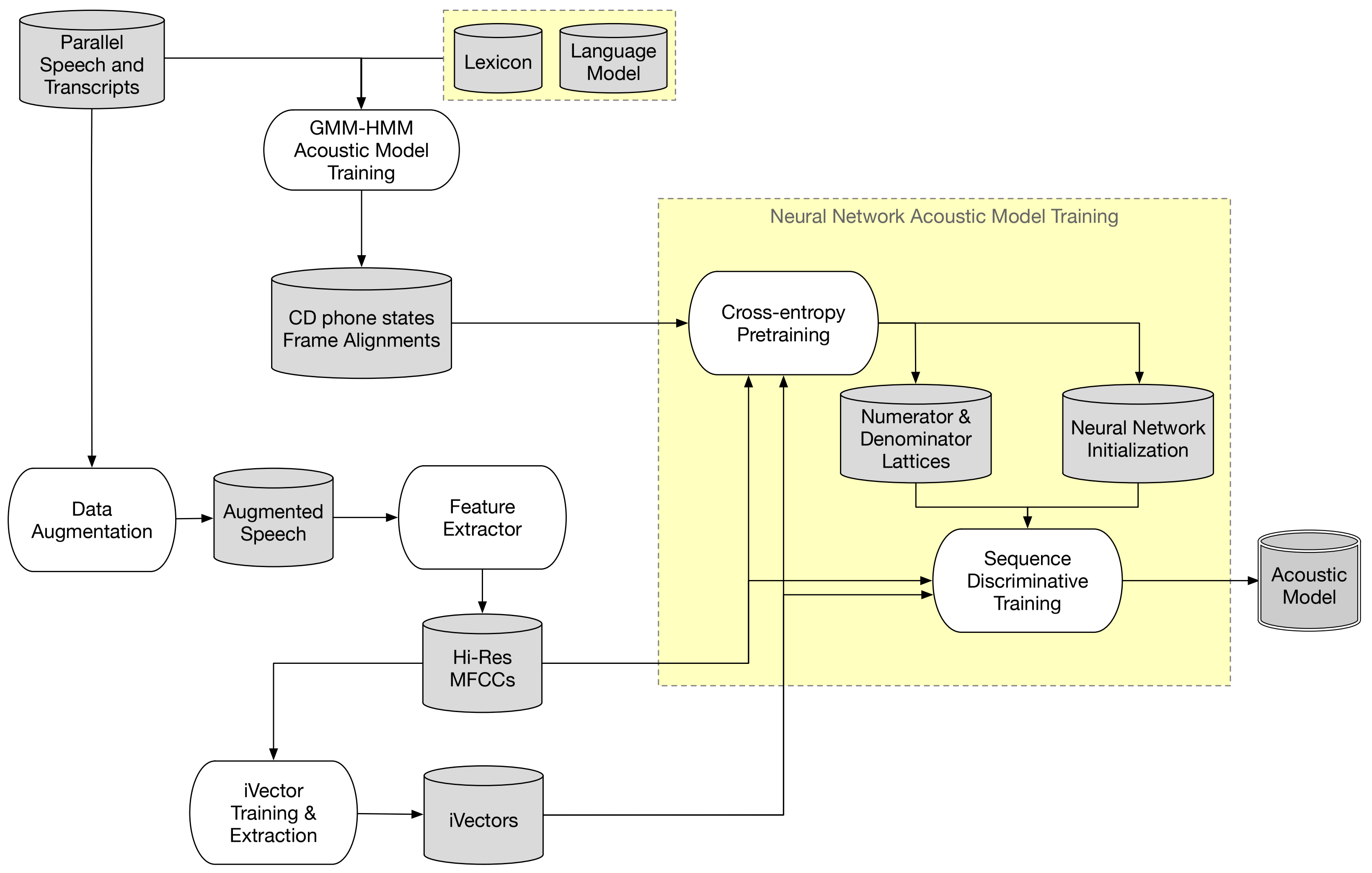

Parallel Speech and Transcripts
Lexicon
Language Model
GMM-HMM Acoustic Model Training
CD phone states Frame Alignments
Neural Network Acoustic Model Training
Cross-entropy Pretraining
Numerator & Denominator Lattices
Neural Network Initialization
Sequence Discriminative Training
Acoustic Model
Data Augmentation
Augmented Speech
Feature Extractor
Hi-Res MFCCs
iVector Training & Extraction
iVectors

Figure 3.1: Training procedure for HMM-DNN acoustic models using the conventional method of cross-entropy pre-training followed by sequence discriminative training. (The lexicon and language model training procedure is not detailed.)

3.3.1.1 Cross-entropy pre-training

Frame-level pre-training is typically done with minibatches composed of shuffled frames as it reduces the bias in the gradient estimated from a minibatch. This is standard procedure for feed-forward DNN training. Recently frame-shuffling has been found to be beneficial even for RNN acoustic models [54]. However as this can be computationally expensive chunk-based training is preferred for RNNs.

Fixed length sequences (chunks) were used for cross-entropy pre-training of the RNNs. This is similar to chunk back-propagation through time (BPTT) approach described by Chen *et al.,* in [123]. For evaluating frame level objective function a minibatch of 100 chunks with 200 ms in each chunk was used. The chunk width was limited to 200 ms, also used in [35], were shown to avoid gradient explosion issues. However to ensure that the state of the RNN is similar to its state in the actual sequence before estimating the first posterior vector, each chunk in the network is provided with a left context (40 frames). Similarly in the case of BLSTMs a right context (40 frames) was also provided to initialize the backward LSTMs. To match the train and test conditions, (B)LSTM decoding was also performed using chunks of length 20 frames. However an additional left/right chunk context of 50 frames was used as it was found to be beneficial.

3.3.1.2 sMBR training

Sequence training was done on the DNN, based on a state-level variant of the Minimum Phone Error (MPE) criterion, called sMBR. The training recipe mostly follows Vesely *et* *al.,* [120], although it has been modified for the parallel-training method. In addition to this both the sMBR criterion and the prior computation have also been modified.

Modified sMBR objective : In the sMBR objective function insertion errors are not penalized, which could lead to larger number of insertions when decoding with sMBR trained acoustic models. Correcting this asymmetry in the sMBR objective function, by penalizing insertions, was shown to improve performance of models trained with sMBR objective function when using reverberant training data by 10% WER relative [97]. Further this modified objective function was also found to be beneficial for sMBR training with telephone speech data. Hence this modified objective function was used.

More specifically, in standard sMBR training [113, 114], the frame error is always set to zero if the reference is silence, which means that insertions into silence regions are not penalized. In other words, frames where the reference alignment is silence are treated specially. (Note that in our implementation several phones, including silence, vocalized noise and non-spoken noise, are treated as silence for these purposes.) In our modified sMBR training method, we treat silence as any other phone, except that all pdfs of silence phones are collapsed into a single class for the frame-error computation. This means that replacing one silence phone with another silence phone is not penalized (e.g. replacing silence with vocalized-noise is not penalized), but insertion of a non-silence phone into a silence region is penalized. This is closer to the WER metric that we actually care about, since WER is generally computed after filtering out noises, but does penalize insertions. We call our modified criterion the *"one-silence-class"* modification of sMBR.

Modified prior computation : To compute the context-dependent state pseudo-likelihoods from the posteriors estimated by the neural network, the posteriors are divided by a prior. Manohar *et al.,* [124] found that the method of using the mean posterior (computed over a subset of the training data) as the prior, in place of the prior estimated from the alignments, gave an improved performance when decoding with sMBR trained models. Hence this prior estimation method was adopted in our system.

The lattice forward-backward computation for estimating the sequence objective function was done using chunks of width 1.5 seconds as opposed to the 200 ms chunks used during cross-entropy pre-training. Any degradation in posterior estimates due to the change in the chunk-widths during cross-entropy pre-training and sMBR training is expected to be corrected during sMBR training.

3.3.2 Purely sequence discriminative training

Recently connectionist temporal classification (CTC) objective function [125] has provided significant improvements in speech recognition applications [126, 127] when training with large amounts of training data. A critical change in this training recipe is the use of a sequence level cost function directly without the need for any frame-level pre-training. This modification to the training recipe impacts the choice of neural network architectures, as will be detailed in later chapters.

Some of the ideas adopted in CTC based training recipes have also been found useful in the context of MMI-based sequence training [128]. These are listed below :

- Training from scratch without initialization from a cross-entropy system
- The use of 3-fold reduced frame rate [129]
- Limiting the range of time frames where supervised labels can appear by using Finite State Acceptors [130]

Compared to conventional sequence discriminative training this new training procedure has some critical changes. These are

- ***Phone level denominator language model*** : As using a word level language model can be too slow for direct sequence training we use a 4-gram phone level language model.
- ***Topology and decision trees*** : In place of the conventional 3-state HMM which require minimum of 3 frames for traversal we use a HMM topology which can be traversed in 1 frame. More specifically we use a 2 state topology where the second state can be skipped. Further the phonetic context decision tree is re-built for this new topology.
- ***Regularization***: Three different regularizers are used to ensure better generalization. These are cross-entropy regularization, output l_2 regularization and *leaky HMM* regularization.
- ***Time constrained numerator graph***: To allow for easy splitting of numerator graph for training on GPUs we add time constraints on the numerator phone graphs. This is

accomplished by composing the phone level utterance specific finite state acceptors (FSAs) with a frame level mask defining overlapping phone boundaries. A tolerance of ~ 50 ms is allowed for the phone boundaries derived from frame alignments.

This purely sequence discriminative training recipe is summarized in Figure 3.2, for greater detail readers are recommended to read Povey *et al.,* [128].

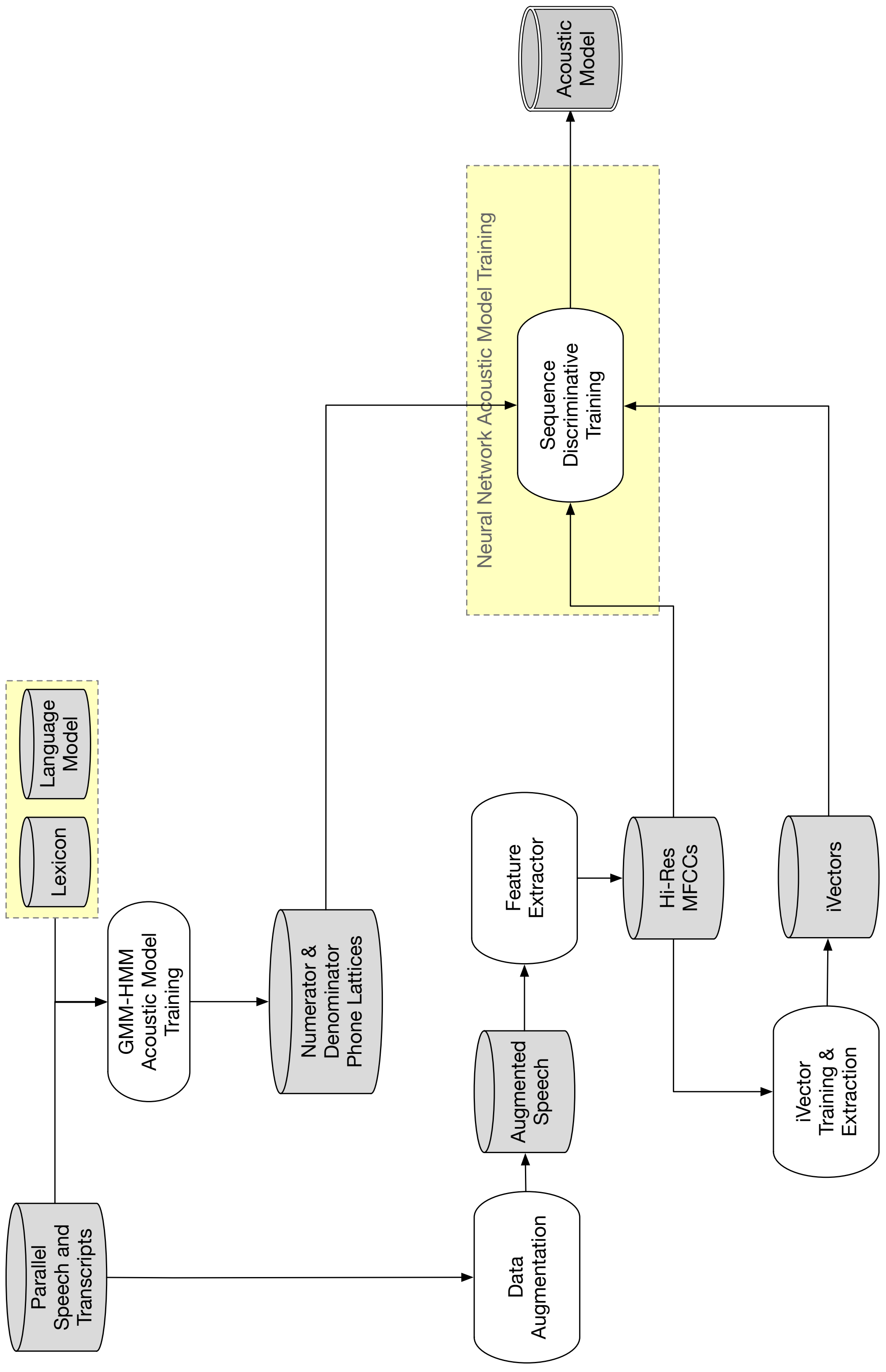

Parallel Speech and Transcripts
Lexicon
Language Model
GMM-HMM Acoustic Model Training
Numerator & Denominator Phone Lattices
Data Augmentation
Augmented Speech
Feature Extractor
Hi-Res MFCCs
iVector Training & Extraction
iVectors
Neural Network Acoustic Model Training
Sequence Discriminative Training
Acoustic Model

Figure 3.2: Purely sequence discriminative training procedure for HMM-DNN acoustic models

3.4 Model

The baseline acoustic models in this thesis are the simple feed-forward DNN and (B)LSTMs. In this section we will briefly describe the implementation of these models in the *nnet3* toolkit, which is used for the experiments in this thesis. The *nnet3* toolkit, by Povey *et al.,* [131] in Kaldi speech recognition toolkit [88] was used to perform neural network training and inference. This toolkit allows computation of neural networks whose computation can be written as an acyclic computation graph. In case of recurrent neural networks where the current state of the neural network depends on the state of the network at another instant of time the computation graph is unfolded in time to create an acyclic computation graph.

3.4.1 Projected LSTM (LSTMP) model

The LSTM implementation closely follows the equations described in Sak *et al.,* [35]. The experiments in this thesis the projected variant of LSTMs typically abbreviated as LSTMP. The equations representing this computation cell are reproduced here, from [35] for the reader's convenience.

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + w_{ic}c_{t-1} + b_i) \tag{3.3}$$

$$f_t = \sigma(W_{fx}x_t + W_{rf}r_{t-1} + w_{fc}c_{t-1} + b_f) \tag{3.4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cr}r_{t-1} + b_c) \tag{3.5}$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + w_{oc}c_t + b_o) \tag{3.6}$$

$$m_t = o_t \odot h(c_t) \tag{3.7}$$

$$r_t = W_{rm}m_t \tag{3.8}$$

$$p_t = W_{pm}m_t \tag{3.9}$$

r_t and p_t which represent the recurrent and non-recurrent projections are concatenated to form the output of the LSTMP computation cell.

$W_{ix}, W_{ir}, W_{fx}, W_{rf}, W_{cx}, W_{ox}, W_{or}, W_{rm}, W_{pm}$ are rectangular matrices and w_{ic}, w_{fc}, w_{oc} are diagonal matrices. The dimensions of these matrices are determined by the hyper-parameters c_t, r_t, p_t. These hyper-parameters are chosen according to the training data available. For the widely used Switchboard 300 hour LVCSR task these are $c_t = 1024, r_t = 256, p_t = 256$.

Based on our preliminary experiments we chose a stack of 3 LSTMP cells as our baseline. Further increase in the size of the LSTMP stack or the hyper-parameters led to minor improvements while increasing the computational cost significantly.

3.4.2 Bidirectional projected LSTM (BLSTMP) model

The bidirectional LSTMP (BLSTMP) computation cell has an additional backward direction LSTMP computation cell. This backward LSTMP operates on the same inputs as the forward LSTMP however in the reverse direction. Further the outputs of both the forward and backward LSTMP cells are concatenated to form the BLSTMP cell output.

Let $\mathcal{F}(.;\theta)$ represent LSTMP transformation represented in Section 3.4.1, with parameters θ, which consumes the input sequence x_t and outputs r_t and p_t. As bidirectional models consume the input sequence in both the forward and backward directions we represent the direction of the transforms using a $\overrightarrow{\cdot}$ for forward transforms and outputs and $\overleftarrow{\cdot}$ for backward transforms and outputs. Finally $[.,.]$ represents the concatenation operation.

$$[\overrightarrow{r_t}, \overrightarrow{p_t}] = \overrightarrow{\mathcal{F}}(x_t; \theta_f) \tag{3.10}$$

$$[\overleftarrow{r_t}, \overleftarrow{p_t}] = \overleftarrow{\mathcal{F}}(x_t; \theta_b) \tag{3.11}$$

The concatenated output $[[\overrightarrow{r_t}, \overrightarrow{p_t}], [\overleftarrow{r_t}, \overleftarrow{p_t}]]$ represents the output of the BLSTMP.

