LEARNING FEATURE REPRESENTATION FOR AUTOMATIC SPEECH RECOGNITION

by

Pegah Ghahremani

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

Baltimore, Maryland

October, 2019

Abstract

Feature extraction in automatic speech recognition (ASR) can be regarded as learning representations from lower-level to more abstract higher-level features. Lower-level feature can be viewed as features from the signal domain, such as perceptual linear predictive (PLP) and Mel-frequency cepstral coefficients (MFCCs) features. Higher-level feature representations can be considered as bottleneck features (BNFs) learned using deep neural networks (DNNs). In this thesis, we focus on improving feature extraction at different levels mainly for ASR.

The first part of this thesis focuses on learning features from the signal domain that help ASR. Hand-crafted spectral and cepstral features such as MFCC are the main features used in most conventional ASR systems; all are inspired by physiological models of the human auditory system. However, some aspects of the signal such as pitch cannot be easily extracted from spectral features, but are found to be useful for ASR. We explore new algorithm to extract a pitch feature directly from a signal for ASR and show that this feature,

appended to the other feature, gives consistent improvements in various languages, especially tonal languages.

We then investigate replacing the conventional features with jointly training from the signal domain using time domain, and frequency domain approaches. The results show that our time-domain joint feature learning setup achieves state-of-the-art performance using MFCC, while our frequency domain setup outperforms them in various datasets.

Joint feature extraction results in learning data or language-dependent filter banks, that can degrade the performance in unseen noise and channel conditions or other languages. To tackle this, we investigate joint universal feature learning across different languages using the proposed direct-from-signal setups. We then investigate the filter banks learned in this setup and propose a new set of features as an extension to conventional Mel filter banks. The results show consistent word error rate (WER) improvement, especially in clean condition.

The second part of this thesis focuses on learning higher-level feature embedding. We investigate learning and transferring deep feature representations across different domains using multi-task learning and weight transfer approaches. They have been adopted to explicitly learn intermediate-level features that are useful for several different tasks.

Primary Readers and Advisors: Daniel Povey, Sanjeev Khudanpur

ABSTRACT

Secondary Readers: Hynek Hermansky, Najim Dehak

Acknowledgments

I would like to thank my advisors Daniel Povey and Sanjeev Khudanpur for giving me the valuable opportunity to investigate the topics of my interests and their great guidance and support during my graduate life.

Also, I want to thank my committee members - Hynek Hermansky and Najim Dehak for their company which helped me keep motivated and for their valuable feedback. I am grateful for the suggestions from Shinji Watanabe, which resulted in improving the thesis.

I would like to thank Vimal Manohar, Hossein Hadian, Vijay Peddinti, David Snyder, Xiahui Zhang, Matthew Wiesner, Chunxi Liu, Hainan Xu, Harish Mallidi, Phani Sankar and Raghavendra Pappagari. Many thanks to Jan Trmal for his help with Kaldi issues.

Thanks to Jasha Droppo, Mike Seltzer and Geoffry Zweig, Microsoft Research, for their fantastic assistance during my internship. They helped me to expand my research view to a great extent.

Finally, I would especially like to thank my amazing family for their love,

support, and constant encouragement over the years. In particular, I would like to thank my amazing husband, Hossein, for his unconditional love and support during all these years. Without him, it would have been impossible for me to pursue my Ph.D. degree. Thank you to my wonderful parents, Zinat and Ali, for being the best parents and role models. I undoubtedly could not have done this without you. Last but not least, I want to thank my parents-in-law, Ata and Farideh, my wonderful sister, Parmida, and my sisters-in-law, Maryam and Fatemeh, for all of their support and encouragement.

Dedication

This thesis is dedicated to my beloved parents, sister and my wonderful husband.

Contents

List of Tables

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

The emergence of deep neural networks (DNNs) as acoustic models results in considerable advancement in automatic speech recognition (ASR) in recent years. DNNs are able to extract discriminative feature representations that are robust to distortions and variability in speech signals [**?**]. DNNs are applicable in (a) learning new feature representations from the signal domain, (b) generating phone posteriors or internal representations (e.g., BNF), and (c) building another model with a discriminative front-end [**?**]. Convolution neural networks (CNNs) [**?**, **?**, **?**] and recurrent neural networks (RNNs) [**?**, **?**, **?**] have also been exploited as acoustic models.

1.1 Problem

ASR maps a speech signal to the corresponding sequence of words. To perform this, a series of acoustic features are extracted from the speech signal. Most speech recognition systems use a frame-based model in which an input waveform is converted into a sequence of frames of features with equal dimensions. The goal of feature extraction in ASR is to represent a window of speech samples with a feature vector that represents the underlying phonetic content of the speech. Most conventional speech recognition systems have primarily focused on using traditional handcrafted features such as log Mel filter banks (FBANKS). These features are low dimensional representations of the speech signal, and they preserve the information required to achieve high ASR performance. The Mel features are derived by element-wise multiplication of the magnitude spectrum with positive Mel filter weights followed by L_2 pooling. Gammatone features are also computed by convolution of the time-domain signal with gammatone filters [**?**] followed by average pooling. These features are inspired by physiological models of the human auditory system, and may not be the most appropriate features for the final ASR objective of word error rate (WER) reduction. Recent advances in deep learning have led to performance improvement in ASR. With the increase in computational power and the availability of large speech corpora, it should be possible to learn features

automatically from speech databases. DNN model can also take input features with large dimensions and combine information from different sources. Feature extraction in ASR can also be regarded as learning more abstract feature embedding using DNNs.

In this thesis, we aim to answer two main questions. First, can we improve ASR performance by defining new features, which are complementary to conventional features, and also can we get any improvement by joint feature extraction and classification training from the signal domain using DNNs compared to conventional features? Second, what is the best approach to learn deep feature representations using DNN and transfer them across different languages and datasets with various mismatched conditions in ASR?

1.2 Thesis outline

A brief outline of this thesis is as follows:

Hand-designed feature extraction

In the first part of this thesis (Chapter 2), we start exploring new algorithms to extract hand-designed features directly from a signal, that are complementary to spectral features for use in ASR. Pitch is a time-domain aspect of the signal that cannot be easily extracted from spectral features like MFCC.