The hyper-parameters *i.e.,* sizes of $\overrightarrow{c_t}, \overrightarrow{r_t}, \overrightarrow{p_t}, \overleftarrow{c_t}, \overleftarrow{r_t}, \overleftarrow{p_t}$ were chosen to be the same for both the LSTMP cells. For the Switchboard 300 hour LVCSR task these are $c_t = 1024, r_t = 256, p_t = 256$. Once again a stack of 3 BLSTMP cells was chosen for the experiments reported in this thesis. It can be seen that the number of parameters in the BLSTMP model are significantly larger than that of the LSTMP model. Experiments by

Yiming Wang, further increasing the parameters of the LSTMP model led to a degradation in the performance.

3.5 Training

This section describes briefly the parallel training procedure used in *nnet3*. For greater details the readers can refer to [89].

The training procedure in *nnet3* is geared towards using large amounts of training data on multiple GPU-equipped machines. Further in order to be as hardware-agnostic as possible, the procedure reduces the network traffic across these multiple machines by adopting a model averaging method for synchronization, in place of gradient averaging [132, 133] or asynchronous stochastic gradient descent [134].

In this method the neural network parameters from the parallel workers are periodically averaged (typically every minute or two) and the averaged parameters are redistributed to the workers. The method relies on data parallel training where each worker sees different data.

To ensure that the model synchronization after large intervals leads to convergence the parameter updates are computed using an approximate and efficient implementation of Natural Gradient for Stochastic Gradient Descent (NG-SGD) [89].

3.5.1 Recurrent neural network training

Some salient aspects in the recurrent neural network training are the use of *shrinkage* and clipping of gradients *w.r.t.* activations. In *shrinkage* the parameters of the network are scaled by constant (0.99) when the mean derivate at the sigmoid non-linearities in the network falls below a threshold (0.15). This scaling increases the probability of affine matrix multiplication outputs remaining in the linear range of the sigmoid non-linear units in the network.

3.6 Language model and lexicon

We typically use n-gram language models (LMs) in our experiments. Further we restrict our attention to three and four grams. However recurrent neural network (RNN) LMs have been used in the ASpIRE far-field LVCSR task. Hence we describe this LM briefly. For greater detail readers are recommended to refer to Peddinti *et al.,* [98].

3.6.1 N-gram LM

N-gram language models were used in the decoding to generate word lattices. A trigram language model (LM) was first trained on the 3 million words of the training transcripts, which was later interpolated with another trigram LM trained on 22 million words of the Fisher English transcripts (LDC2004T19 and LDC2005T19). The same process is repeated for building a 4gram LM. We used SRIs language modeling toolkit SRILM [135] for building our LMs, with Kneser-Ney smoothing. The final trigram LM has 1.6 million trigrams and the 4gram LM has 1.7 million 4grams. The trigram LM is used for decoding; and the 4gram LM, is only used for rescoring the lattices generated by the trigram LM.

3.6.2 RNN-LM rescoring

3.6.2.1 Training

The RNNLM toolkit (0.3e) [136] is used to train the recurrent neural network language models (RNN-LMs). A 40,000 subset of the most frequent words from training transcripts are chosen as the language model vocabulary. A 10,000 utterance subset of the training transcripts are chosen as a heldout set, while the remaining transcripts are used for training the RNN-LMs. The RNN-LM model has 200 hidden units in the neural network. The number of word classes was chosen to be 350. The words in the 40,000 vocabulary were assigned to the 350 word classes according to their unigram frequency in the training corpus. The maximum order of the n-gram features [137] was set to 4, and truncated back-propagation through time (BPTT) [138] was performed during training with a step size of 2. Words from the training transcripts that were not selected in the RNN-LM vocabulary were mapped to a special word "$\langle RNN\quad UNK\rangle$" before training. The unigram probabilities of those words were also collected from training transcripts which later served as penalties when we compute the likelihood of sentences containing those words. For words that were not in the training transcripts nor in the RNN-LM vocabulary, a fixed unigram probability of $1e7$ was used instead.

3.6.2.2 N-best rescoring

The conventional N-best RNN-LM rescoring procedure was used. In this procedure, the N-best hypotheses are first extracted from lattices, with their associated acoustic score, original language model score, graph cost as well as frame level alignments. The RNN-LM likelihood is then computed for each hypothesis, with the out-of-vocabulary words properly penalized as described in the previous section.

In the experiments in [98], it was found that interpolating the RNN-LM likelihood with the original language model did not help much, so the original language model score was simply replaced with the RNN-LM score. The acoustic score, RNN-LM score, graph cost and the frame level alignment of all the hypotheses were then packed back to create a new lattice, with which we ran the decoding. It worth mentioning that it is important to generate the N-best hypotheses with the optimal acoustic scale.

3.6.2.3 Lattice rescoring

One drawback of applying RNN-LMs on N-best list is the N-best list only covers a subset of the hypotheses from the original lattice. Therefore if the original language model is not powerful enough, and the correct word falls out of the N-best list, there is no way for RNN-LM to recover it. A simple solution is to generate as many hypotheses as possible, which of course will increase the decoding cost. Applying RNN-LM rescoring directly on the lattice however can increase the lattice size exponentially since the number of distinct RNNLM context states will grow exponentially. A general solution is to derive appropriate equivalence classes for context states. In [139], two different methods to cluster the context states were evaluated:

- clustering context states using n-gram history, and
- clustering context states based on the context vector distance.

The authors were able to get the same performance from these two methods.

In [98], we cluster the context states using n-gram history since this is computationally cheap. We implemented our lattice based RNN-LM rescoring within the weighted finite state transducer (WFST) framework. During the rescoring, a grammar WFST is generated on-the-fly, whose states each correspond to a unique n-gram sequence. The weight of the arc given the states is given by $P(w_n|w_1, ..., w_{n1})$, where w_n is the word on the arc, and $w_1, ..., w_{n1}$ is the n-gram history that the state corresponds to, which can be computed from RNN-LM. In the actual implementation we store the RNN-LM context vector instead the word sequence, so that the RNN-LM can compute $P(w_n|h)$ directly, where h is the context vector of the word sequence whose latest $n1$ words are $w_1, ..., w_{n1}$. Ideally, we would like h to correspond to the best possible sequence in the lattice entering the state. Our preliminary results however shows that we may not benefit a lot from using the context vector from the best possible word sequence entering the state. Therefore in our current implementation we simply use the context vector from the first word sequence we see whose latest $n1$ words correspond to the state.

3.6.3 Enhanced lexicon

This section briefly describes the lexicon generation procedure for greater detail readers can refer [140].

CMUdict (0.7a) was used as training lexicon in our experiments, and the vocabulary was restricted to the words that appear in the training transcripts. CMUdict comes with multiple pronunciations for some words, therefore we estimate the pronunciation probabilities during the training. We also model inter-word silence probabilities as described in [140]. The statistics for modeling pronunciation and silence probabilities were estimated from training data alignment, and they were later encoded into the lexicon finite state transducer during the decoding. We estimate pronunciation probabilities for a word with multiple pronunciations via simple relative frequency, with proper smoothing techniques [141, 142, 143]. Directly using the simple relative frequency however has an undesirable consequence that a word with several equiprobable pronunciations is unfairly handicapped *w.r.t* words that have a single pronunciation: *e.g.* the past tense of "read" *w.r.t* the color read "red". Max-normalization, whereby the pronunciation probabilities are scaled so that the most likely pronunciation of each word has "probability" 1, has been found helpful in speech recognition [144]. We therefore applied max-normalization for pronunciation probabilities in our work. For a given sequence of words, we assume there is either a silence or non-silence event between two consecutive words. Since such an event usually depends on the neighbouring words, we further assume that it only depends on the two surrounding words, i.e., we model the event using $P(s|w.p_i, w'.p_j)$ and $P(n|w.p_i, w'.p_j)$, where $w.pi$ and $w'.p_j$ are the surrounding pronunciations, s and n represent silence and non-silence event. For computation simplicity, we decompose this into two parts:

- probability of inter-word silence (or non-silence) following the pronunciation, and
- probability of inter-word silence (or non-silence) preceding the pronunciation

Further details on the computation of probabilities can be found in [140].

3.7 Summary

In this chapter we described the training recipes of the acoustic and language models used in our LVCSR system. As these recipes are constantly updated and can change during the course of our experimentation spread over several years we highlight any changes from this description in the individual sections.

Chapter 4

Sub-sampled time-delay neural networks

In this chapter a sub-sampled variant of time delay neural networks (TDNN) is proposed. The sub-sampling process reduces the computational complexity compared to TDNN. It is shown to be a viable alternative to recurrent neural network architectures, for modeling long-span temporal contexts. We initially propose the use of sub-sampled time delay neural networks in the context of sequence training with cross-entropy pretraining, which involves frame-shuffling. We further explore variants to this architecture to exploit the lack of frame-shuffling in the context of purely sequence discriminative training.

4.1 Time delay neural networks

Modeling long term temporal dependencies is critical in acoustic modeling. As discussed in Chapter 2's Section 2.2.2 recurrent neural networks are a natural fit for modeling these sequential dependencies. However due to issues in optimization such as vanishing and exploding gradients alternative neural networks are of interest.

As described by Goodfellow *et al.,* [37] parameter sharing across time steps enables generalization to variable length sequences in recurrent neural networks; and this property is also shared by time delay neural networks which perform 1D convolution on temporal sequences. TDNNs [58] can be considered a precursor to the convolutional neural networks [145], without explicit pooling operations. They have been previously shown to be effective in modeling temporal contexts [56, 57, 58].

The salient feature of these networks is the use of temporally local transforms which are less sensitive to temporal distortions, as compared to affine transforms which span the entire input context. When processing a wider temporal context, in a standard feed-forward DNN, the initial layer learns an affine transform for the entire temporal context. In a TDNN architecture the initial transforms are learnt on narrow contexts and the deeper layers process the hidden activations from a wider temporal context. Hence the deeper layers have the ability to learn wider temporal relationships, on representations with lower resolutions which are comparatively more stable to distortions. The use of hierarchical temporal convolution enables learning of multi-scale representations [57].

During back-propagation, due to tying, the lower layers of the network are updated by a gradient accumulated over all the time steps of the input temporal context. Thus the lower layers of the network are forced to learn translation invariant feature transforms. The tying of transforms also increases the statistical strength of the update.

A major issue with TDNNs is the linear increase in parameters with increase in the input temporal context used to compute one output frame; as either the contexts of the existing filters have to be increased or the depth of the network has to be increased to span this additional context. Further there is also a linear increase in computation when TDNNs are used in frame shuffled cross-entropy pretraining as intermediate convolutions cannot be shared across neighboring outputs.

4.1.1 Subsampling

In a typical TDNN, hidden activations are computed at all time steps. However there are large overlaps between input contexts of activations computed at neighboring time steps. Under the assumption that neighboring activations are correlated, they can be sub-sampled. In this section sub-sampling is explored as a mechanism to not only reduce the computational cost but also to alleviate the issue of linear increase in parameters.

4.1.2 Proposed approach

In the proposed approach the sampling rate of the neural network is decreased exponentially with the depth of the network, which is similar to applying strided convolution at all the temporal convolutional layers. However we support non-uniform subsampling of temporal frames.

In our sub-sampling method at each hidden layer of the network, we generally splice no more than two frames *i.e.,* only the frames at the edges of the temporal convolution kernel are used as an input. To model wider temporal contexts we just increase context of the existing filters; and due to the sub-sampling this does not increase the number of parameters or the computational cost. However we always ensure that the information from all the frames in the input context is used in the computation of each output frame. Thus the proposed sub-sampling strategy alleviates the problem of linear increase both in parameters and computation with increase in input context, associated with TDNNs.

A typical sub-sampled TDNN employed in our experiments is used for the pictorial description of the proposed sub-sampling method. We specify the dependency graph for the computation of one output frame. For TDNN architectures this translates into specification of splicing indices which define the temporal convolution kernel input at each layer. These splicing indices are specified *w.r.t.* input's time indexing. *e.g.* `{-7,2}` means that the input to the temporal convolution at a given time step `t` is a spliced version of previous layer outputs at times `t-7, t+2`.

Figure 4.1 shows the computation graph for a TDNN with 4 temporal convolution layers with kernel contexts of `{-7,+2}, {-3,+3}, {-1,+2} & {-2,-1,0,1,2}`. The frames in red in Figure are the ones evaluated in order to compute one output frames, while the red and blue frames correspond to the computation in a normal TDNN. The lines

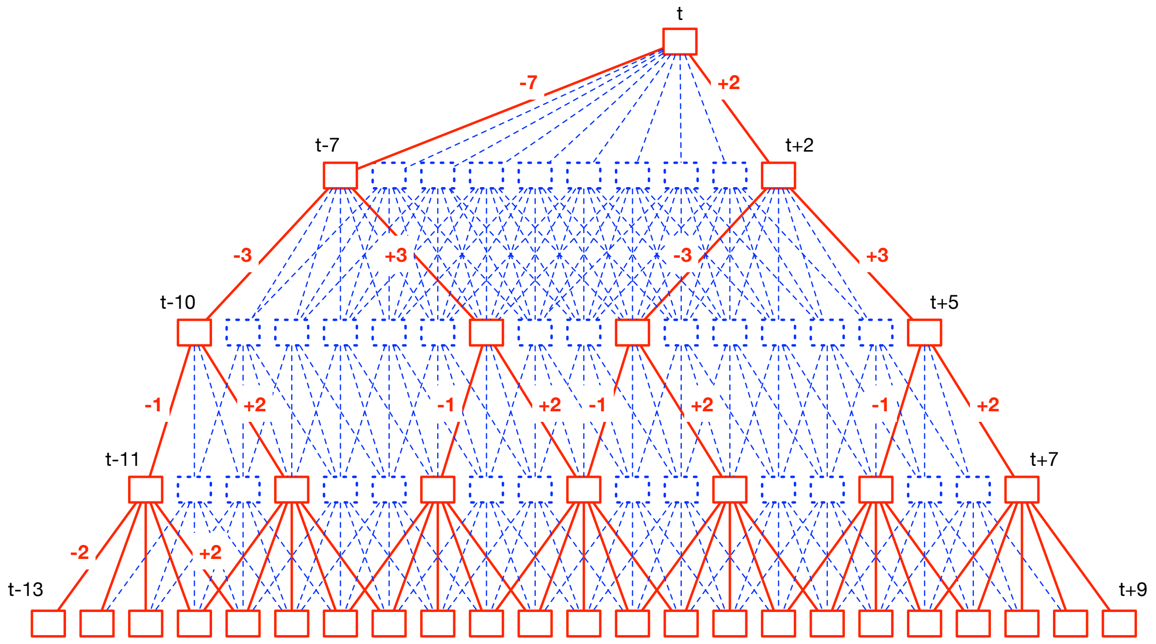


Figure 4.1: Computation in TDNN with sub-sampling (red) and without sub-sampling (blue+red). The numbers in red represent the kernel contexts.

between the frames correspond to the dependencies. As described above, in the subsampled TDNN the layer output at time t just depends on the previous layer outputs at the edges of the convolution kernel. To ensure that we span the entire input context of the network the lowest temporal convolution layer is not sub-sampled. It can be seen that the overlap in the inputs of the convolutional kernels is minimal even at the lowest convolutional layer.

Compared to the non-subsampled TDNN, represented by the frames in red and blue the computational savings are obvious. With the current sub-sampling scheme the overall necessary computation is reduced during the forward pass and backpropagation, due to selective computation of time steps. The training time of TDNN in Figure 4.1, without sub-sampling, is $\sim 10x$ compared to that of DNN with same number of layers, hidden units and input context. With proposed sub-sampling it is $\sim 2x$ the training time of DNN. Thus the sub-sampling process speeds up the TDNN training by $\sim 5x$.

We use asymmetric input contexts, with more context to the left, as this reduces the latency of the neural network in online decoding, and also because this seems to be more optimal from a WER perspective. Asymmetric context windows of up to 16 frames in past and 9 frames in the future were explored in this set of experiments. It was observed that further extension of context on either side was detrimental to word recognition accuracies, though the frame recognition accuracies improved (this phenomenon is widely known).

4.1.2.1 Prior Work

This sub-sampling strategy, if performed uniformly across layers, is similar to strided convolution which is popularly applied in convolutional neural networks. It has been proposed in the context of wavelet based signal processing as dilated convolution [146, 147]. Dilated convolution based neural networks have also been successfully applied in text-to-speech systems under the name *Wavenet* [148]. A salient difference of our approach compared to these works is the support for non-uniform sampling.

In the context of speech recognition convolutional networks with striding at the intermediate layers have been used in [149, 150]. Sub-sampling at the middle of the network was also used in stacked bottle-neck networks [151].

4.1.3 Results

4.1.3.1 Comparison with DNNs

The initial set of results presented in Table 4.1 compare the performance of simple feed-forward DNN and TDNN models on the Switchboard 300 Hour LVCSR task. We present results on Switchboard subset as well as the complete Hub5 ’00 evaluation set. Only the results in the SWB column should be compared with the Hub5 ’00 results presented in [152], [153] and [154].

Each neural network has 4 hidden layers with *p*-norm input dimension of 3000 and group size of 10. The experimental setup is similar to that described in Chapter 3 and the readers can refer to Peddinti *et al.,* [155] for the exact specification.

From Table 4.1, comparing DNN-A and TDNN-A, it can be seen that even with standard temporal contexts TDNNs perform better than DNNs. The number of parameters in the DNN-A system were increased to match the TDNN-A system, by increasing the number of hidden units. The results corresponding to this system are presented in the row titled DNN-A_2. It can be seen that despite matching the number of parameters the TDNN system performs better than the DNN system, for the same temporal context. A comparison of DNN-A, DNN-B and TDNN-D shows that DNNs are not as effective as TDNNs in processing wider temporal contexts. Comparing TDNNs A through E, $[-13, 9]$ was found to be the optimal temporal context.

Table 4.1: Performance comparison of DNN and TDNN with various temporal contexts

Model	Network Context	Layerwise Context					WER	
		1	2	3	4	5	Total	SWB
DNN-A	$[-7, 7]$	$[-7, 7]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.1	15.5
DNN-A_2	$[-7, 7]$	$[-7, 7]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	**21.6**	**15.1**
DNN-B	$[-13, 9]$	$[-13, 9]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.3	15.7
DNN-C	$[-16, 9]$	$[-16, 9]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.3	15.7
TDNN-A	$[-7, 7]$	$[-2, 2]$	$\{-2, 2\}$	$\{-3, 4\}$	$\{0\}$	$\{0\}$	21.2	14.6
TDNN-B	$[-9, 7]$	$[-2, 2]$	$\{-2, 2\}$	$\{-5, 3\}$	$\{0\}$	$\{0\}$	21.2	14.5
TDNN-C	$[-11, 7]$	$[-2, 2]$	$\{-1, 1\}$	$\{-2, 2\}$	$\{-6, 2\}$	$\{0\}$	20.9	14.2
TDNN-D	$[-13, 9]$	$[-2, 2]$	$\{-1, 2\}$	$\{-3, 4\}$	$\{-7, 2\}$	$\{0\}$	**20.8**	**14.0**
TDNN-E	$[-16, 9]$	$[-2, 2]$	$\{-2, 2\}$	$\{-5, 3\}$	$\{-7, 2\}$	$\{0\}$	20.9	14.2

4.1.3.2 Comparison across several LVCSR tasks

Table 4.2: Baseline vs TDNN on various LVCSR tasks with different amount of training data

Database	Size	WER		Rel. Change
		DNN	TDNN	
Res. Management	3 hrs	2.27	2.30	-1.3
Wall Street Journal	80 hrs	6.57	6.22	5.3
TedLIUM	118 hrs	19.3	17.9	7.2
Switchboard	300 hrs	15.5	14.0	9.6
Librispeech	960 hrs	5.19	4.83	6.9
Fisher English	1800 hrs	22.24	21.03	5.4

Experiments were done using Kaldi speech recognition toolkit [88] on Resource Management [156], Wall Street Journal [157], TedLIUM [158], Switchboard [159], Librispeech [160] and the english portion of Fisher corpora [161]. The amount of training data available for acoustic modeling varies from 3-1800 hours across the setups mentioned.

An average relative improvement 5.52% was observed over the baseline DNN architecture through the use of TDNN architecture to process wider contexts. It is to be noted that the number of parameters in the system are not matched between DNN and TDNN architectures. However the individual systems were tuned for best performance, given the architecture.

In the Resource Management medium-vocabulary task, we did not see gains from TDNNs. This could be due to the slight increase in parameters in the TDNN architecture when processing larger input contexts.

4.1.3.3 Comparison with unfolded RNNs

Table 4.3: Results on SWBD LVCSR task with data augmentation and enhanced lexicon

Acoustic Model + Language Model	WER	
	Total	SWB
TDNN - D + pp	21.9	14.8
TDNN - D + pp + fg	20.4	13.6
TDNN - D + pp + fg + sp + vp	19.2	12.9
TDNN - D + pp + fg + sp + vp + silp	19.0	12.7
TDNN - D + pp + fg + sp + vp + sequence training	17.6	11.4
TDNN - D + pp + fg + sp + vp + sequence training + pa	17.1	11
unfolded RNN + fMLLR features + iVectors [153]	-	12.7
unfolded RNN + fMLLR features + iVectors + sequence training [153]	-	11.3
CNN/DNN joint training + fMLLR features + iVectors [154]	-	12.1
CNN/DNN joint training + fMLLR features + iVectors + sequence training[154]	-	**10.4**

pp : pronunciation probabilities	sp : speed perturbation
fg : 4-gram LM rescoring	vp : volume perturbation
silp : word position dependent silence probabilities	pa : prior adjustment

A smaller TDNN with layer-wise contexts of TDNN-D was built, as it would be suitable for online speech recognizers. The size was reduced by decreasing *p*-norm input dimension from 3000 to 2750. This system has 7.7 million parameters. It was able to achieve a word error rate of 11.0%, which is better than the result reported for unfolded recurrent networks in [153]. Table 4.3 shows the contributions of each technique described in our experimental setup *e.g.,* volume perturbation, data augmentation, enhanced lexicon. We would like to remind the readers that iVectors are used in all the TDNN systems in Table 4.3.

4.1.4 Subsampled TDNNs and contiguous output computation

As described in Section 4.1.1 the subsampling strategy described above significantly reduces the computational cost of cross-entropy pre-training when frame shuffling is employed. However the computational gains during sequence training or inference are minimal as contiguous outputs are computed during these stages. Figure 4.2 represents the TDNN computation graph for a sequence of outputs. It can be clearly seen that the even after sub-sampling the intermediate convolutions required for a single output, the computation of neighboring outputs necessitates the computation of the sub-sampled indices. Thus exploiting this additional information, available during sequence training and inference, is of our interest. In the next section we describe the changes in the neural network training procedure which enables us to accomplish this.

4.2 Purely sequence training of neural networks

Recently training with just sequence level cost functions has been shown to be very effective for acoustic modeling [129, 162]. In this scenario frame-shuffling is no longer

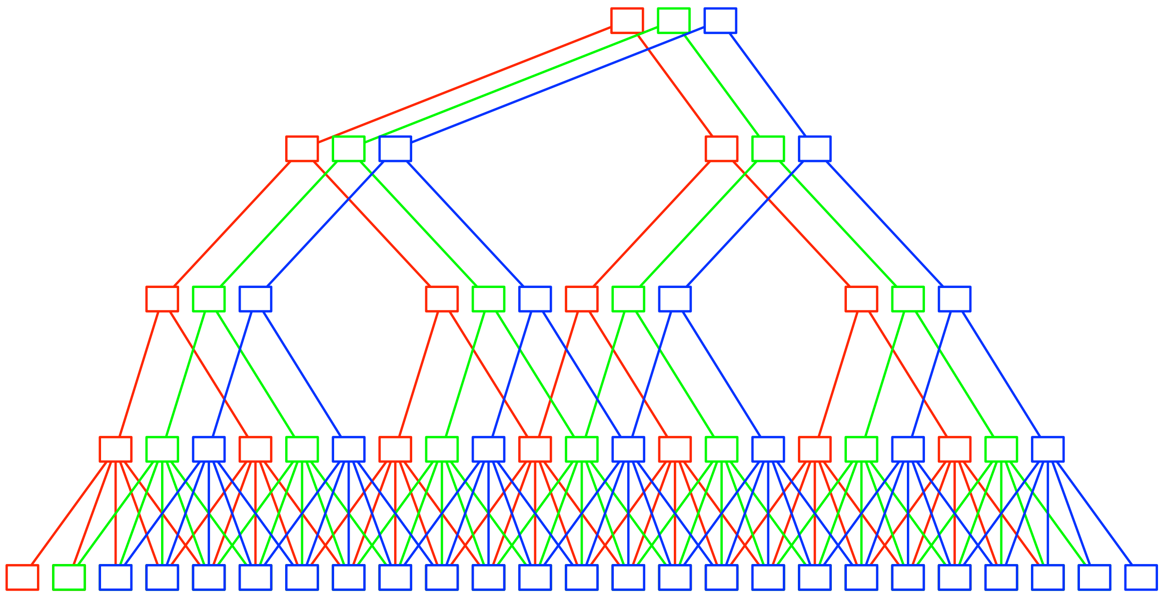


Figure 4.2: Computation in a subsampled TDNN when computing contiguous outputs. It can be seen that even the hidden layer outputs, sub-sampled for a given neural network output, have to be computed for neighboring outputs.

applicable. Thus the computation can be amortized over all the outputs in the sequence and sub-sampled TDNNs can match the output frame rate even at deeper layers.

Sak *et al.,* [129], Povey *et al.,* [162] and Pundak *et al.,* [163] have shown that lower output frame rate models outperform conventional frame rate models, while providing great savings in computation. They propose the use of reduced frame rates of $25 - 33$ Hz for the neural network outputs, in place of the 100 Hz output frame rates. Hence we change the frame rate at all the layers in the TDNN to match the output frame rate (33 Hz).

Amortization of computational costs across outputs in the sequence also motivated us to explore the use of higher frame rates (100 Hz) than the output frame rate (33 Hz) at the lower layers of the TDNN. We restrict the higher frame rates to the lower layers as this preserves the computational efficiency; and as the gains were negligible when increasing the frame rate even at the higher layers.

We compare the performance of the subsampled TDNN variants described above in Table 4.4. TDNN-F corresponds to the subsampled TDNN in 4.1.1 with exponentially decreasing frame rate, with layer depth, per output. TDNN-G corresponds to the subsampled TDNN with the same frame rate as the output at all the layers. TDNN-H corresponds to the subsampled TDNN with higher frame rates at lower layers and finally TDNN-I corresponds to a variant of TDNN-H with tuned temporal contexts.

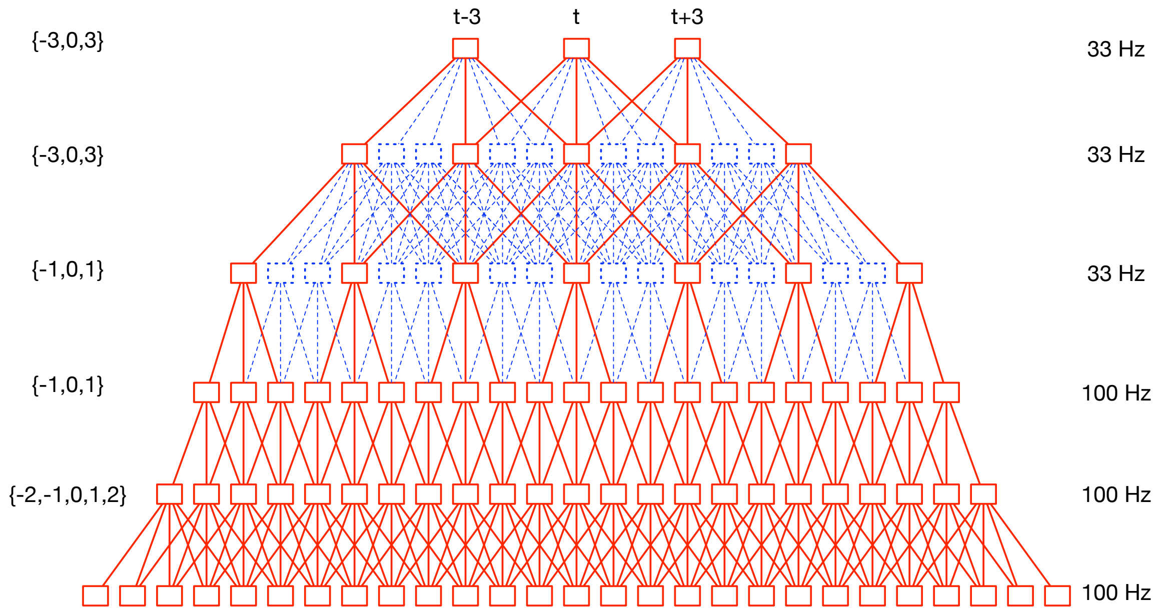


Figure 4.3: Computation in sub-sampled TDNNs trained with purely sequence discriminative training. Only sequences of outputs are computed; and these are computed at a reduced frame rate of 33 Hz. The number enclosed in $\{.\}$ on the left represent the temporal convolution kernel context and the numbers on the right represent the frame rate the convolution layer.

The configurations of the sub-sampled TDNNs described above and their performance is shown in Table 4.4. We specify the TDNN architectures in terms of the splicing indices which define the temporal convolution kernel input at each layer. e.g. {`-3,0,3`} means that the input to the temporal convolution at a given time step `t` is a spliced version of previous layer outputs at times `t-3, t, t+3`. It can be seen that using the higher frame rates at lower layers and tuning the temporal contexts of the layers (TDNN-I) provides 10.3% relative gain over the sub-sampled TDNNs proposed in [155] (TDNN-F). This performance comparison was done on the 300 hour Switchboard LVCSR task on the Hub '00 test set and the models were purely sequence discriminatively trained. A major change compared to the TDNNs described in previous sections of this chapter is the use of rectified linear unit (ReLU) [164] as the non-linearity.

Table 4.4: Comparison of sub-sampled TDNN architectures on the 300 Hr Switchboard LVCSR Task

Model	Layer-wise context							Network context	WER (%)		
									SWBD	CHM	Total
TDNN-F	{-2,-1,0,1,2}	{-1,2}	{-3,3}	{-7,2}	{0}	{0}	{0}	[-13, 9]	11.1	21.8	16.5
TDNN-G	{-2,-1,0,1,2}	{-1,2}	{-3,0,3}	{-3,0,3}	{-3,0}	{0}	{0}	[-12, 10]	10.5	21.9	16.3
TDNN-H	{-2,-1,0,1,2}	{-1,0,1}	{-1,0,1}	{-3,0,3}	{-3,0,3}	{-3,0}	{0}	[-13, 10]	10.3	20.7	15.5
TDNN-I	{-1,0,1}	{-1,0,1}	{-1,0,1}	{-3,0,3}	{-3,0,3}	{-3,0,3}	{-3,0,3}	[-15, 15]	9.6	19.9	14.8

∗ Please note that the overall temporal context of the neural network is kept similar, except in TDNN-D. There are 625 filters in each TDNN layer. However due to the change in temporal convolution kernel context there is a slight increase in parameters as we move from TDNN-F to TDNN-I.

From Table 4.4, comparing TDNN's F & G it can be seen that matching the output frame rate (33 Hz) at deeper layers does not improve the performance of the model significantly. However TDNN-H shows that significant gains can be achieved by operating the lower TDNN layers at frame rates closer to the input frame rate (100 Hz). Finally from TDNN-I it can be seen that increasing the right temporal context of the model can lead to significant gains. It can be seen that the higher frame rates have been restricted to the lower layers in TDNN-I. This was done as operating even the higher layers at this higher frame rate did not lead to gains while further increasing the computation cost. Comparing TDNN-F and TDNN-I an overall relative improvement of 10.3% can be observed.

In addition to increasing the frame rates at lower layers we also explored the use of average pooling, anti-aliasing filters before subsampling and per-feature-index temporal convolution layers before subsampling. All these variants resulted in no further improvements.

4.3 Summary

In this chapter we proposed a subsampled TDNN network which was shown to effectively model long temporal contexts, while significantly reducing the computational cost of conventional training *i.e.,* sequence discriminative training with frame-level pretraining. Further we modified this subsampled TDNN architecture to exploit the recent changes in neural network training which eliminate the need for cross-entropy pretraining.

Chapter 5

Low latency models with temporal convolution and LSTMs

In this chapter we compare the performance of the subsampled TDNNs with the bidirectional and unidirectional long short term memory neural networks (B)LSTMs. We discuss the latency of subsampled TDNN and (B)LSTMs, and show that BLSTMs which provide the best performance have significantly large latency. Finally we propose a low latency model which combines the strengths of the temporal convolution and unidirectional LSTMs, and performs similar to the BLSTMs.

5.1 Comparison of TDNNs with LSTMs

The best subsampled TDNN configuration TDNN-I from chapter 4 was compared with the stacked (B)LSTM models. The (B)LSTM models have three[1] layers of (B)LSTM. These models are denoted LFR-LSTM and LFR-BLSTM as all their layers operate at a low frame rate (LFR) of 33 Hz similar to [129] and [163] .

Based on our observations with TDNNs, we explored the use of higher frame rate (100 Hz) at lower (B)LSTM layers. This architecture is similar to the hierarchical subsampling networks, proposed in [165] and more recently applied in [166] and [167]. We denote these models as MFR-LSTM and MFR-BLSTM as they use a mixed frame rate (MFR) across layers. Figure 5.1 represents the computation in the MFR-LSTM.