Pitch and probability of voicing

Tone plays a lexical role in determining the meaning of words in tonal languages such as spoken Mandarin [**?**]. In nontonal languages, intonation may be used for higher-level meanings associated with emotion and do not change the base meaning of words. Many research efforts have been conducted to incorporate tone information in ASR, especially for tonal languages. In Section 2.1, we present a new algorithm that produces pitch and probability-of-voicing estimates for use as features in ASR systems. The results show considerable WER improvements compared to conventional pitch extraction algorithms [**?**].

Joint feature extraction and classification training

While chapter 2 explores extracting hand-designed features from the signal, in the next part of this thesis (Chapter 3), we use deep learning approaches to do feature extraction and classification jointly from the speech signal. Data-driven feature extraction techniques, which jointly train feature extraction and classification, are expanded considerably with the development of deep learning algorithms [**?**, **?**, **?**, **?**, **?**]. In direct-from-signal models, the first layers of the network are designed to learn filter banks directly from the raw waveform. Most of the prior research using raw waveform systems are still behind the

conventional Mel space systems especially in small to medium training data conditions ($10 - 300$ hours of training data).

Time-domain joint feature extraction

The direct-from-signal setup described in Section 3.1 contains the convolution in the first layer of the network followed by pooling and compression. The convolution layer operates on a raw speech signal with long filters to mimic bandpass filters. Most feature extraction techniques use static compression methods such as 10^{th}-root and log to reduce the feature's dynamic range. The log compression is used on the filter outputs to reduce the dynamic range of features in this setup. Next, the filter outputs are aggregated over a portion of the time axis using our proposed Network-in-Network (NiN) pooling structure (described in Subsection 3.1.4).

Many variations for a given phoneme in the form of phase shifts and temporal distortions can be detrimental to learn directly from the raw waveform. Hand-designed features like MFCC and PLP and the ones proposed in Chapter 2 are more invariant to these variations. To handle this issue, convolution over time is done and the information aggregation over the convolution filter outputs using pooling are used in the model. The success of the direct-from-signal models highly depends on the choice of convolution and pooling methods [**?**, **?**, **?**, **?**]. We explore various pooling techniques i.e., no pooling, max [**?**],

p-norm [**?**], average pooling. We propose a new type of pooling method, NiN pooling, that is a special case of the network-in-network architecture, with repeated blocks interleaved between layers of rectified linear units (ReLU). We achieve state-of-the-art performance using the proposed structure as a pooling layer [**?**].

Frequency-domain joint feature extraction

The time-frequency duality suggests extending direct-from-signal feature extraction techniques and jointly learns the filter banks on the real and imaginary part of the Fourier transform of the input signal [**?**] (complex linear component, i.e., CLP). The setup described above (Section 3.1) performs filtering and pooling in the time-domain. Filtering and pooling in the frequency domain can help to avoid the complexity of training convolution layer and parameter tuning. Since the phase information is less important for single microphone ASR, the filter training and pooling can be done on the input signal energy to reduce complexity in the CLP model. In Section 3.2, we simplify our previous approach in Section 3.1 (i.e., time-domain feature learning) by operating in the frequency domain. We include a Fourier transform layer in the network and let the network learn the filter banks in the frequency domain.

Frequency-domain feature learning has been previously used in [**?**] and [**?**], however, we propose a new normalization layer which helps with better train-

ing stability and better convergence of the filters. Additionally, we employ a different weight constraint approach which further improves the results. We use the proposed frequency-domain layer in state-of-the-art ASR setup and show significant WER improvements on various well-known large vocabulary databases.

Joint feature extraction application in emotion identification task

Humans express emotional state-related information through numerous subtle ways including low-level acoustic descriptors like pitch, voicing probability, energy, zero-crossing rate, Mel filter bank features, formant locations, bandwidths, harmonics-to-noise ratio and jitter. Some of these features may or may not be directly represented by common features. Joint feature extraction in this setup attempts to learn a specific set of filters, jointly optimized to minimize emotion identification objectives.

Joint feature extraction shows some improvements in ASR, and the goal in Chapter 4 is to see improvement from the direct-from-signal setup in another speech task. We investigate the effect of the proposed setups of Chapter 3 in a speech-based emotion identification (Section 4). The results in emotion identification shows improvements over 257-dimensional magnitude FFT vectors

based on the DNN setup reported in [**?**].

Universal feature extraction

In Chapter 5, we extend exploring joint feature extraction by learning multilingual features by sharing knowledge across different languages. We investigate knowledge transfer across languages via learning a language-universal feature extractor that is trained over a group of languages. In this setup, a separate output layer is used for each language, while all other hidden layers jointly model the variability of all the source languages.

Multilingual bottleneck feature extraction

In Section 5.1, we use conventional MFCC features to train a universal multilingual model and use BNFs that are extracted from this model as an additional language-independent feature vector to improve the ASR performance for a target in-domain language [**?**].

Universal filter bank learning

The primary challenge in joint feature extraction setup is to learn language or data-independent filter banks, which generalizes to other languages and datasets. In Section 5.2, we explore the proposed direct-from-signal model

on multi-English and multi-language datasets. One of the main objectives in this section is to use the best direct-from-raw-waveform setup in the universal acoustic model setup to train universal bandpass filters. This model uses a universal phone set ASR system, and it allows us to leverage existing resources in other languages. We also investigate the setup in the multi-English dataset to learn a set of filter banks, which generalizes to different noise and channel conditions, by pooling multiple English corpora.

Data-driven feature learning

The main topic of Chapter 6 is a new set of data-driven filters. Based on the learned filters in frequency-domain layers in Section 3.2, we propose a new set of approximated filters, that enable faster training of the acoustic models while delivering the same improvement as the proposed joint feature extraction setup. We propose new warping functions to approximate the center frequencies based on the learned filters. Also, we investigate new methods to approximate bandwidths for new filters.

Deep feature representation across domains

As discussed in Section 5.1, multi-task learning and weight transfer across different languages, as two transfer learning approaches, help to learn better

feature representations and improve ASR performance. In Chapter 7, we investigate these 3 transfer learning approaches to transfer knowledge between models in different domains. These approaches have been adopted to explicitly learn intermediate-level features in the neural network that are useful for several different tasks. The intermediate representation in neural networks trained on speech data appears not to be specific to any particular task, while the higher layers are task-specific. We also investigate sequence-trained teacher-student framework [**?**] as a transfer learning approach in sampling-rate mismatched scenario and compare its performance with weight transfer.