Table 5.1: Performance comparison of TDNN, LSTM and BLSTM on the 300 Hr Switchboard LVCSR task

Model	WER (%)		
	SWBD	CHM	Total
TDNN-I	9.6	19.9	14.8
LFR-LSTM	10.1	21.0	15.6
LFR-BLSTM	9.6	19.2	14.5
MFR-LSTM	9.9	19.7	14.8
MFR-BLSTM	9.0	18.1	**13.6**

Table 5.1 compares the models discussed in this section with the best TDNN model. Firstly, it can be seen that operating the lower LSTM layers at a higher frame rate is beneficial. It results in a relative improvement of $\sim 6\%$ in BLSTM and $\sim 5\%$ in LSTM.

[1]The depth and other hyper-parameters of the LSTM and BLSTM models have been tuned. Further increase in depth leads to minor improvements with significant increase in computation costs.

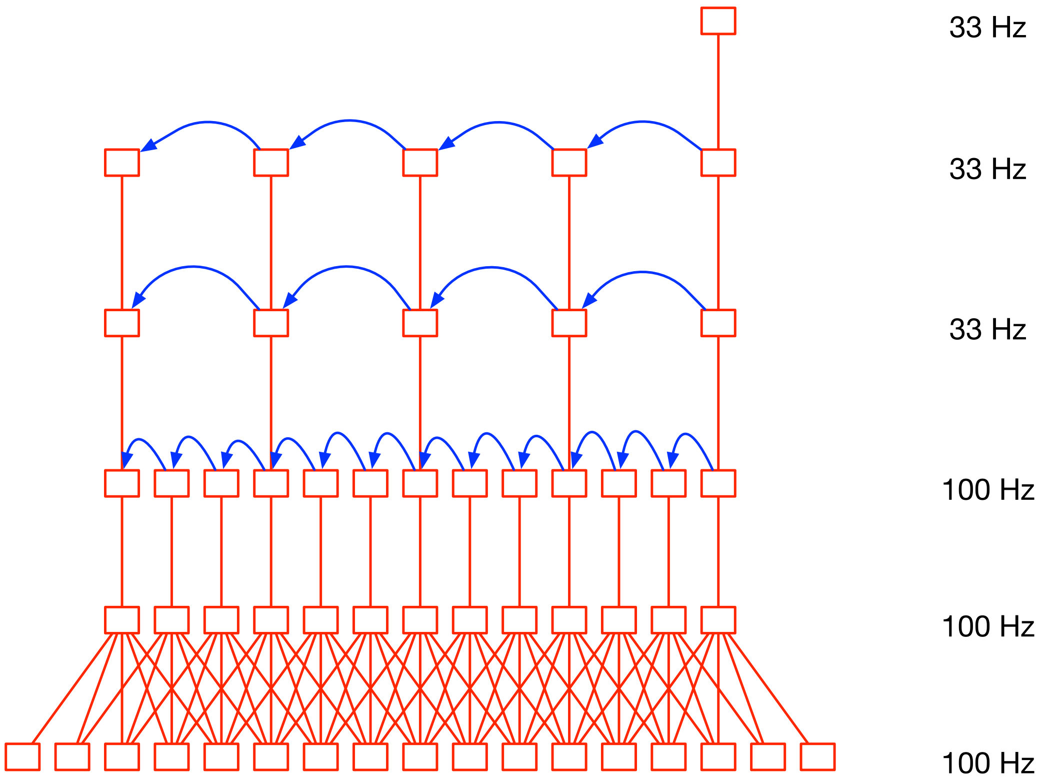


Figure 5.1: Dependencies among activations at various layers and time-steps in the stacked LSTM network with the lowest LSTM layer operating at 100 Hz

However the overall computational complexity increases by 30% during inference compared to the corresponding LFR models, for our input sizes of interest. Operating even the higher (B)LSTM layers at a higher frame rate did not lead to gains, while further increasing the computational complexity.

Secondly, it can be seen that both the TDNN and LSTM models perform worse than both the BLSTM models. The superior performance of the bidirectional recurrent models compared to their unidirectional counterparts can be attributed to the modeling of the future context which could not be matched even with the use of output delay in LSTMs.

5.1.1 Model latency

Latency of an acoustic model is affected by input context, chunk-width, chunk contexts and output delay. Further each TDNN layer adds to the latency due its kernel context. As the recurrent state of the forward LSTM can be propagated across chunks, the chunk left context does not add to the latency. Further with forward LSTMs chunk-width does not add to latency, as we can perform inference in frame-level increments. However when backward LSTMs are used inference is performed in chunk-level increments to amortize the backward LSTM cost over the entire chunk. Thus chunk-width and chunk right context add to the latency in BLSTM models.

As sequence level objective functions are typically computed over longer duration chunks ($\sim$ 1.5 seconds), the chunk-width and chunk right context can significantly add to the latency *e.g.* for the models in Table 5.1 the latencies are as shown below

- TDNN-I : kernel contexts $(10 + 10 + 10 + 30 + 30 + 30 + 30$ ms) = 150 ms
- LFR and MFR LSTMs : input context (20 ms) + label delay (50 ms) = 70 ms
- LFR and MFR BLSTMs : input context (20 ms) + chunk width (1500 ms) + chunk right context (500 ms) = 2020 ms

It can be clearly seen that the superior performance of BLSTMs comes at the expense of significant increase in latency. This has motivated a variety of research efforts to reduce their latency (see Section 2.2.1.2). All these variants rely on the use of backward recurrent layers to model the future context , they differ in the initialization of this backward recurrent layer. Hence inference is always done in chunk level increments to amortize the cost of the backward recurrent layer which ultimately keeps the latency significantly high.

5.2 Combining temporal convolution with LSTMs

In this section we propose neural networks which use temporal convolution to model the dependencies in the future context in unidirectional LSTMs. The use of temporal convolution enables inference with frame-level increments audio. Previous use of convolution operation in recurrent layers have been discussed in Section 2.2.3. Specifically Sainath *et al.,* [78] had proposed combining convolutional layers and recurrent layers; however as they used spectro-temporal convolution the requirement of spectral locality restricts the placement of convolutional layers to below the LSTM stack. In this work we focus on temporal convolution which does not require spectral locality and hence affords the exploration of more combinations, including the interleaving of convolutional and recurrent layers [2]. Temporal convolution was also used by Amodei *et al.,* [79] but just above or below the recurrent layer stack.

We explore three different ways of combining temporal convolution and LSTMs *viz.*,

- Stacking LSTMs over TDNNs (TDNN-LSTM-A)
- Stacking TDNNs over LSTMs (TDNN-LSTM-B)

[2]Recent experiments by Gaofeng Cheng have shown that combining spectro-temporal convolution with the architectures proposed in this chapter could further improve the results [168]. This work is in progress.

- Interleaving TDNNs and LSTMs (TDNN-LSTM-C)

Figure 5.2 represents the computation in the TDNN-LSTM-C network. It can be seen that all the LSTM layers, which can be computationally expensive, operate at the 33 Hz frame rate.

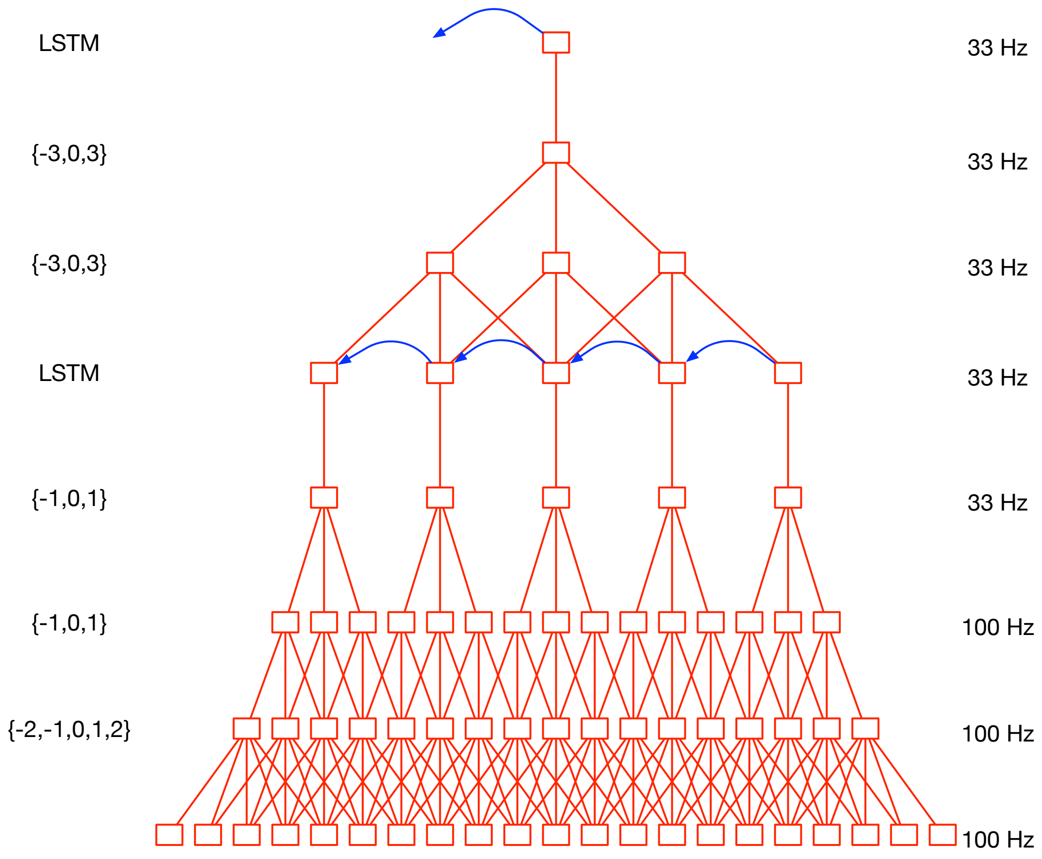


Figure 5.2: Dependencies among activations in a stacked TDNN-LSTM network with interleaved temporal convolutions. The convolution kernel input contexts are on left and the layer-wise frame rates are on the right.

Further, to compare the benefits of performing temporal convolution in BLSTM models, we also interleave temporal convolution with the forward and backward LSTMs (TDNN-BLSTM-A) or with forward and backward LSTM stack *i.e.,* the BLSTM layer (TDNN-

BLSTM-B).

As only the lower TDNN layers operate at a 100 Hz frame rate in TDNN-LSTM-C and TDNN-BLSTM-A, we also verify if additional gains can be had by operating even the lowest recurrent layer at 100 Hz frame rate, similar to MFR-(B)LSTM. These models are denoted as TDNN-LSTM-D and TDNN-BLSTM-C, respectively.

5.2.1 Recurrence scaling

As seen from Section 5.1.1 it is critical to carryover the forward LSTM state to enable inference in frame level increments in unidirectional LSTMs and thus reduce the latency. In order to reduce the mismatch between the left contexts seen during training and inference, when models trained on fixed length left contexts are used in inference with infinite context, *i.e.,* with the entire left context of the utterance seen till the current frame, recurrence scaling is employed. In recurrence scaling a fixed constant factor (0.8) is multiplied with the history state to enable forced forgetting in the LSTMs. This reduces the mismatch in left contexts and can ensure better generalization at the cost of reducing the temporal modeling power of the LSTMs.

5.3 Results

In this section we provide a comparison of acoustic models on the 300 hour SWBD LVCSR task; Table 5.2 presents this comparison. Table 5.3 compares performance of systems trained with and without recurrence scaling.

We perform decodes with two different posterior estimation methods, *Matched Inference* where the state of the network for a chunk is estimated using a chunk left context, similar to the training; or *State-saving Inference* where the state is copied from the final state in the previous chunk.

From Table 5.2: TDNN-LSTMs B and C perform better among A, B and C; while C has lower real-time factor (RTF). From Tables 5.2 and 5.3: there is a difference in performance between TDNN-LSTM-C and MFR-BLSTM, but this slightly reduces when trained without recurrence scaling.

From Table 5.2: Interleaving TDNNs with BLSTMs rather than forward and backward LSTMs separately was better (TDNN-BLSTM-A vs B); both these models perform better than LFR-BLSTM. However there was no benefit compared to MFR-BLSTM, in terms of performance, though RTFs are lower than MFR-BLSTM.

Operating the lowest (B)LSTM layer in TDNN-(B)LSTMs at 100 Hz, *i.e.*, TDNN-LSTM-D and TDNN-BLSTM-C led to performance gains with additional computational cost, when compared to TDNN-LSTM-C and TDNN-BLSTM-A, respectively.

From Table 5.3 : It can be clearly seen that recurrence scaling helps better generalize to longer sequence lengths i.e., for state-saving inference. We are currently exploring other mechanisms to better generalize to chunk-widths not seen during training. These include use of frame-level dropout of recurrent states [169] with longer chunk-widths.

Table 5.2: Performance comparison of various models in the 300 Hr SWBD LVCSR task†

Model	Architecture*	Latency (ms)	Matched Inference			State-saving Inference		
			WER(%)		RTF	WER(%)		RTF
			SWBD	Total		SWBD	Total	
TDNN-I	$T^{100}T^{100}T^{100}TTTT$	150	9.6	14.8	0.8	9.6	14.8	0.9
LFR-LSTM	$L_fL_fL_f$	70	10.1	15.6	2.5	10.7	16.2	1.2
MFR-LSTM	$L_f^{100}L_fL_f$	70	9.9	14.8	2.7	10.2	15.3	1.8
TDNN-LSTM-A	$T^{100}T^{100}T^{100}TTTTL_fL_fL_f$	200	9.5	14.6	2.7	9.7	14.8	1.8
TDNN-LSTM-B	$L_f^{100}L_f^{100}L_f^{100}T^{100}T^{100}T^{100}TTTT$	200	9.4	14.3	4.0	9.3	14.3	2.3
TDNN-LSTM-C	$T^{100}T^{100}T^{100}L_fTTL_fTTL_f$	200	9.2	14.2	3.7	9.4	14.4	1.9
TDNN-LSTM-D	$T^{100}T^{100}T^{100}L_f^{100}TTL_fTTL_f$	200	9.0	13.9	4.8	9.4	14.4	2.4
LFR-BLSTM	$[L_f, L_b], [L_f, L_b], [L_f, L_b]$	2020	9.6	14.5	4.7			
MFR-BLSTM	$[L_f^{100}, L_b^{100}][L_f, L_b][L_f, L_b]$	2020	9.0	13.6	6.6			
TDNN-BLSTM-A	$T^{100}T^{100}T^{100}[L_fT, L_bT][L_fT, L_bT][L_f, L_b]$	2170	9.2	14.1	4.6			
TDNN-BLSTM-B	$T^{100}T^{100}T^{100}[L_f, L_b]TT[L_f, L_b]TT[L_f, L_b]$	2170	9.1	13.8	4.7			
TDNN-BLSTM-C	$T^{100}T^{100}T^{100}[L_f^{100}T^{100}, L_b^{100}T^{100}][L_fT, L_bT][L_f, L_b]$	2130	9.0	13.8	9.6			

* forward LSTM - L_f, backward LSTM - L_b, TDNN - T, default layer frame-rate - 33 Hz and other frame rates are specified in the super-script.
[., .] - layers which operate on the same input and whose outputs are appended e.g.BLSTM = $[L_f, L_b]$.
L_f, L_b : cell size - 1024, recurrent and non-recurrent projections-256;
TDNN filters/layer : in TDNN-LSTMs - 1024 and in TDNN-BLSTMs - 512
TDNN layer contexts are same as TDNN-I (see Table 4.4)

† State-saving inference for BLSTMs is difficult to implement in our framework, so we do not present these results.

Table 5.3: Impact of recurrence scale : 300 Hr SWBD LVCSR Task

Model	Total WER on Hub'00 (%)			
	Without scaling		With scaling	
	MI	SSI	MI	SSI
TDNN-LSTM-A	14.2	15.6	14.6	14.8
TDNN-LSTM-C	13.9	16.0	14.2	14.4
MFR-BLSTM	13.5	-	13.6	-

MI : Matched Inference SSI : State saving Inference

5.4 Summary

In this chapter we introduced the neural networks which combine the strengths of temporal convolution and unidirectional LSTMs to enable effective modeling of long temporal contexts, superior to LFR-BLSTMs and comparable to MFR-BLSTMs, while restricting the overall latency to 200 ms.

Chapter 6

Far field recognition

Far-field speech recognition has received a lot of focus in the past few years due to the advent of personal assistant devices like *Amazon Echo* [170], *Google Home* [101] and *Apple Homepod* [171]. Far-field speech is challenging due to the phenomena of reverberation. In reverberant environments the reflections of a signal affect the signal over several time frames. These long term interactions are due to multiple paths from each sound source to the microphone, each with its own delay. To tackle these longer term interactions between the direct speech signal and the corrupting sources, speech recognizers have to account for long-term acoustic context [172]. Hence it is of interest to test the proposed models in the context of reverberant speech recognition.

In this chapter we will compare the acoustic models in two different far-field speech recognition tasks. These are the AMI task [173, 174, 175] and the ASpIRE task [99]. Section 6.1 discusses the AMI task while Section 6.2 describes the ASpIRE task.

In this chapter in addition to comparing the proposed acoustic models with (B)LSTMs we describe the contributions made in far-field acoustic modeling as part of this thesis. In Section 6.1 we initially detail the importance of parallel clean audio in acoustic modeling for robust ASR, and propose a method to reduce performance gap between systems trained with and without parallel clean audio. In Section 6.2 we discuss the importance of voice-activity detection for iVector estimation in reverberant environments and also show that volume perturbation can be used to significantly reduce the impact of severe mismatch in audio gain levels.

6.1 AMI far-field speech recognition task

There are three LVCSR tasks [173, 175] designed using the AMI meeting corpora [174]. These are the individual headset microphone (IHM), single distant microphone (SDM) and multiple distant microphone (MDM) tasks; named based on the type of audio used in the creation of the *train*, *dev* and *eval* sets.

6.1.1 Far field ASR and parallel audio

The AMI corpus, with parallel speech recordings from all these microphones, provides an opportunity to analyze the importance of alignment quality in far-field speech recognition systems. In addition to the three standard AMI LVCSR systems, which use alignments from the HMM-GMM systems trained using the corresponding audio, we also trained systems using alignments generated from the IHM audio. Table 6.1 summarizes the results of the 8 such LVCSR systems which were trained with the cross-entropy criteria. It can be seen that there is a significant reduction in word error rate (WER) (7.75% relative, on average) when using alignments from IHM audio. Further these relative improvements increase when using better acoustic models *i.e.,* TDNN vs BLSTM acoustic models.

Table 6.1: Comparison of AMI LVCSR systems trained with cross-entropy criterion, using close-talk and distant microphone alignments

Model	LVCSR task	Alignments	WER (%)	
			dev	*eval*
TDNN	SDM	SDM	45.8	50.3
	SDM	IHM	41.8	46.6
	Rel. Change		8.7%	7.3%
	MDM	MDM	41	44.7
	MDM	IHM	38.2	42
	Rel. Change		6.8%	6.0%
BLSTM	SDM	SDM	42.5	45.6
	SDM	IHM	38.5	41.8
	Rel. Change		9.4%	8.3%
	MDM	MDM	38.6	41.0
	MDM	IHM	35.5	38.3
	Rel. Change		8.0%	6.6%

When such parallel recordings are available, the alignments used for training the acoustic models can be generated from close-talk microphone audio recordings. However in typical large data scenarios, where actual far-field audio is collected, assuming the availability of close-talk microphone recordings is not practical.

In this section, we identify the possible reasons for the performance difference between the ASR systems that are trained using alignments generated from distant microphone recordings, and those trained with alignments generated from parallel close-talk microphone recordings. Further, we propose a two pronged strategy to reduce this performance gap. Firstly, we use the lattice-free maximum mutual information (MMI) objective function [128], which is tolerant to minor mis-alignment errors, to train the neural networks from random initialization. Secondly, we use lattice based quality estimate for selecting reliable utterances for training. The combination of these two techniques reduces the performance gap from $\sim 8\%$ to $\sim 1.5\%$. We present results on both single distant microphone and multiple distant microphone scenarios of the AMI LVCSR task.

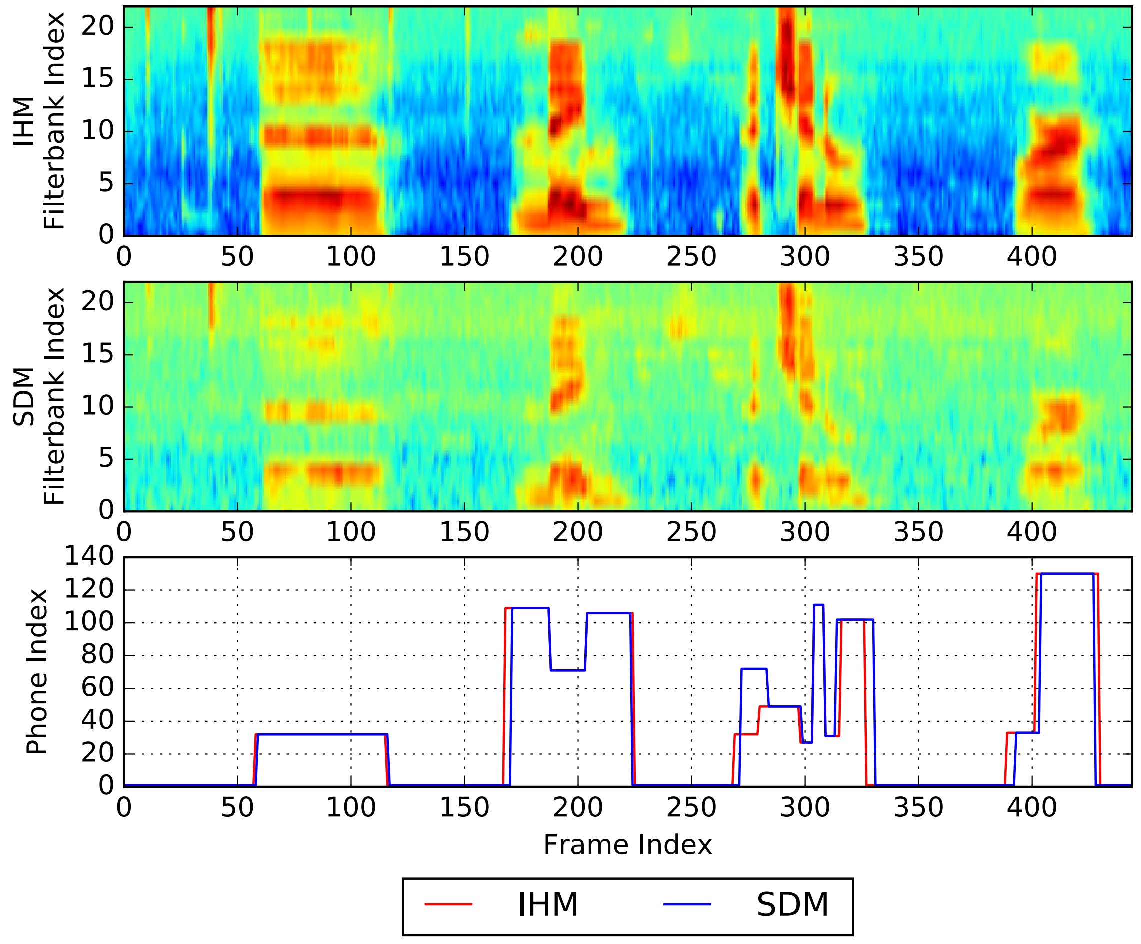
6.1.2 Analysis of alignment errors

Motivated by the observations in Table 6.1, we performed a comparison of alignments generated from IHM and SDM systems. We randomly sampled utterances from the AMI corpus and identified some prominent categories of errors.

6.1.2.1 Minor mis-alignment errors

A majority of the errors were minor mis-alignment errors. Figure 6.1 shows the log mel filter-bank coefficients from the IHM and SDM recordings; and compares the phone alignments generated by the corresponding HMM-GMM systems. It can be seen that there are just minor differences between these two alignments. The significant difference in alignments, between frames 250 and 300, occurs due the choice of different pronunciations (*EY* vs *AH*) for the word *"a"* by the IHM and SDM systems.

Figure 6.1: TS3006b_MTD021PM (1668.61-1673.08 seconds) *"Uh name a channel or"*

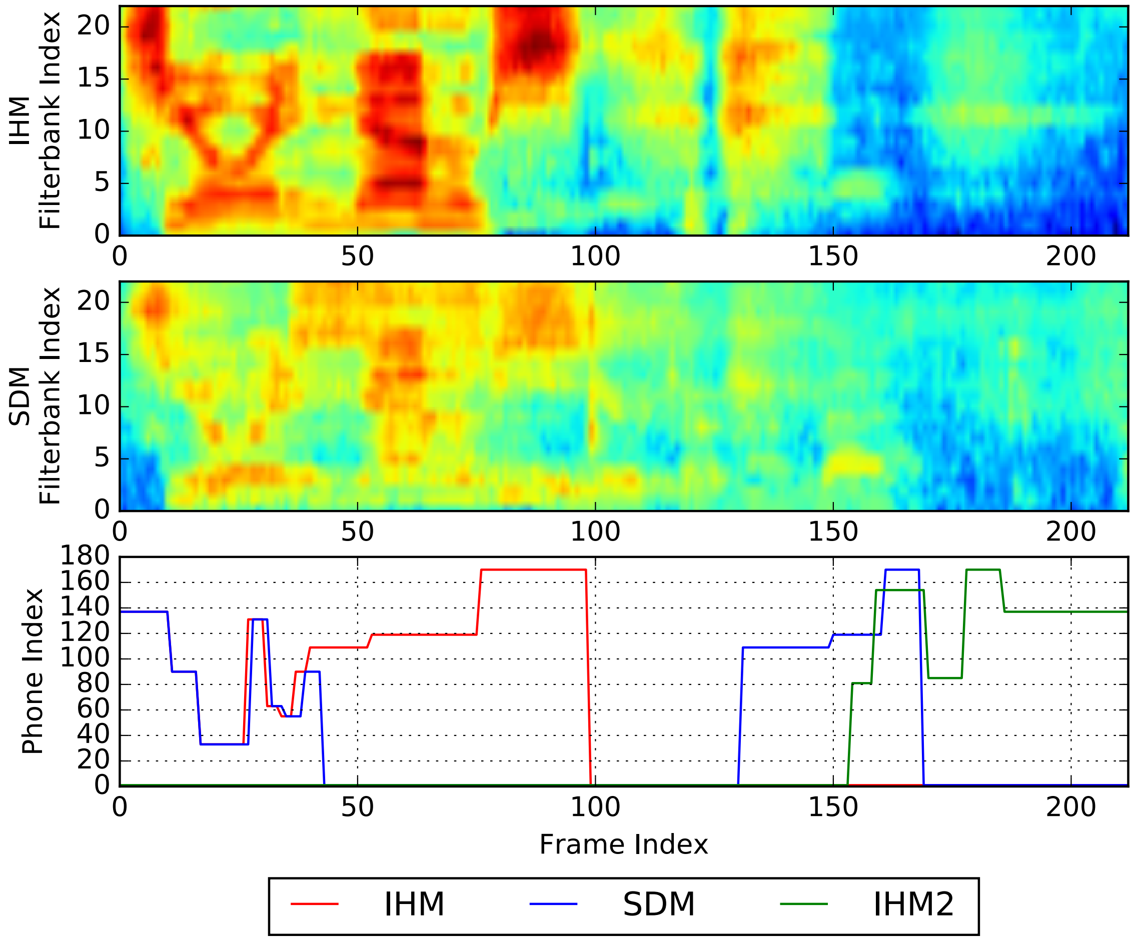


6.1.2.2 Speaker overlap errors

In a significant number of utterances, there were speaker overlaps in both IHM and SDM audio. For the IHM audio, the transcription corresponded to dominant speaker, as expected. However, this was not necessarily the case in SDM audio. These errors worsened the quality of the SDM alignments. Figure 6.2 represents one such utterance. In this plot the green line shows the alignment for the competing speaker using his IHM audio. It can be seen that the speech of this speaker, identified by the non-zero green line, distorts the

SDM alignment. Figure 6.3 shows the amount of training data with different degrees of overlaps.

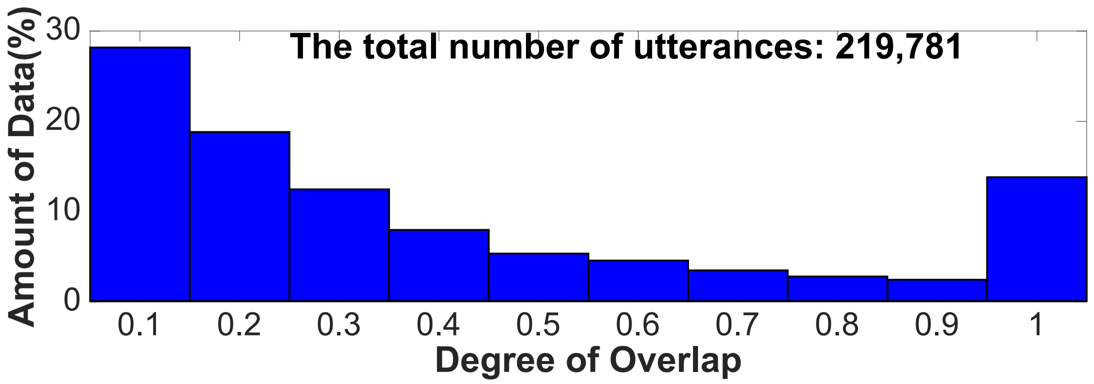
Figure 6.2: TS3009d_MTD033PM (1991.99-1994.14 seconds) :
Transcribed speech : *"She already knows"*
Overlapping untranscribed speech: *"Who is she you're talking about"*



6.1.2.3 Transcription errors

As in other databases, there were minor transcription errors. However, AMI corpora had errors where there were significant portions of untranscribed non-overlapping speech

Figure 6.3: Histogram of data with different degree of overlaps $\text{DegreeOfOverlap}(Utt1) = \frac{\max_{Utt2} \text{Duration}(Utt1 \cap Utt2)}{\text{Duration}(Utt1)}$.

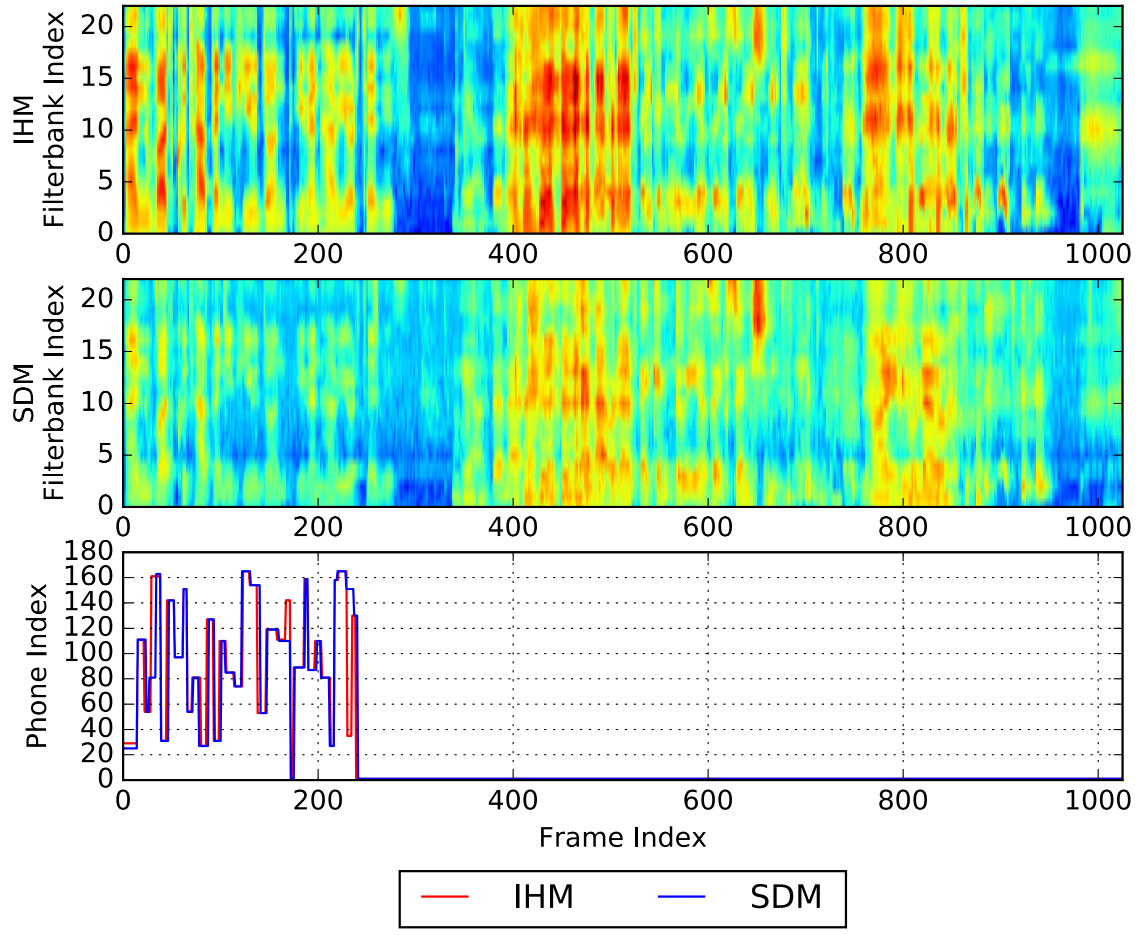


in the utterances. Figure 6.4 provides one such example. Significant portion of the signal corresponds to a second talker's response and it is untranscribed. Further, both the IHM and SDM systems align this trailing speech to silence.

6.1.3 Two pronged strategy

We propose a two pronged strategy to tackle the errors described in Section 6.1.2. Firstly, to make the learning algorithm robust to minor mis-alignment errors, we use the lattice-free MMI objective function [128]. This approach is described in Section 6.1.3.1. Secondly, we filter the utterances that might have speaker overlap or transcription errors. To accomplish this, we utilize a quality measure for utterances proposed by Manohar and Povey *et al.,* and reported in [176]. Description of this quality measure is reproduced in Section 6.1.3.2 for the reader's convenience.