1.3 Contributions

We make the following contributions in this dissertation through different chapters:

Chapter 2

- Proposed a new pitch extraction algorithm that produces pitch and probability-of-voicing estimates for use as features in ASR [**?**].
- These features show significant performance improvement in tonal languages and substantial improvements for non-tonal languages and it is now widely used in different applications.

Chapter 3

- Proposed time-domain joint feature learning setup that outperforms current joint feature learning setups and achieves state-of-the-art performance on ASR application [**?**]. This setup also achieves best performance in the speech-based emotion identification [**?**].

- Proposed frequency-domain joint feature learning setup that is faster to train and shows WER improvements on various large vocabulary databases [**?**].

Chapter 6

- Investigated learned filters in the frequency-domain joint feature extraction setup and proposed a new set of approximated filters that deliver the improvement gained from the proposed joint feature extraction setup.

- Proposed modified Mel filter bank features based on the learned filter banks, that shows some improvement on various clean and noisy databases.

Chapter 7

- Investigated different transfer learning methods in ASR in more detail [**?**, **?**] and proposed a weight transfer setup for use in ASR that shows con-

siderable improvement on different datasets.

Chapter 2

Hand-designed feature extraction

2.1 Pitch and probability of voicing

In this section, we present an algorithm that produces pitch and probability of voicing (POV) estimates for use as features in ASR systems. These features show significant performance improvements in tonal languages ASR systems and even substantial improvements for non-tonal languages. Our method, which we are calling the Kaldi pitch tracker (implemented in the Kaldi ASR toolkit), is a highly modified version of the getf0 (RAPT) algorithm [**?**]. Unlike the original getf0, we do not make a hard decision whether any given frame is voiced or unvoiced; instead, we assign a pitch even to unvoiced frames while

constraining the pitch trajectory to be continuous. Our algorithm also produces a quantity that can be used as a POV measure; it is based on the normalized autocorrelation measure that is used by our pitch extractor. We present results on data from various languages in the BABEL project [**?**] and show a significant improvement over systems without tonal features and systems where pitch and POV information is obtained from SAcC, another existing pitch tracker [**?**], or getf0.

Our goal in this section is to obtain well-performing pitch and POV features for use in speech recognition, and accurately produce a standardized pitch feature for use in the Kaldi ASR toolkit [**?**]. In Subsection 2.1.1, we review different pitch trackers to select the best previously published pitch extraction algorithms; in Subsection 2.1.2, we describe our proposed method. We detail, in Subsection 2.1.3, the pitch post-processing methods for the baseline pitch and POV features, and our proposed pitch and POV features. We describe our ASR system and datasets and show experimental results in Subsection 2.1.7.

2.1.1 Existing pitch extraction methods

We started our work by obtaining various off-the-shelf pitch extractors, namely Yin [**?**], getf0 [**?**], SAcC [**?**], Wu [**?**], SWIPE [**?**] and YAAPT [**?**]. We compared their accuracy as pitch trackers (see Subsection 2.1.7). For this, we used the Keele database [**?**], which consists of about half an hour of speech manually

labeled for pitch and voicing. We selected three of the best-performing methods for further study; these were SAcC, Yin, and getf0 (we did not consider YAAPT at this point because of its greater complexity; it is based on getf0).

Next, as will be seen in the experimental section, we compared the pitch features of SAcC, Yin, and getf0 in an ASR task. For this comparison, we processed the pitch as described in Subsection 2.1.3 and used the voicing feature from SAcC. Other feature extractors don't generate POV feature. These experiments did not show substantial differences between the various pitch extractors, so we used getf0 as our starting point as it seemed to perform slightly better than Yin, and it is a relatively simple algorithm to implement (SAcC showed better performance, but it is a fairly complex method).

2.1.2 The Kaldi pitch extractor

Our algorithm is a highly modified version of the getf0 algorithm. It is based on finding lag values that maximize the normalized cross correlation function (NCCF). Like most pitch extraction algorithms, our algorithm has some parameters that are set by hand and are shown in Table 2.1. It should not be necessary to change any of these values when applying them to other datasets with different sampling rates.

Probably, the most significant change from getf0 is that rather than making hard decisions about voicing on each frame, we treat all frames as voiced and

Parameter	Value	Explanation
min-f0	50	Minimum possible frequency value (Hz)
max-f0	400	Maximum possible frequency value (Hz)
window-width	0.025	Length in seconds of window used for NCCF
window-shift	0.01	Frame-shift, in seconds (should match that used for baseline features e.g., PLP)
soft-min-f0	10	Minimum f0, applied in soft way; must not exceed min-f0.
nccf-ballast	0.625	Increasing this factor reduces NCCF for quiet frames, helping ensure pitch continuity in unvoiced regions
penalty-factor	0.1	Factor that penalizes frequency change
delta-pitch	0.005	Smallest relative change in pitch that our algorithm measures
lowpass-cutoff	1000	Low-pass cutoff that we apply to the raw signal
lowpass-filter-width	2	Integer that determines filter width of low-pass filter (more gives wider filter with sharper cutoff)
resample-frequency	4000	Sample frequency for NCCF; must exceed twice lowpass-cutoff.
upsample-filter-width	5	Integer that determines filter width when upsampling NCCF

Table 2.1: Parameters of our algorithm, and their default values

allow the Viterbi search to interpolate across unvoiced regions naturally. To make this happen, we had to make a few changes. We do not limit the search to the relative local maxima of the NCCF– we allow it to take any value on a reasonably fine grid. Also, we alter the penalty on the change Δ in log-pitch from proportional to $|\Delta|$ to Δ^2, which causes the algorithm to interpolate across constant regions of the NCCF linearly. In addition, we add a "ballast" term to the NCCF formula which makes it approach zero for "quiet" areas of the signal; for this to work, we have to energy-normalize the signal globally. This requires lookahead, as does the Viterbi search. We also created a modified version of the algorithm for online use which is available in Kaldi.