Figure 6.4: IN1002_MIO076 (686.66-696.94 seconds) :
Transcription:*"And what could happen if you don't even have your"*
Untranscribed trailing speech: *"Then I would have taken two year's extension"*



6.1.3.1 Alignment error tolerant objective function

In Section 3.3.2 the purely sequence discriminative training method proposed by Povey *et al.* [128] had been introduced. This new method of training has been shown to provide significant improvements compared to conventional sequence discriminative training methods across different LVCSR tasks. However, our interest in this objective function arises from the fact that it is inherently tolerant to alignment errors, as we can specify a range of time frames for a particular context-dependent phone state using a desired tolerance. In this section, we highlight this particular aspect of the cost function. Readers are encouraged to refer to [128] for more details about this objective function.

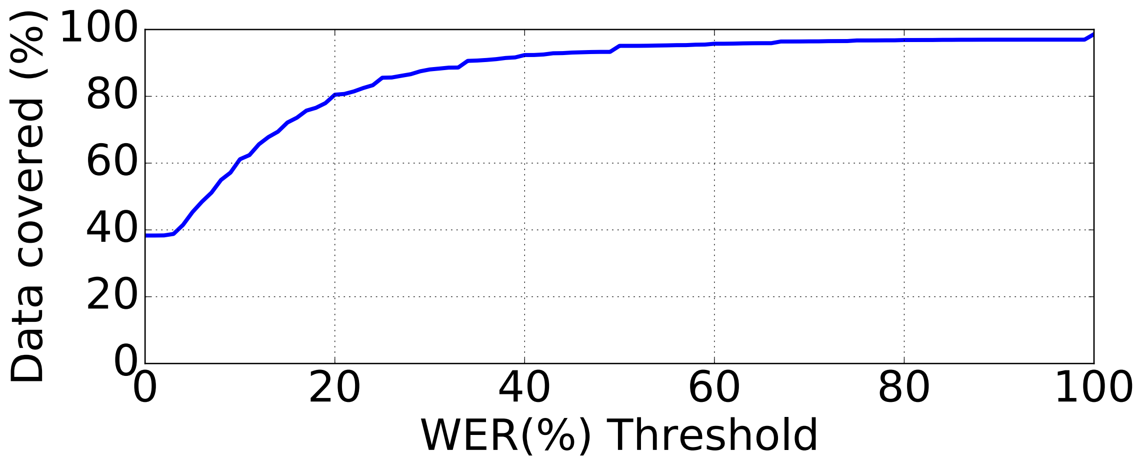
The derivative computation for the MMI objective requires the computation of state occupancy statistics using the forward-backward algorithm on the numerator and denominator graphs [122]. The denominator graph is built using a phone n-gram language model. The numerator graph creation is of relevance to this section.

Prior to training the neural net, a GMM-based system is used to generate lattices representing alternative pronunciations of the training utterances. These lattices are processed into phone graphs and then compiled into utterance-specific Finite State Acceptors (FSAs) as for conventional training. Separately, the lattices are also processed into frame-by-frame masks of what phones are allowed to appear on what frames: a user-specifiable tolerance allows a phone to appear slightly before or after where it appeared in the lattice. As the frame-by-frame phone mask built from the lattices has a tolerance, we expect the gradient computation to be tolerant to minor misalignment errors. We found a 50 ms tolerance to be optimal for both reverberant and telephone speech tasks.

6.1.3.2 Filtering based on quality estimate

Manohar and Povey proposed the use of lattice based quality estimates to resegment and filter reliable parts of an utterance [177]. The lattice oracle WER based quality estimate described here is part of this clean-up procedure and has been used here to filter the

Figure 6.5: Amount of data retained with each lattice oracle WER threshold



utterances without resegmentation. We describe this quality estimate briefly for the reader's convenience.

Given a training utterance and its corresponding transcript, the procedure to find the lattice oracle WER is given in Algorithm 1. For step 4, we use the same algorithm as in [178] for finding the edit distance between a lattice and a reference, but replace the lattice forward-backward with a Viterbi search.

Algorithm 1 Procedure to compute lattice oracle WER

```
Input: Utterance $u$ with transcript $\mathcal{R}$
Input: $\mathcal{W}$ = List of 100 most common words in the training set
1: procedure LATTICE ORACLE($u, \mathcal{R}$)
2:     Build a biased unigram language model using the words in $\mathcal{R}$ and $\mathcal{W}$
3:     Decode the utterance $u$ using the language model to get a lattice $\mathcal{L}$
4:     Find a path in the lattice $\mathcal{L}$ that has the minimum Levenshtein edit distance from
       $\mathcal{R}$. Let its score be $d$.
Output: $\frac{d}{|\mathcal{R}|}$, where $|\mathcal{R}|$ is the number of words in $\mathcal{R}$.
5: end procedure
```

We make use of the lattice oracle WER in a simple filtering scheme. Figure 6.5 shows the percentage of data covered under different oracle WER thresholds. It can be seen that

Table 6.2: Impact of alignment quality based filtering on TDNN acoustic models trained with lattice-free MMI critera for SDM LVCSR task

FER (%) threshold	Data retained	WER (%)	
		dev	*eval*
50	82	44.2	48.1
60	95	43.1	46.9
70	96	42.8	46.6
80	97	43.6	47.2
All	All	43.2	47.3

$\sim 95\%$ data can be preserved with WER thresholds around 50%. Utterances with larger oracle WER, including the 5% that has greater than 100% oracle WER, predominantly have either speaker overlaps (Section 6.1.2.2) or transcription errors (Section 6.1.2.3).

One drawback of this approach is that short segments that have a single word (*e.g.*, *"Yeah"*, *"Okay"*) in the reference will almost always be given an oracle WER of 0, because if that word is in the decoded lattice, it will be picked up by the Viterbi search. However, these amount to very little data.

6.1.3.3 Filtering Based on Frame-Level Alignment Quality

As use of IHM alignments reduced the WERs significantly, we treated these as ground truth labels and measured the duration normalized Levenshtein distance between the per-frame phone alignments of SDM and IHM systems. This frame error rate (FER) was used to filter out utterances in the SDM task (see Table 6.2), resulting in similar improvements as with lattice oracle WER.

The utterance in Figure 6.1 had a lattice oracle WER of 20.00% and an FER of 7.2%. The utterance in Figure 6.2 had a lattice oracle WER of 0.00 and an FER of 46.48%. The utterance in Figure 6.4 had a lattice oracle WER of 20.00% and and FER of 7.02%.

6.1.4 Experimental Setup

The overall experimental setup is similar to the one described in Chapter 3. In this section we describe the changes relevant to the AMI LVCSR task.

The HMM-GMM systems for generating the alignments and lattices, used to train the neural network acoustic models, are as described by Swietojanski *et al.,* in [175]. However, unlike in [175], we perform speaker-adaptive training of the HMM-GMM systems for all the three tasks, as we found the alignments from SAT HMM-GMM systems to be beneficial for neural network training on all three tasks.

The MDM LVCSR systems have an additional stage of beam-forming to combine the audio captured from different channels of the distant microphone. The BeamformIt toolkit [179] was used for delay-sum beamforming.

To train the SDM and MDM LVCSR systems with alignments generated from IHM data, we identified parallel segments in IHM audio corresponding to the utterances in SDM/MDM data. The IHM SAT HMM-GMM system was used to generate alignments and lattices from these parallel utterances.

The lattice-free MMI technique uses fixed length chunks of 1.5 seconds to perform sequence training. As nearly 50% of the utterances in the AMI corpus were less than 1.5 seconds long we combined neighboring utterances to reach the 1.5 second minimum utterance length.

6.1.5 Results

Table 6.3 contrasts the WER for various training & test conditions with different training criteria/data-sets.

Table 6.3: Comparison of rel. changes in WER(%) when using alignments from IHM and SDM/MDM data to train TDNN acoustic models

LVCSR task	Alignments	Cross-entropy		Lattice-free MMI		Lattice-free MMI + Data filtering	
		dev	*eval*	*dev*	*eval*	*dev*	*eval*
SDM	SDM	45.8	50.3	43.2	47.3	42.8	46.1
SDM	IHM	41.8	46.6	41.3	45.3	41.6	45.4
Rel. Change		8.7%	7.3%	4.4%	4.2%	2.8%	1.5%
MDM	MDM	41	44.7	40.5	43.2	38.5	41.5
MDM	IHM	38.2	42	38.1	42	38.1	41.5
Rel. Change		6.8%	6.0%	5.9%	2.78%	1.0%	0%
IHM	IHM	24.4	25.1	22.6	22.5	22.4	22.4

First, compare across the row for the SDM task with IHM training alignments to note that the MMI training and data filtering have only a modest impact when parallel clean+noisy recordings are available: minimal difference in *dev* WER and small improvement in *eval* WER. The same is also true in the MDM task with IHM training alignments.

More importantly, compare across the row for SDM task with SDM alignments to note that MMI training results in a significant reduction in WER relative to cross-entropy training, and the data filtering step yields further gains. The same observation holds for the MDM task with MDM alignments.

Finally contrasting the IHM training alignments with the SDM training alignments for the SDM task, note that while the cross-entropy training criterion suffered a $7\% - 8\%$

degradation in WER relative to IHM alignments, the MMI criterion by itself limits the WER degradation to about 4%, and the data filtering brings down this difference to about 2%. The same trend holds even more strongly for the MDM task – the relative degradation from IHM alignments to MDM alignments is reduced from $6\% - 7\%$ to $0\% - 1\%$.

This last set of results supports the main claim made in this section, namely that the proposed method – using an alignment-tolerant MMI training objective after filtering out the most problematic part of the training data – mitigates strongly against degradation in WER when parallel clean+noisy speech is not available for training acoustic models.

Table 6.4 compares the impact of using different lattice oracle WER thresholds on the acoustic model quality, as measured in WER on *dev* and *eval* sets. It can be seen that at 45% lattice oracle WER threshold we see the maximum gains. This preserves $\sim 95\%$ of the data in the train set of the corpora. It can be seen that the same data filtering step does not have a significant impact on the acoustic models trained with the cross-entropy criterion.

Table 6.4: Impact of data filtering on TDNN acoustic models trained with cross-entropy or lattice-free MMI criteria, for SDM LVCSR task

WER threshold (%)	WER (%)			
	Cross-entropy		Lattice-free MMI	
	dev	*eval*	*dev*	*eval*
40	45.4	50.3	43.1	46.9
45	45.5	50.1	42.8	46.1
50	45.5	50.1	42.8	46.6
All	45.8	50.3	43.2	47.3

Our experiments with the lattice-free MMI objective function did not result in gains with BLSTM models on this task, though [128] suggests that gains should be expected. We attribute this so the small amount of data in the AMI task, and are currently investigating hyperparameter settings for BLSTM training that are most suitable for this task.

6.1.6 Comparison of acoustic models

In this section we will compare the acoustic models proposed in this thesis *viz.,* subsampled TDNN and subsampled TDNN + LSTM acoustic models with the (B)LSTM acoustic models. This comparison is performed in the best setting described in the previous section *i.e.,* using lattices generated using the IHM data for SDM and MDM acoustic models. For these comparisons we use matched inference *i.e.,* the neural network posteriors in the recurrent models are estimated using a fixed context to the left of the acoustic chunk, as the major comparisons are done with BLSTM models. These results are presented in Table 6.5. The architectures of these models are same as those described in Table 5.2.

It can be seen that both the LFR and MFR BLSTMs perform better than the TDNN. The TDNN-LSTM model performs slightly better than both the BLSTM models. Once again there were no significant gains when combining TDNN and BLSTM models.

Table 6.5: Performance comparison in the AMI LVCSR task with matched inference‡

Model	WER (%)					
	IHM		SDM		MDM	
	Dev	Eval	Dev	Eval	Dev	Eval
TDNN-D	21.7	22.1	39.9	43.9	36.6	40.1
LFR-BLSTM	21.0	20.9	38.8	42.0	35.4	38.4
MFR-BLSTM	20.6	20.3	37.4	40.5	34.5	37.3
TDNN-LSTM-C	20.8	20.5	37.3	40.4	34.1	36.8
TDNN-BLSTM-A	20.7	20.7	37.0	40.4	34.2	36.6

6.2 ASpIRE far-field speech recognition task

The ASpIRE LVCSR task has been created using data released as part of the ASpIRE far-field recognition challenge help by IARPA [99]. This challenge uses the English portion of the Fisher database [161] for acoustic and language model training. Two data sets *dev* of 5 hrs and *dev-test* of 10 hrs were provided as part of ASpIRE challenge. Each set is composed of 10 minute recordings. The end points for the speech portions of the recording were also provided for the *dev* set. However in order to emulate the decoding scenario of *dev-test*, we report performance on *dev* set without the knowledge of segment information.

A major challenge in this task has been the severe mismatch between the training data which is telephone speech and test data which is rerecorded reverberant speech. To tackle this mismatch reverberation was simulated in the training data as described in Section 3.2.2. Each utterance in the 1800 hour Fisher English database was reverberated three times with three different room impulse responses. The reveberated training data was ~ 5400 hours $(3 * 1800)$. It can be seen that training the acoustic models in a timely manner requires parallel training and acoustic models which are amenable for parallelization during training, such as subsampled TDNNs, are of interest.

In this section we detail the changes in the experimental setup specific to ASpIRE task. A critical change in the ASpIRE LVCSR task compared to the other LVCSR tasks described in this thesis is the unavailability of utterance endpoints in the recordings. This impacts the iVectors which are sensitive to reverberation and noise in the data. Section 6.2.1 describes the iVector extraction strategy suitable for far-field recognition tasks.

Further we will show that the impact of using the enhanced lexicon (see Section 3.6.3) in this robust speech recognition task is significantly more compared to other LVCSR tasks [140]. Finally we will show the impact of using RNN-LM language models and sequence discriminative training.

For these comparisons we will use the conventionally trained acoustic models, *i.e.,* cross-entropy pretraining and sequence discriminative training; and further restrict our attention to TDNNs as they are significantly faster to train compared to the (B)LSTM acoustic models.

6.2.1 iVector Extraction

In this section we describe the iVector estimation process adopted during training and decoding. We discuss issues in estimating iVectors from noisy unsegmented speech recordings, and in using these noisy estimates of iVectors as input to neural networks.

We noticed that the iVector adaptation was not sufficiently effective in adapting to test signals that had substantially different energy levels than the training data. For the results reported here, this issue was resolved by normalizing the test-signal energies to be the same as the average of the training data.

6.2.1.1 iVector Extraction during training

The iVector estimator was trained on a 100 hour subset of training data: this includes the training of the Gaussian mixture model used for the UBM, and the estimation of the total-variability (T) matrix. Then, for the entire training data, iVectors were estimated. In order to ensure sufficient variety of the iVectors in the training data, rather than estimating a separate iVector per speaker we estimate them in an online fashion, where we only use frames prior to the current frame (for some arbitrary ordering of the utterances). We reset this history every two utterances, so that we still have some training-data variety even when there are only a few speakers.

6.2.1.2 iVector extraction during decoding

During decoding, the constraints of online extraction were not enforced and iVectors were estimated in an offline fashion from statistics accumulated over fairly large portions of the speaker's data (at least 60 seconds).

The prior term in the iVector extraction is quite important when applying these iVector based methods to data that is dissimilar to the training data. In our iVector estimation we always scale the per-frame posteriors by 0.1 (equivalent to scaling the prior term up by 10). For the ASpIRE challenge we made a further modification: if the total count of (scaled) statistics for iVector extraction exceeds a predefined limit (75 for these experiments), we scale the statistics down to that value, which again is equivalent to scaling the prior term up. Due to the posterior scale of 0.1, this effect kicks in after we exceed 750 frames of features.

6.2.1.3 iVectors from reliable speech segments

In the current LVCSR task (see below), audio recordings 5-10 minutes in length were provided without speech end-point information. The recordings had long durations of contiguous silence, similar to single channel recordings of conversational telephone speech. We found empirically that excluding the silence from the statistics for iVector estimation was very helpful. Even keeping a small amount of silence around every speech segment (similar to the amount we saw in training) was harmful; possibly the nature of the silence in the ASpIRE test data was so different from what was seen in the artificially reverberated and noise-added training data, that it affected the iVector in unexpected ways. Hence only feature vectors from the speech segments were used for i-vector estimation. We explored two techniques to detect speech segments, which are described below.

1. iVectors using two-pass decoding

In the first method, we perform a first-pass decode of the audio data using iVectors derived from both speech and non-speech regions. Reliable speech segments are identified from this first-pass decode. Audio segments corresponding to words with confidence measures of 1.0 (derived from lattice posteriors) and with durations less than one second were considered reliable (over half the words recognized had a confidence of at least 1.0). We also excluded the words “mm” and “mhm”. A second pass decode was then performed using the iVectors estimated from these reliable speech segments. This led to 8.9% relative improvement in WER, versus using all the data for iVector estimation.