We low-pass the signal to 1kHz which improves accuracy as well as making the algorithm more efficient by allowing us to work with a sub-sampled signal. Moreover, we obtain a feature based on the values of the NCCF, not just the lag at which it is maximized, not just the lag at which it is maximized, which is related to the POV and helps in ASR.

Resampling method

For completeness, we will specify the method we use to resample signals. Let the sampled source signal be viewed as a continuous function of time $s(t)$, where the n'th sample x_n becomes a Dirac delta function shifted to time n/S where S is the sampling rate, and scaled by x_n/S. We define a filter function $f_{C,w}(t)$, parameterized by a cutoff frequency $C \leq S/2$, and an integer width factor $w \geq 1$. Let the window function $w(t)$ be a raised-cosine (Hanning) window with support on $\left[\frac{-w}{2C}, \frac{w}{2C}\right]$. Then define

$$f_{C,w}(t) = 2C \operatorname{sinc}(2Ct) w(t) \tag{2.1}$$

where sinc is the normalized sinc function. To take a sample of the signal at an arbitrary time t, we simply evaluate $\int_u s(u) f(t-u)$ which is the sum $s'(t) = \sum_n x_n \frac{f_{C,w}(t-n/S)}{S}$. Naturally, we only evaluate this for the values of n for which the summand is nonzero.

Subsampling and normalization

Let the input to the algorithm be a discretely sampled signal, sampled with sampling frequency S. The first stage is to use the resampling method above, with the filter parameterized by lowpass-cutoff and lowpass-filter-width, to resample the signal at a sampling frequency resample-frequency. Next, we normalize the resampled signal's dynamic range by dividing by the root-mean-square signal value (if it is nonzero). Let the result be the signal x_n, with $n = 0, 1, \ldots N-1$. We then apply pre-emphasis, setting $y_n = x_n - \text{preemph-coeff}\, x_{n-1}$.

Computing the NCCF

First, we need to establish the range of lags over which to compute the NCCF. These depend on the frequency range we search over. Define the quantities min-lag $= 1/$max-f0, max-lag $= 1/$min-f0, which are the minimum and maximum lags (in seconds) at which we need the NCCF, and furthermore define upsample-filter-frequency as resample-frequency$/2$ which is the filter cutoff we will use when upsampling the NCCF. Then with filter-width w (in seconds) defined as upsample-filter-width/upsample-filter-frequency.

Let outer-min-lag $=$ min-lag $- w/2$ and outer-max-lag $=$ max-lag $+ w/2$, which gives us a slightly larger range of lags over which to compute the NCCF (we need to extend the range by half the width of the filter function we'll use when up-sampling the NCCF).

Consider the frame-index $t = 0, 1, \ldots$. The time span of the signal that we need to process starts at the closest sample to the time $t \cdot$ window-shift and is of the length window-width + outer-max-lag (in seconds). We produce output for all frame-indices t such that this time span is wholly within the time span of the input file. Let $\mathbf{w}_t = (w_{t,0}, w_{t,1}, \ldots)$ be the sequence of samples used for frame t; this is a subsequence of the sequence x_n, of the length $\lceil (\text{window-width} + \text{outer-max-lag}) \cdot \text{resample-frequency} \rceil$ samples, but with its mean subtracted away. Let $\mathbf{v}_{t,i}$ represent the sub-sequence of $\mathbf{w}_t$ starting at position i and of length $n = \lceil \text{window-width} \cdot \text{resample-frequency} \rceil$, so for instance $\mathbf{v}_{t,3} = (w_{t,3}, \ldots, w_{t,n+3})$. Where convenient, we will view these sequences as vectors. The NCCF for frame t and lag-index l is

$$\phi_{t,l} = \frac{\mathbf{v}_{t,0}^T \mathbf{v}_{t,l}}{\sqrt{||\mathbf{v}_{t,0}||_2^2 ||\mathbf{v}_{t,l}||_2^2 + n^4 \, \text{nccf-ballast}}}, \tag{2.2}$$

where $||\mathbf{x}||_2^2 = \mathbf{x}^T\mathbf{x}$. We compute this for all l s.t. outer-min-lag $\leq l/\text{resample-frequency} \leq$ outer-max-lag.

Upsampling the NCCF

Next, we upsample the NCCF in a non-linear way: that is, we measure the NCCF at the geometrically increasing sequence of lag values

$$L_i = \text{min-lag}\,(1 + \text{delta-pitch})^i\,,\ i \geq 0,\ L_i \leq \text{max-lag}, \tag{2.3}$$

where the condition $L_i \leq$ max-lag determines the maximum index i. For each index i, we compute the NCCF $\Phi_{t,i}$ which is the NCCF ϕ_{t,L_i} measured on frame t at lag L_i, using the resampling method described in Subsection 2.1.2 parameterized by upsample-filter-frequency and upsample-filter-width.

Defining the cost function

Suppose the range of the frame-index t is $0 \leq t < T$ and the range of the lag index i is $0 \leq i < I$ (we will mention later how these ranges are determined). The pitch trajectory is obtained by minimizing a cost function defined on a sequence of indices $\mathbf{s} = (s_t)_{t=0}^{T-1}$; each element s_t is interpreted as a lag-index i, so $0 \leq s_t < I$. The cost function consists of a local cost for each time t plus a term that penalizes changes in frequency:

$$C(\mathbf{s}) = \sum_{t=0}^{T-1} \text{local-cost}(t, s_t) + \sum_{t=1}^{T-1} \text{penalty-factor}(\log(L_{s_t}/L_{s_{t-1}}))^2, \tag{2.4}$$

where the configuration value penalty-factor controls how strongly we penalize changes in frequency, and we define

$$\text{local-cost}(t, i) = 1 - \Phi_{t,i}(1 - \text{soft-min-f0}\, L_i) \tag{2.5}$$

View $1 - \Phi_{t,i}$ as the basic local cost, and the multiplicative factor on $\Phi_{t,i}$ as a kind of penalization of high lags, which will tend to keep the selected lags substantially below $1/$soft-min-f0.