2. iVectors using Voice Activity Detection (VAD)

As two-pass decoding is computationally expensive, we attempted a GMM-based Voice Activity Detection (VAD) to detect regions of speech. This technique was implemented by *Vimal Manohar*. The VAD method used is a hybrid feature and model-based method inspired by [180]. It works by training a HMM-GMM system with 3 GMMs - *Silence*, *Speech* and *Noise* on approximately 10 minute long chunks of the audio recordings. The features used with the GMMs are 12 (excluding C0) mean-normalized MFCCs along with their deltas and delta-deltas and zero-crossing rates along with its delta and delta-delta. The GMMs are initially bootstrapped using frame-alignments of the augmented training set described in the previous section. For this, the phones are mapped into 3 classes – silence, speech and noise. The GMMs are iteratively trained using Viterbi decoding followed by re-estimation.

For the first few iterations of training, only the low-energy and high zero-crossing rate frames from the non-speech frames are selected for *Silence* GMM and *Noise* GMM training respectively. The later iterations use all the frames of the respective classes. We have used the same training procedure as in [180]. The number of gaussians in each model is increased every iteration until the number of Gaussians for *Silence*, *Noise* and *Speech* are 7, 18 and 16 respectively.

A Bayesian information criterion (BIC) is used to determine if the *Noise* GMM is to be retained. If the *Noise* GMM is to be removed, then the entire process is repeated using only the *Silence* and *Speech* GMMs bootstrapped from the augmented training data.

The regions selected as speech by the VAD are used for iVector estimation and a single-pass decode is performed using these iVectors. With this method, we were able to improve the speed over the two-pass decoding system by a factor of ~ 2, while keeping the WER degradation on the dev set to 1.2%, relative.

Table 6.6 compares systems trained with and without iVectors, and different types of iVector extraction methods. We tried four different ways to extract iVectors:

- extracting iVectors from both speech and non-speech frames,
- extracting iVectors from speech frames, but in a online mode (i.e., for the current frame, only use speech frames before the current frame),
- extracting iVectors from speech frames, but in a offline mode (a first pass decoding was used to identify the speech regions) and
- extracting iVectors from speech frames computed from VAD.

Table 6.6: Comparison of systems with and without iVectors

Acoustic Model	dev WER
TDNN B w/o iVectors	34.8
TDNN B + iVectors[1]	33.8
TDNN B + iVectors[2]	33.1
TDNN B + iVectors[3]	30.8
TDNN B + iVectors[4]	31.2

[1] estimated on speech and non-speech frames
[2] estimated on speech frames online
[3] estimated on speech frames offline
[4] estimated on speech frames from VAD

From the table it's clear that it is beneficial to use iVectors in our system, and it is also critical to extract iVectors only from speech frames.

6.2.2 Impact of volume perturbation

Our iVector based neural network system relies on the neural network to learn the necessary normalization, based on mean shifts captured in the iVector (see Section 3.1.3). However in well curated audio databases there is low variance in audio volume, leading to low variance in iVector *w.r.t.* mean shifts. In scenarios like the ASpIRE LVCSR task where there is significant difference in the audio volume levels additional volume normalization is necessary. However as this normalization can significantly increase the latency of the system we explore volume perturbation of training data as a mechanism to increase volume variance in training data and possibly improve generalization.

Performing volume perturbation of the training data, where each recording in the training data was scaled with a random variable drawn from a uniform distribution over $[\frac{1}{64}, 8]$, emulates mean shifts in the MFCC domain. The volume perturbation was done on the artificially reverberated speech audio.

Table 6.7 compares the impact of volume normalization of test data and volume perturbation of training data on system performance. It can be seen that even with volume perturbation the acoustic model was not able to tackle the volume mismatch observed in ASpIRE test data. However volume perturbation led to a relative improvement of 13% when dealing with non-normalized test data. Further increasing the range of the uniform distribution used for sampling the volume scaling factors deteriorated the results. Volume normalization of the test data led to a relative improvement of 19.5% in WER when using the TDNN-B system trained on non-volume perturbed data. These results are promising.

Table 6.7: Comparison of systems w/ & w/o volume perturbed training data and w/ & w/o volume normalized test data

Acoustic Model	Training Data	Test Data	*dev* WER
TDNN B			38.3
TDNN B		vol. norm.	30.8
TDNN B	vp		33.3
TDNN B	vp	vol. norm.	30.9

vp :	volume perturbation of data after reverberation
vol. norm. :	volume normalization

6.2.3 Impact of enhanced lexicon and RNN-LM

6.2.3.1 Enhanced Lexicon

Table 6.8: Impact of pronunciation and silence probabilities

Model	*dev* WER
TDNN B*	32.1
TDNN B* + pronprob	31.6
TDNN B* + pronprob + silprob	30.8

* without pronunciation and silence probabilities.

Table 6.8[1] shows performance of using pronunciation and inter-word silence probabilities in the lexicon FST during decoding. As it's shown in the table, it is generally helpful to model pronunciation and silence probabilities in the lexicon FST. Further we observed that the impact of using this enhanced lexicon FST is significantly larger in this robust speech recognition task compared to other LVCSR tasks reported in [140].

Table 6.9: Impact of RNN-LM rescoring on TDNN B model

Model	*dev* WER
4gram LM Baseline	30.8
RNN-LM N-best top 100	30.2
RNN-LM N-best top 500	29.9
RNN-LM N-best top 1000	29.9
RNN-LM lattice max 4gram	29.9
RNN-LM lattice max 5gram	29.8
RNN-LM lattice max 6gram	29.8

6.2.3.2 RNN-LMs

Our RNN-LM rescoring results are shown in Table 6.9. We do the rescoring on both N-best lists and lattices. For N-best list rescoring, we tried to keep 100, 500 and 1000 best paths respectively, and we found it was enough to keep 500 best paths. For lattice rescoring, we used the RNN-LM to compute likelihood over a fixed context and it's shown in the table that keeping context of 5gram is sufficient in this particular case.

6.2.4 Sequence Training

Table 6.10: Results with sequence training of TDNN models

Acoustic Model	*dev* WER
TDNN A	31.7
TDNN A + sequence training[1]	34.0
TDNN A + sequence training[2]	30.6
TDNN B	30.8
TDNN B + sequence training[2]	29.5
TDNN B + sequence training[2,3]	29.1

[1] with sMBR criterion
[2] with modified sMBR criterion
[3] prior-adjustment

[1]All the other results shown in this section are with pronunciation and silence probabilities

Table 6.10 shows results of TDNNs using sequence training. The standard sMBR criterion was detrimental to the performance; but using the modified sMBR criterion described in Section 3.3.1.2, gains were observed on *dev* set. However these did not translate to *dev-test* set. With sequence training there was 4.2% relative improvement on dev set and 4.3% relative decrease on *dev-test* set. Further it can be seen that using priors computed from mean posteriors led to an improvement in the performance. TDNN-B with modified sequence training and modified prior computation is used in the next section.

6.2.5 Comparison across test-sets

From Table 6.11 it can be seen that sMBR training has mixed results. On careful analysis of the results on evaluation set it was observed that the sMBR system outperformed cross-entropy system for 70% of the 120 speakers. However the WER drastically increased for the other 30%. It was also observed that the sMBR system was prone to insertion errors, despite the use of the modified-sMBR objective function.

Table 6.11: Comparison across test-sets

Model	*dev*	*test*	*eval*
TDNN B*	30.8	27.7	44.3
TDNN B + RNN-LM	29.8	**26.5**	**43.4**
TDNN B + sMBR	29.1	28.9	43.9
TDNN B + sMBR + RNN-LM	**28.3**	28.2	**43.4**

* submitted to the evaluation

6.2.6 Comparison of acoustic models

The comparison of acoustic models on the ASpIRE task are shown in Table 6.12. It can be seen that TDNN-LSTM models are comparable to the LFR-BLSTM acoustic models. MFR-BLSTMs were not trained for this task due to the prohibitive training time. In addition to this comparison this particular table also tracks the progress in the ASpIRE LVCSR task since the publication of [98]. The changes, which include modification in the cost function [128], modification in the data reverberation recipe [87] and a modified acoustic model [181], result in a 27% relative improvement. TDNN-B corresponds to a rerun of the system which was our winning submission in the ASpIRE challenge.

Table 6.12: Comparison of TDNN, TDNN-LSTM and (B)LSTMs

Cost function	Acoustic Model	*dev* WER
Cross-entropy†	TDNN B‡‡	31.0
	LFR-BLSTM	29.4
LF-MMI†	TDNN D	27.8
	LFR-BLSTM	25.7
LF-MMI*	TDNN D	27.0
	LFR-BLSTM	24.6
	TDNN-LSTM	24.4‡

† Using data reverberation recipe with real RIRs and isotropic noise, described in Section 3.2.2.1

* Using data reverberation recipe with simulated RIRs and point source noises, described in Section 3.2.2.2

‡ Uses a modified training recipe without layerwise pre-training which can slightly improve the results.

‡‡ Best system submitted to the ASpIRE challenge.

Chapter 7

Conclusion and future work

In this thesis we proposed acoustic models which were able to model long span temporal contexts while ensuring that the latency of the model was restricted to 200 ms. The latency was reduced by enabling inference in frame-level increments. This is accomplished by using temporal convolution to model the future temporal context in recurrent neural networks. Further we also showed that hierarchical temporal convolution networks *i.e.,* time delay neural networks (TDNNs) can perform better than unidirectional LSTMs. We showed that the proposed temporal convolution based architectures are also computationally efficient as they use sub-sampling. Further as they do not use recurrent connections they can be trivially parallelized thus reducing the wall clock time of both training and inference.

The proposed architectures were used in acoustic models for far-field speech recognition. Thus demonstrating the capability of the proposed models in tackling long span interactions which abound in reverberant speech. Further due to the reduced computational complexity we showed that significantly larger amounts of training data can be used for training in a given amount of time. This can be especially helpful when trying to simulate various test scenarios during training, to reduce mismatch.

Finally we detailed acoustic modeling techniques for far-field speech recognition to reduce the performance gap between models trained with and without alignments from parallel clean data, and to effectively extract iVectors from noisy data.

7.0.1 Future work

In this thesis we restricted our attention to to HMM-DNN acoustic models, where a conditional independence assumption is made across neighboring output predictions. More recently sequence to sequence transducer neural networks [166, 182, 183], which do not make this assumption have been shown to outperform conventional acoustic models [184]. However even online variants of these models have significantly large latencies, compared to HMM-DNN models, due to the use of chunk based inference and "*attention*" mechanism. We would like to explore the applicability of the proposed latency reducing variants in this context.

Bibliography

[1] N. Morgan, J. Cohen, S. H. Krishnan, S. Chang, and S. Wegmann, "Final report: Ouch project (outing unfortunate characteristics of hmms)," 2013.

[2] D. Yu and L. Deng, *Automatic Speech Recognition*. Springer, 2012.

[3] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate bayesian a posteriori probabilities," *Neural computation*, vol. 3, no. 4, pp. 461–483, 1991.

[4] N. Morgan and H. Boaaurlard, "An introduction to hybrid HMM/connectionist continuous speech recognition," 1995.

[5] A.-r. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4273–4276.

[6] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, no. 3. IEEE, may 2013, pp. 6645–6649.

[7] K. Walker and S. Strassel, “The rats radio traffic collection system,” in *Proc. Odyssey*, 2012.

[8] J. L. Miller, F. Grosjean, and C. Lomanto, “Articulation Rate and Its Variability in Spontaneous Speech: A Reanalysis and Some Implications,” *Phonetica*, vol. 41, no. 4, pp. 215–225, jul 1984.

[9] J. Bilmes, “Maximum mutual information based reduction strategies for cross-correlation based joint distributional modeling,” in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, vol. 1. IEEE, 1998, pp. 469–472.

[10] H. H. Yang, S. V. Vuuren, S. Sharma, and H. Hermansky, “Relevance of time - frequency features for phonetic and speaker-channel classification,” *Speech Communication*, vol. 31, no. 1, pp. 35–50, may 2000.

[11] G. Hickok and D. Poeppel, “The cortical organization of speech processing,” *Nature Reviews Neuroscience*, vol. 8, no. 5, pp. 393–402, 2007.

[12] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, 1995.

[13] J. Andén and S. Mallat, “Deep scattering spectrum,” *Signal Processing, IEEE Transactions on*, vol. 62, no. 16, pp. 4114–4128, 2014.

[14] S. B. Davis and P. Mermelstein, “Comparison of parametric representations for

monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 4, pp. 357–366, 1980.

[15] X. Yang, K. Wang, and S. A. Shamma, "Auditory representations of acoustic signals," *Information Theory, IEEE Transactions on*, vol. 38, no. 2, pp. 824–839, 1992.

[16] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[17] M. Athineos and D. P. Ellis, "Frequency-domain linear prediction for temporal features," in *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*. IEEE, 2003, pp. 261–266.

[18] M. Athineos, H. Hermansky, and D. P. Ellis, "Plp2: Autoregressive modeling of auditory-like 2-d spectro-temporal patterns," in *Workshop on Statistical and Perceptual Audio Processing (SAPA)*, no. EPFL-CONF-83126, 2004.

[19] A. Hagen, "Robust speech recognition based on multi-stream processing," Tech. Rep., 2001.

[20] S. van Vuuren and H. Hermansky, "Data-driven design of rasta-like filters." in *Eurospeech*, 1997.

[21] M. Kleinschmidt, "Localized spectro-temporal features for automatic speech recognition." in *INTERSPEECH*. Citeseer, 2003.

[22] S. K. Nemala, K. Patil, and M. Elhilali, “A multistream feature framework based on bandpass modulation filtering for robust speech recognition,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, no. 2, pp. 416–426, 2013.

[23] S. Y. Zhao, S. V. Ravuri, and N. Morgan, “Multi-stream to many-stream: using spectro-temporal features for asr.” in *INTERSPEECH*, 2009, pp. 2951–2954.

[24] S. Greenberg and B. E. Kingsbury, “The modulation spectrogram: In pursuit of an invariant representation of speech,” in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 3. IEEE, 1997, pp. 1647–1650.

[25] S. Ganapathy, S. Thomas, and H. Hermansky, “Modulation frequency features for phoneme recognition in noisy speech,” *The Journal of the Acoustical Society of America*, vol. 125, no. 1, pp. EL8–EL12, 2009.

[26] H. Hermansky and N. Morgan, “Rasta processing of speech,” *Speech and Audio Processing, IEEE Transactions on*, vol. 2, no. 4, pp. 578–589, 1994.

[27] H. Hermansky, “The modulation spectrum in the automatic recognition of speech,” in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*. IEEE, 1997, pp. 140–147.

[28] H. Hermansky and S. Sharma, “Traps-classifiers of temporal patterns.” in *ICSLP*, 1998.

[29] B. Y. Chen, Q. Zhu, and N. Morgan, "Learning long-term temporal features in lvcsr using neural networks." in *INTERSPEECH*, 2004.

[30] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 4, pp. 788–798, 2011.

[31] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *ASRU*, 2013, pp. 55–59.

[32] M. Karafiát, L. Burget, P. Matějka, O. Glembek, and J. Černockỳ, "ivector-based discriminative adaptation for automatic speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 152–157.

[33] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1713–1725, 2014.

[34] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7398–7402.

[35] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural

network architectures for large scale acoustic modeling." in *Proceedings of Interspeech*, 2014, pp. 338–342.

[36] S. B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[37] I. Goodfellow, Y. Bengio, and A. Courville, "Sequence modeling: Recurrent and recursive nets," in *Deep Learning*. MIT Press, 2016, ch. 10, pp. 363–409.

[38] H. S., "Untersuchungen zu dynamischen neuronalen netzen," Ph.D. dissertation, T.U. Munchen, 1991.

[39] K. Doya, "Bifurcations of recurrent neural networks in gradient descent learning," *IEEE Transactions on neural networks*, vol. 1, pp. 75–80, 1993.

[40] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[41] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 1310–1318.

[42] A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks,” p. 124, 2008.

[43] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[45] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2342–2350.

[46] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. Ranzato, “Learning longer memory in recurrent neural networks,” *arXiv preprint arXiv:1412.7753*, 2014.

[47] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[48] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, “A clockwork rnn,” in *Proceedings of The 31st International Conference on Machine Learning*, 2014, pp. 1863–1871.

[49] Q. V. Le, N. Jaitly, and G. E. Hinton, "A simple way to initialize recurrent networks of rectified linear units," *arXiv preprint arXiv:1504.00941*, 2015.

[50] A. Zeyer, R. Schluter, and H. Ney, "Towards online-recognition with deep bidirectional LSTM acoustic models," *Proceedings of Interspeech*, vol. 08-12-Sept, pp. 3424–3428, 2016.

[51] Y. Zhang, G. Chen, D. Yu, K. Yaco, S. Khudanpur, and J. Glass, "Highway long short-term memory rnns for distant speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5755–5759.

[52] P. Doetsch, M. Kozielski, and H. Ney, "Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition," *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, vol. 2014-Decem, pp. 279–284, 2014.

[53] K. Chen, Z.-J. Yan, and Q. Huo, "Training Deep Bidirectional LSTM Acoustic Model for LVCSR by a Context-Sensitive-Chunk BPTT Approach," in *Proceedings of the Interspeech*, 2015.