Optimizing the cost function

The algorithm we use to compute the sequence s that minimizes $C(\cdot)$ is based on the Viterbi algorithm. A naive implementation would take quadratic time in the number I of lag-indices. Let the Viterbi back-trace on time $t > 0$ and lag i be $b(t, i)$; this evaluates an integer index (like i) that is the optimal lag-index on time $t-1$ that we "point back to" from (t, i). Due to convexity w.r.t i in the transition-cost, we can show that $b(t, i + 1) \geq b(t, i)$. We can use this to obtain an exact search algorithm that takes time closer in practice to linear in I (although not provably so; it is data-dependent). Let the forward-cost be written as $c(t, i)$. Ignoring all end effects for purposes of exposition, the basic idea is that, on time t, we first do a "forward pass" for $i = 0$ to $I-1$, and set $c(t, i)$ and $b(t, i)$ while only considering the previous forward-costs $c(t-1, j)$ for $j = b(t, i-1)$ to i. Then, in a "backward pass" for $i = I-1$ to 0, we see whether

we can get a better forward-cost and corresponding backtrace than we already have by considering the previous forward-costs $c(t-1, j)$ for $j = i+1$ to $b(t, i+1)$. Let the result of this computation be the state-sequence $\mathrm{s} = (s_t)_{t=0}^{T-1}$.

Results

The output of this algorithm is the pitch and the NCCF values for each frame. The pitch for frame t equals $1/L_{s_t}$, with lags L_i as defined in Equation (2.3). The NCCF values are computed at the selected lags, so on frame t we output Φ_{t,s_t} (see Subsection 2.1.2); however, for purposes of this output we compute the NCCF without the nccf-ballast term in Equation (2.2) (and treating $0/0$ as zero in case of a zero sequence in the signal). This means that we need to do the upsampling computation of the NCCF twice. Next, we describe how we post-process the output for use as features for ASR.

2.1.3 Pitch post-processing methods

Baseline pitch post-processing method

The post-processing that we used for all the non-Kaldi pitch features is based on [**?**, p.46,54], and is similar to the system of the "Swordfish" team [1] in the BABEL program (IARPA-BAA-11-02); our experimental setup is part of

[1] Thanks to Arlo Faria who developed the pitch processing for that system

the "Radical" team's larger system. First, if there are regions where the pitch extractor says there is no voicing, we interpolate the pitch values from the adjacent voiced region in a straight line across the gap; or for unvoiced regions at file boundaries, we continue the first or last pitch value. We also add a little noise to the pitch values at this point. Then, we take the log of the resulting pitch values. We then apply the mean subtraction, subtracting the mean of a window of length 151 frames, centered on the current frame. To the resulting pitch, we apply short-time smoothing, averaging over a centered window of 5 frames. The reason why it is necessary to add noise and do short-time averaging is that many pitch extractors (including SAcC) output pitch, that is quantized to discrete values, produces a "blocky-looking" pitch trace. These operations help make the pitch trace smoother.

The POV feature is obtained as follows, and note that for all baseline systems we used the POV estimates from SAcC. We used $\log((p+0.0001)/(1.0001-p))$ as the feature, where $0 \leq p \leq 1$ is the POV estimate from SAcC. So the output is two features representing pitch and POV. We append these to the PLP features, and treat them the same way we would treat extra PLP coefficients (i.e., we apply cepstral mean subtraction, and delta computations or splicing followed by LDA).

2.1.4 Probability of voicing measure

Processing NCCF into a probability of voicing measure

The basis for our POV measures are the NCCF values for each frame. Their range is $[-1, 1]$, but it is usually positive. We process the raw NCCF value in two ways.

Accurate probability of voicing

The first method is only used as a part of the pitch mean-subtraction algorithm we describe below; it processes the NCCF value into a reasonably accurate POV measure. The following formula was obtained by plotting the log of count(voiced) / count(unvoiced) on the Keele database as a function of the NCCF, and manually creating a function to approximate it.

Let the NCCF on a given frame be written c. Compute its absolute value:

$$l = -5.2 + 5.4\exp(7.5(a-1)) + 4.8a - 2\exp(-10a) + 4.2\exp(20(a-1)) \qquad (2.6)$$

Here, $a = |c|$ and l is intended to approximate the log-likelihood ratio $\log(p(\text{voiced})/p(\text{unvoiced}))$. Then let $p = 1/(1+\exp(-l))$, and p is a reasonable approximation to the POV on this frame.

Method for use as a feature

The other method we use to process the NCCF produces a value that seems to result in good performance when used as a feature. This method was designed to give the feature a reasonably Gaussian distribution (although there are still noticeably separate peaks for voiced and unvoiced frames). If $-1 \leq c \leq 1$ is the raw NCCF, we let the output feature be $f = 2\left((1.0001 - c)^{0.15} - 1\right)$.

2.1.5 Normalization of pitch

We use the short-time mean subtraction approach of [**?**], however, for the POV weighting: on each time t we subtract a weighted average pitch value, computed over a window of width 151 frames centered at t and weighted by the POV value p described in Subsection 2.1.4. Please note that the improvement in WER from incorporating the weighting in the mean subtraction was quite small: of the order of 0.1% WER averaged across various languages.

2.1.6 Delta feature

We have extended our post-processing by adding a third feature consisting of the delta-log-pitch computed directly from the un-normalized log pitch, in the usual way (using ± 2 frames of context). The motivation was to get an exact delta-pitch feature without inaccuracies caused by the moving-window mean subtraction. This, together with the previous two features, is appended to the

raw MFCCs or PLP and shows around 0.4% absolute improvement on top of the results we present below.

2.1.7 Results

Kaldi BABEL pipeline

Our system is mostly as described in [**?**], although we have made various improvements since then. We measure our systems using %WER and using actual term weighted value (ATWV), which is a measure of keyword search effectiveness [**?**]; larger values are better. We train on the so called LimitedLP training data, which is around 10 hours $\times$ number of languages.

Of the languages we tested, only Vietnamese[2] and Cantonese[3] have tone marked in the dictionary. We configured Kaldi in such a way that the acoustic decision tree can ask about tone. Of the other languages, Zulu[4] is the only one that is considered to be a tonal language, but its lexicon is not marked for tone. We also tested on Assamese[5] and Bengali[6]. In all cases, we tested on the official BABEL Dev10h development set. For the keyword search, the development keyword phrase lists, usually provided by other participants in the program, lists were used. Please note, while we see improvements in WER/ATWV

[2]Language collection release IARPA-babel-107b-v0.7.
[3]Language collection release IARPA-babel-101b-v0.4c_sub-train1
[4]Language collection release IARPA-babel-206b-v0.1d
[5]Language collection release IARPA-babel-102b-v0.4.
[6]Language collection release IARPA-babel-103b-v0.3.

by using pitch, even in the atonal BABEL languages, separate experiments on Switchboard English showed no improvement from our features, compared with just MFCC. Perhaps, English is exceptional in some way.

In some of our experiments, we appended fundamental frequency variation (FFV) features [**?**]. These are seven-dimensional features which are informative about pitch changes. These features were part of our standard pipeline when our pitch features were based on SAcC, but we find that they are not helpful in combination with the features from our improved pitch tracker.

Experimental results

Figure 2.1 compares our pitch tracker with various baselines, using the Keele corpus. Our pitch tracker provides substantially better accuracy than the others (but bear in mind that we tuned it on this setup and that the error ratio of some of the baselines may be inflated because they classified some frames as unvoiced).

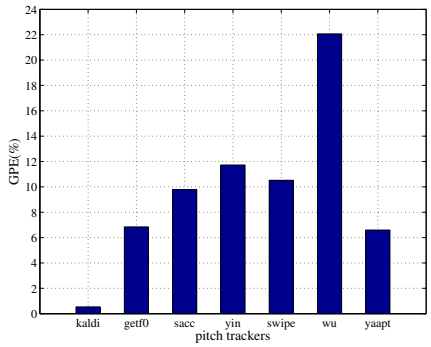

Figure 2.1: Gross pitch error (% voiced frames >10% pitch error): Keele database

In Table 2.2, we compare the Yin, getf0, and SAcC pitch trackers on Vietnamese data. Because not all the pitch trackers provide a POV, we used the SAcC POV. We also added FFV [**?**] features, as these were part of our original SAcC-based recipe. SAcC was still the best of the original pitch trackers we tested, but due to the simplicity of getf0 we felt it was the best starting point for our work.

Features	WER
Yin pitch + SAcC POV + FFV	68.1
GetF0 pitch + SAcC POV + FFV	68.0
SAcC pitch + SAcC POV + FFV	67.6

Table 2.2: Off-the-shelf pitch extractors (Vietnamese LimitedLP)

		Vietnamese		Cantonese	
Pitch	POV	%WER	%ATWV	%CER	%ATWV
-	-	71.3	20.0	63.3	18.5
SAcC	SAcC	68.9	22.0	60.6	20.8
getf0	SAcC	68.8	21.3	60.1	20.0
Kaldi	SAcC	67.1	24.0	58.1	24.1
Kaldi	Kaldi	65.6	27.1	56.5	23.5

Table 2.3: Comparing pitch and POV algorithms on tonal languages

Table 2.3 shows various combinations of pitch and POV features, on tonal languages, without FFV features. As shown, the Kaldi pitch and POV features are each better than the corresponding SAcC-based feature.

We tested our pitch extractor on two atonal languages, Bengali and Assamese. The Kaldi pitch tracker has good performance in atonal languages too. Figure 2.2 shows that adding our tone features results in very good gains

in these languages. So as can be seen, we get consistent improvement in tonal and atonal languages on the ASR system.

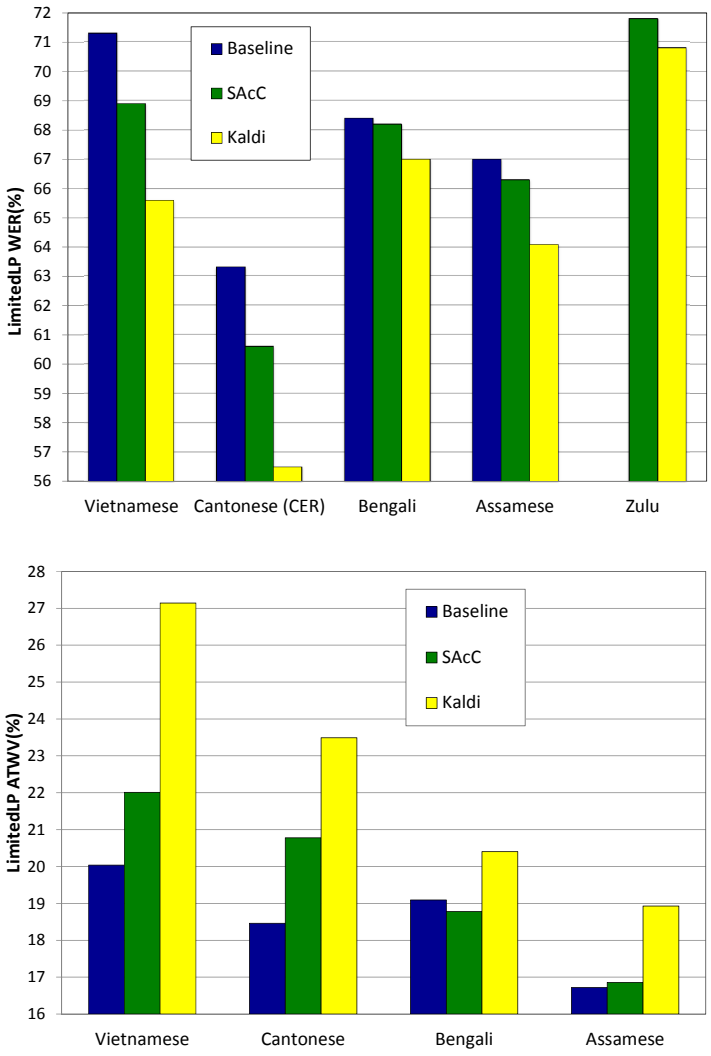

Figure 2.2: (a) WER (upper figure) (b) ATWV (lower figure) results on BABEL LimitedLP *Dev10h* for No-pitch vs. SAcC Pitch+POV vs. Kaldi Pitch+POV

Chapter 3

Joint feature extraction and classification training

In recent years, different studies have proposed different methods for DNN-based feature extraction and joint acoustic model training and feature learning from a raw waveform for large vocabulary speech recognition. However, conventional pre-processed methods such as MFCC and PLP are still preferred in the state-of-the-art speech recognition systems as they are perceived to be more robust. Besides, the raw waveform methods do not significantly outperform the conventional methods. In this chapter, we investigate direct-from-signal joint feature learning, and propose new time-domain (i.e., Section 3.1) and frequency-domain (i.e., Section 3.2) setups, which allow doing acoustic modeling directly from the raw waveform.