[54] A.-r. Mohamed, F. Seide, D. Yu, J. Droppo, A. Stoicke, G. Zweig, and G. Penn, "Deep bi-directional recurrent networks over spectral windows," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 78–83.

[55] S. Xue and Z. Yan, “Improving latency-controlled BLSTM acoustic models for online speech recognition,” in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on.* IEEE, 2017.

[56] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, “Phoneme recognition using time-delay neural networks,” *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 3, pp. 328–339, 1989.

[57] K. Lang, A. Waibel, and G. E. Hinton, “A time-delay neural network for isolated word recognition,” *Neural Networks*, vol. 3, pp. 23–43, 1990.

[58] K. J. Lang and G. E. Hinton, “The Development of Time-Delay Neural Network Architecture for Speech Recognition,” Carnegie Mellon University, Tech. Rep., 1988.

[59] A. Waibel, “Modular construction of time-delay neural networks for speech recognition,” *Neural computation*, vol. 1, no. 1, pp. 39–46, 1989.

[60] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, “Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on.* IEEE, 2012, pp. 4277–4280.

[61] O. Abdel-Hamid, L. Deng, and D. Yu, “Exploring convolutional neural network structures and optimization techniques for speech recognition.” in *Interspeech*, 2013, pp. 3366–3370.

[62] P. Swietojanski, A. Ghoshal, and S. Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1120–1124, 2014.

[63] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[64] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, M. Bacchiani *et al.*, "Speaker location and microphone spacing invariant acoustic modeling from raw multichannel waveforms," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 30–36.

[65] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[66] T. Yoshioka, S. Karita, and T. Nakatani, "Far-field speech recognition using cnn-dnn-hmm with convolution in time," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4360–4364.

[67] J.-T. Huang, J. Li, and Y. Gong, "An analysis of convolutional neural networks for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4989–4993.

[68] W. Chan and I. Lane, "Deep convolutional neural networks for acoustic modeling in low resource languages," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2056–2060.

[69] L. Tóth, "Modeling long temporal contexts in convolutional neural network-based phone recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4575–4579.

[70] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Convolutional neural networks for acoustic modeling of raw time signal in lvcsr," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[71] D. Palaz, M. Magimai-Doss, and R. Collobert, "Analysis of cnn-based speech recognition system using raw speech as input," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[72] M. Bi, Y. Qian, and K. Yu, "Very deep convolutional neural networks for lvcsr," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[73] T. Zhao, Y. Zhao, and X. Chen, "Time-frequency kernel based cnn for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[74] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun, "Very deep multilingual con-

volutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4955–4959.

[75] V. Mitra and H. Franco, "Time-frequency convolutional networks for robust speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 317–323.

[76] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4845–4849.

[77] D. Yu, W. Xiong, J. Droppo, A. Stolcke, G. Ye, J. Li, and G. Zweig, "Deep convolutional neural networks with layer-wise context expansion and attention." in *INTERSPEECH*, 2016, pp. 17–21.

[78] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.

[79] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182.

[80] J. Li, A. Mohamed, G. Zweig, and Y. Gong, "Lstm time and frequency recurrence for automatic speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on.* IEEE, 2015, pp. 187–191.

[81] A. Graves, S. Fernández, and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in *Proceedings of ICANN*, 2007.

[82] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid long short-term memory," in *Proceedings of ICLR*, 2016.

[83] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio, "Renet: A recurrent neural network based alternative to convolutional networks," *arXiv preprint arXiv:1505.00393*, 2015.

[84] T. N. Sainath and B. Li, "Modeling time-frequency patterns with lstm vs. convolutional architectures for lvcsr tasks." in *INTERSPEECH*, 2016, pp. 813–817.

[85] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[86] R. Lippmann, E. Martin, and D. Paul, "Multi-style training for robust isolated-word speech recognition," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'87.*, vol. 12. IEEE, 1987, pp. 705–708.

[87] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data

augmentation of reverberant speech for robust speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5220–5224.

[88] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *Proceedings of ASRU*, 2011.

[89] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," *Proceedings of ICLR*, 2015.

[90] A. Senior and I. Lopez-Moreno, "Improving dnn speaker independence with i-vector inputs," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 225–229.

[91] M. J. Gales and P. C. Woodland, "Mean and variance adaptation within the mllr framework," *Computer Speech & Language*, vol. 10, no. 4, pp. 249–264, 1996.

[92] S. Garimella, A. Mandal, N. Strom, B. Hoffmeister, S. Matsoukas, and S. H. K. Parthasarathi, "Robust i-vector based adaptation of dnn acoustic model for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[93] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (vtlp) improves speech

recognition," in *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, 2013, pp. 625–660.

[94] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 554–557.

[95] N. Kanda, R. Takeda, and Y. Obuchi, "Elastic spectral distortion for low resource speech recognition with deep neural networks," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 309–314.

[96] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of Interspeech*, 2015.

[97] V. Peddinti, G. Chen, D. Povey, and S. Khudanpur, "Reverberation robust acoustic modeling using with time delay neural networks," in *Proceedings of Interspeech*.

[98] V. Peddinti, G. Chen, V. Manohar, T. Ko, D. Povey, and S. Khudanpur, "Jhu aspire system: Robust lvcsr with tdnns, ivector adaptation and rnn-lms," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 539–546.

[99] M. Harper, "The automatic speech recogition in reverberant environments (aspire)

challenge," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 547–554.

[100] R. Hsiao, J. Ma, W. Hartmann, M. Karafiát, F. Grézl, L. Burget, I. Szöke, J. H. Černockỳ, S. Watanabe, Z. Chen *et al.*, "Robust speech recognition in unknown reverberant and noisy conditions," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 533–538.

[101] "Google Home," https://madeby.google.com/home/, accessed: 2017-08-06.

[102] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home," in *Interspeech 2017*, 2017.

[103] H. Kuttruff, *Room acoustics*. Crc Press, 2016.

[104] S. Nakamura, K. Hiyane, F. Asano, T. Nishiura, and T. Yamada, "Acoustical sound database in real environments for sound scene understanding and hands-free speech recognition." in *LREC*, 2000.

[105] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, A. Sehr, W. Kellermann, and R. Maas, "The reverb challenge: Acommon evaluation framework for dereverberation and recognition of reverberant speech," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2013 IEEE Workshop on*. IEEE, 2013, pp. 1–4.

[106] M. Jeub, M. Schafer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Digital Signal Processing, 2009 16th International Conference on*. IEEE, 2009, pp. 1–5.

[107] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[108] L. Bahl, P. Brown, P. De Souza, and R. Mercer, "Maximum mutual information estimation of hidden markov model parameters for speech recognition," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*, vol. 11. IEEE, 1986, pp. 49–52.

[109] S. Kapadia, V. Valtchev, and S. Young, "Mmi training for continuous phoneme recognition on the timit database," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 491–494.

[110] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted mmi for model and feature-space discriminative training," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 4057–4060.

[111] D. Povey and P. C. Woodland, "Minimum phone error and i-smoothing for improved discriminative training," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1. IEEE, 2002, pp. I–105.

[112] V. Goel and W. J. Byrne, “Minimum bayes-risk automatic speech recognition,” *Computer Speech & Language*, vol. 14, no. 2, pp. 115–135, 2000.

[113] M. Gibson and T. Hain, “Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition.” in *Interspeech*, vol. 6, 2006, pp. 2406–2409.

[114] D. Povey and B. Kingsbury, “Evaluation of proposed modifications to mpe for large scale discriminative training,” in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4. IEEE, 2007, pp. IV–321.

[115] J. Bridle and L. Dodd, “An alphanet approach to optimising input transformations for continuous speech recognition,” in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. IEEE, 1991, pp. 277–280.

[116] B. Kingsbury, “Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling,” in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 3761–3764.

[117] B. Kingsbury, T. N. Sainath, and H. Soltau, “Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization,” in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[118] A.-r. Mohamed, D. Yu, and L. Deng, “Investigation of full-sequence training of

deep belief networks for speech recognition," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[119] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of context-dependent deep networks for conversational speech transcription," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6664–6668.

[120] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks." in *Interspeech*, 2013, pp. 2345–2349.

[121] V. Valtchev, J. Odell, P. C. Woodland, and S. J. Young, "Mmie training of large vocabulary recognition systems," *Speech Communication*, vol. 22, no. 4, pp. 303–314, 1997.

[122] D. Povey, "Discriminative training for large vocabulary speech recognition," Ph.D. dissertation, University of Cambridge, 2005.

[123] K. Chen, Z.-J. Yan, and Q. Huo, "Training deep bidirectional LSTM acoustic models for LVCSR by a context sensitive-chunk BPTT approach," in *Proc. Interspeech*, 2015.

[124] V. Manohar, D. Povey, and S. Khudanpur, "Semi-supervised maximum mutual information training of deep neural network acoustic models." in *INTERSPEECH*, 2015, pp. 2630–2634.

[125] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[126] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[127] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4280–4284.

[128] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi." in *INTERSPEECH*, 2016, pp. 2751–2755.

[129] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[130] H. Sak, F. de Chaumont Quitry, T. Sainath, K. Rao *et al.*, "Acoustic modelling with cd-ctc-smbr lstm rnns," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 604–609.

[131] D. Povey, *Nnet3: Neural network toolkit for generic acyclic computation graphs*, 2017 (accessed March 23 , 2017), http://www.danielpovey.com/kaldi-docs/dnn3_code.html.

[132] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, “1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[133] N. Strom, “Scalable distributed dnn training using commodity gpu cloud computing,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[134] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, “Large scale distributed deep networks,” in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[135] A. Stolcke *et al.*, “Srilm-an extensible language modeling toolkit.” in *Interspeech*, vol. 2002, 2002, p. 2002.

[136] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, “Rnnlm-recurrent neural network language modeling toolkit,” in *Proc. of the 2011 ASRU Workshop*, 2011, pp. 196–201.

[137] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, “Strategies for training

large scale neural network language models," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 196–201.

[138] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[139] X. Liu, Y. Wang, X. Chen, M. J. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4908–4912.

[140] G. Chen, H. Xu, M. Wu, D. Povey, and S. Khudanpur, "Pronunciation and silence probability modeling for asr." in *Interspeech*, 2015, pp. 533–537.

[141] T. Hain, "Implicit modelling of pronunciation variation in automatic speech recognition," *Speech communication*, vol. 46, no. 2, pp. 171–188, 2005.

[142] B. Peskin, M. Newman, D. McAllaster, V. Nagesha, H. Richards, S. Wegmann, M. Hunt, and L. Gillick, "Improvements in recognition of conversational telephone speech," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 53–56.

[143] T. Hain, P. Woodland, G. Evermann, and D. Povey, "The cu-htk march 2000 hub5e

transcription system," in *Proc. Speech Transcription Workshop*, vol. 1. Baltimore, 2000.

[144] E. Fosler, M. Weintraub, S. Wegmann, Y.-H. Kao, S. Khudanpur, C. Galles, and M. Saraclar, "Automatic learning of word pronunciation from data," in *the Proceedings of the International Conference on Spoken Language Processing*, 1996.

[145] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[146] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*. Springer, 1990, pp. 286–297.

[147] P. Dutilleux, "An implementation of the algorithme à trous to compute the wavelet transform," in *Wavelets*. Springer, 1990, pp. 298–304.

[148] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, pp. 125–125.

[149] K. Veselỳ, M. Karafiát, and F. Grézl, "Convolutive bottleneck network features for lvcsr," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 42–47.

[150] L. Tóth, “Convolutional deep rectifier neural nets for phone recognition.” in *Interspeech*, 2013, pp. 1722–1726.

[151] F. Grézl, M. Karafiát, and K. Vesely, “Adaptation of multilingual stacked bottleneck neural network structure for new language,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 7654–7658.

[152] F. Seide, G. Li, X. Chen, and D. Yu, “Feature engineering in context-dependent deep neural networks for conversational speech transcription,” in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 24–29.

[153] G. Saon, H. Soltau, A. Emami, and M. Picheny, “Unfolded recurrent neural networks for speech recognition,” in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[154] H. Soltau, G. Saon, and T. Sainath, “Joint training of convolutional and non-convolutional neural networks,” in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 5572–5576.

[155] V. Peddinti, D. Povey, and S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in *Proceedings of Interspeech*, 2015.

[156] P. Price, W. Fisher, J. Bernstein, and D. Pallett, "The darpa 1000-word resource management database for continuous speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr 1988, pp. 651–654 vol.1.

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

[158] A. Rousseau, P. Deléglise, and Y. Estève, "Ted-lium: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

[159] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: telephone speech corpus for research and development," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Mar 1992, pp. 517–520 vol.1.

[160] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Proceedings of ICASSP*. ieee, 2015.

[161] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: a resource for the next generations of speech-to-text." in *LREC*, vol. 4, 2004, pp. 69–71.

[162] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and

S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Proceedings of Interspeech*, 2016, pp. 2751–2755.

[163] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models," in *Proceedings of Interspeech 2016*, 2016, pp. 22–26.

[164] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[165] A. Graves, "Hierarchical subsampling networks," *Supervised Sequence Labelling with Recurrent Neural Networks*, pp. 109–131, 2012.

[166] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proceedings of ICASSP*. IEEE, 2016, pp. 4960–4964.

[167] L. Lu, L. Kong, C. Dyer, N. A. Smith, and S. Renals, "Segmental recurrent neural networks for end-to-end speech recognition," in *Proceedings of Interspeech*, 2016.

[168] *Comparison of TDNN-LSTMs and CNN-TDNN-LSTMs.*, 2017 (accessed June 15 , 2017), https://github.com/kaldi-asr/kaldi/pull/1685.

[169] G. Cheng, V. Peddinti, D. Povey, V. Manohar, S. Khudanpur, and Y. Yan, "An exploration of dropout with lstms," in *Proceedings of Interspeech*, 2017.

[170] "Amazon Echo," https://en.wikipedia.org/wiki/Amazon_Echo, accessed: 2017-08-06.

[171] "Apple Homepod," https://www.apple.com/homepod/, accessed: 2017-08-06.

[172] T. Yoshioka, A. Sehr, M. Delcroix, K. Kinoshita, R. Maas, T. Nakatani, and W. Kellermann, "Making machines understand us in reverberant rooms: Robustness against reverberation for automatic speech recognition," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 114–126, Nov 2012.

[173] T. Hain, L. Burget, J. Dines, G. Garau, M. Karafiat, M. Lincoln, J. Vepa, and V. Wan, "The ami meeting transcription system: Progress and performance," in *Machine learning for multimodal interaction*. Springer, 2006, pp. 419–431.

[174] I. McCowan, J. Carletta, W. Kraaij, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos *et al.*, "The AMI meeting corpus," in *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, vol. 88, 2005.

[175] P. Swietojanski, A. Ghoshal, and S. Renals, "Hybrid acoustic models for distant and multichannel large vocabulary speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 285–290.

[176] V. Peddinti, V. Manohar, Y. Wang, D. Povey, and S. Khudanpur, “Far-field asr without parallel data,” in *Proceedings of Interspeech*, 2016.

[177] *Data cleanup strategy used in Kaldi.*, 2017 (accessed August 17, 2017), https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/local/run_cleanup_segmentation.sh.

[178] H. Xu, D. Povey, L. Mangu, and J. Zhu, “Minimum bayes risk decoding and system combination based on a recursion for edit distance,” *Computer Speech & Language*, vol. 25, no. 4, pp. 802–828, 2011.

[179] X. Anguera, C. Wooters, and J. Hernando, “Acoustic beamforming for speaker diarization of meetings,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 7, pp. 2011–2022, 2007.

[180] M. Huijbregts and F. de Jong, “Robust speech/non-speech classification in heterogeneous multimedia content,” *Speech Communication*, vol. 53, no. 2, pp. 143 – 153, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639310001421

[181] V. Peddinti, Y. Wang, D. Povey, and S. Khudanpur, “Low latency acoustic modeling using temporal convolution and lstms,” *IEEE Signal Processing Letters*, 2017.

[182] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012.

[183] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: first results," *arXiv preprint arXiv:1412.1602*, 2014.

[184] R. Prabhavalkar, K. Rao, T. N. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," *Proc. Interspeech 2017*, pp. 939–943, 2017.

Vita

Vijayaditya Peddinti received a Bachelor of Technology in Information and Communication Technology from Dhirubhai Ambani Institute of Information and Communication Technology in 2007, Masters of Computer Science by Research from International Institute for Information Technology in 2011. He enrolled in the Electrical and Computer Engineering Ph.D in August 2011 and has been a member of the Center for Language and Speech Processing since then. His research focuses on acoustic modeling for automatic speech recognition. During his graduate study at Johns Hopkins University he had the oppurtunity to work as intern in speech processing teams at IBM Research and Microsoft Research. He received a best paper award in Interspeech-2015 conference and was part of the JHU team which was one of the three winners of the IARPA ASpIRE challenge of far-field speech recognition. He is also a developer of the Kaldi speech recognition toolkit.

He has been working as a research scientist in the acoustic modeling team of Google Inc. since December 2017.