3.1 Time-domain joint feature extraction

3.1.1 Prior work

Most conventional speech recognition systems use handcrafted spectral and cepstral features such as MFCC [**?**], PLP [**?**] and Mel filter bank. All of these are inspired by physiological models of the human auditory system and may not be the most appropriate features for the final ASR objective of WER reduction. DNNs have been shown to be able to easily integrate the feature extraction stage with the classification stage. In [**?**], it was shown we could learn Mel-like filter banks from the power spectrum. Tuske et al. [**?**] proposed using the raw time signal directly as input to a DNN. Since then there have been many publications [**?**, **?**, **?**, **?**, **?**, **?**] analyzing methods to learn directly from the raw waveform. To the best of our knowledge, [**?**] is the only research where a raw waveform system is shown to give a better recognition performance than conventional features. However, they only report results using a large training set. In [**?**], the authors show improvement over conventional features only after appending the raw waveform to the conventional features. It is not clear how any of these approaches would perform on standard LVCSR tasks relative to the state-of-the-art systems that include speaker adaptation. In this section, we show that it is possible to beat the recognition performance of a state-of-the-art MFCC-based DNN system [**?**] on the Wall Street Journal (WSJ) task

and match the performance on the Switchboard [**?**] task.

One of the issues with learning directly from the raw waveform is the large number of variations for a given phoneme in the form of phase shifts and temporal distortions. The input to the neural network (NN) is at a very fast rate (8-16 kHz), and using a broad temporal context would result in requiring a sizeable linear layer in DNN. It results in difficulties in performing backpropagation through time in an RNN. A typical approach dealing with these issues is to perform max-pooling over time [**?**]. In [**?**], conventional pooling approaches such as max, p-norm, and averaging functions are compared. Bhargava et al. [**?**] use a pre-trained bottleneck neural network to extract features that are spliced at a slower 10 ms period over a long temporal window. In this section, we present a novel NiN [**?**] architecture that aggregates filter outputs over time. The NiN is randomly initialized and jointly trained with the rest of the network. We show that with this architecture, the network can train just as fast as our baseline MFCC-based DNN.

In the realm of traditional features, speaker adaptation is usually done using a generative framework that involves transforming features to a different space using fMLLR [**?**] or applying a speaker-dependent bias by appending features like i-vectors [**?**, **?**]. However, i-vectors are not straightforward to work with, especially in mismatched conditions [**?**], and requires careful preprocessing such as segmentation and new architectural tricks [**?**]. We could

not get i-vectors working as well in the raw waveform setup as in the MFCC setup. Instead, we experiment with an adaptation approach that uses activation statistics of hidden units accumulated over about a 2-second long window. We show that using this approach eliminates the performance gap compared to the state-of-the-art MFCC-based DNN system with i-vector speaker adaptation.

3.1.2 Raw waveform processing

The input frames to the neural network are non-overlapping 10 ms long segments of the raw waveform signal. The raw waveform samples are quantized to 16 bits per sample and the mean and variance are normalized at the utterance level. The mean-variance normalization can be important for stable training [**?**].

3.1.3 Data perturbation

The Fourier modulus is translation invariant and stable to additive noise but unstable to small deformations at high frequencies [**?**]. FFT-based features such as MFCC and PLP are invariant to modest translations. A large amount of variations in the raw waveform input for a given phoneme can be detrimental to training. One approach to mitigate this is to artificially perturb the

data to make the network invariant to those perturbations. We exploit different audio augmentation techniques at signal level. The data is augmented by changing the speed of the audio signal and producing different versions of the original signal with speed factors in the range $[0.9, 1.1]$ [**?**]. We also do this for the baseline MFCC setup. To achieve translation invariance, we alter the raw input signal during training by shifting the samples randomly to the right for up to 20% of the frame window size. This means that in different epochs, we might see the same data at different shifts.

Data perturbation can significantly help in improving performance on small datasets such as WSJ. Table 3.1 shows the effect of random shifts on final validation and training log-likelihoods.

Table 3.1: Data perturbation effect on WSJ

Perturbation method	Training CE	Validation CE
No random shift	-0.96	-1.22
With random shift	-0.88	-1.13

3.1.4 Pooling methods

The input speech frame to the neural network in the direct-from-signal setup is at a very fast rate (e.g., 8-16 kHz) and using a wide temporal context would result in requiring a large linear layer. A typical approach is to pool over time, where a max [**?**], p-norm [**?**] and average pooling are the conventional pooling methods. Improper down-sampling of the output of wideband

filters can lead to severe aliasing which is not a reversible operation [**?**]. In this section, we present a new NiN [**?**] architecture that aggregates filter outputs over time. We use this structure as a pooling layer to aggregate the filter's output (see Figure 3.3).

Network-in-network (NiN) nonlinearity

In this section, we introduce a new type of nonlinearity that is a special case of the NiN nonlinearity proposed in [**?**]. It is a many-to-many nonlinearity consisting of two block diagonal matrices, with repeated blocks, interleaved between layers of rectified linear units (ReLU). A normalization layer [**?**] is always added after the NiN nonlinearity to stabilize training. Figure 3.3 shows a graphical representation of the nonlinearity.

The transformation block U_1 of size $m \times k$ maps an input of size m into a higher dimensional space with dimension k, and it is subsequently passed through ReLU nonlinearity. We will refer to the quantity k as the "*NiN hidden dimension.*" The second transformation block U_2 of size $k \times n$ maps it down to a lower dimensional space with dimension n followed by another rectification using ReLU nonlinearity. We will refer to the combination of U_1 block and U_2 block along with the ReLUs as a "*micro neural network block*" as marked in Figure 3.3.

To concisely describe the proposed NiN nonlinearity, we can say that it is a

group of *micro neural network blocks* applied to non-overlapping patches of the input with each block being a nonlinear transformation from m dimensional space to n dimensional space.

If the *micro neural network block* parameters are shared across the NiN, each column of block U_1 can be interpreted as a 1-d convolution filter with a filter size m and a filter shift m. Thus, the same filter is applied to non-overlapping patches to model local connectivity. The shared parameters in the NiN nonlinearity keep its total parameter count low relative to the size of its input and output and allow it to train faster.

We use this nonlinearity at the output of the convolution layer. This type of nonlinearity is helpful in reducing variability and pooling the information without having too many parameters to learn. This nonlinearity learns very powerfully with fewer parameters than the conventional ReLU-based layer, and is constructive for the raw waveform setup.

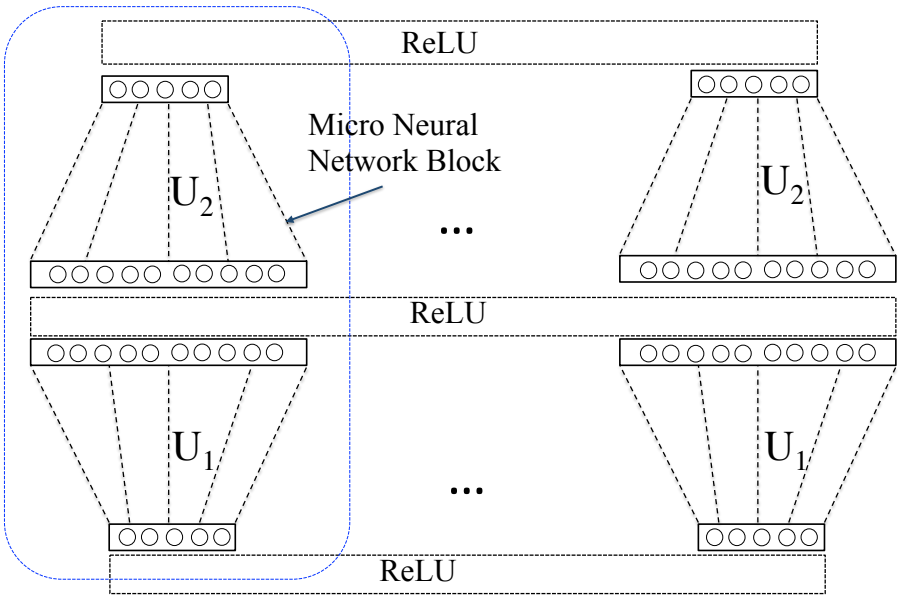

Figure 3.1: Proposed NiN nonlinearity

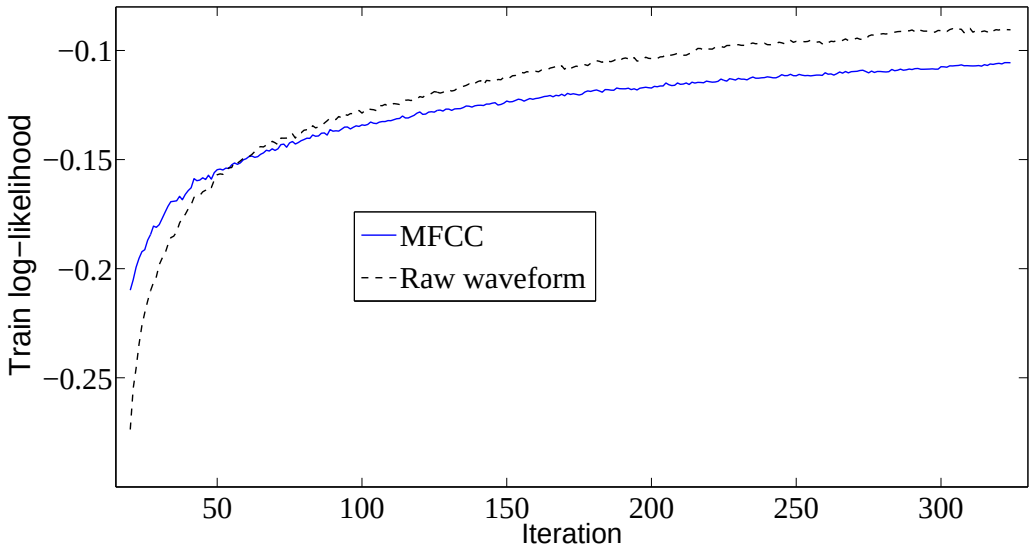

Figure 3.2: Convergence of training objective function in raw waveform setup using NiN nonlinearity vs. MFCC setup using ReLU

Comparison with other pooling methods

We compare the effects of different pooling techniques (i.e., no pooling, max [**?**], p-norm [**?**]) with our proposed NiN pooling structure in Table 3.2. The overall feature extraction block used in all experiments is similar to the architecture described in Subsection 3.1.6 and the main difference is that different pooling techniques are used instead of the NiN block in Figure 3.5. The classification block used in all experiments is the same as the layer structure described in Figure 3.5.

As shown in Table 3.2, NiN pooling as a trainable pooling layer outperforms conventional pooling methods. This layer is a second level time-convolution layer, which enables the network to exploit various sampling rates.

Table 3.2: WER (%) results on the Switchboard LVCSR task

	Hub5'00	
Model	Total	SWBD
No pooling	27.4	18.8
Max pooling	17.3	11.8
p-norm pooling	16.7	11.4
NiN pooling	15.9	10.4

3.1.5 Speaker adaptation in raw waveform setup

Statistic pooling layer

The statistic pooling layer extracts 1^{st} and 2^{nd} order statistics from hidden layer activations. These statistics are computed over a moving window of up to

200 frames (2 seconds) and appended to the input of the next hidden layer.

Given an n-dimensional input, this layer computes a $2n$ dimensional output consisting of

1. n dimensions of the moving average mean of the input
2. n dimensions of the raw diagonal 2^{nd}-order statistics

We expect this layer to capture long-term effects in the signal such as a speaker, channel and environment characteristics. This is particularly useful in the raw waveform setup as the raw signal has more information related to these characteristics, which are not in MFCCs.

3.1.6 CNN-based raw waveform setup

Our raw waveform setup consists of two parts – a feature extraction block and a classification block. The feature extraction block, described in this subsection, consists of a CNN layer to process the raw waveform samples. The CNN outputs are aggregated using the proposed NiN nonlinearity. The classification block in our setup uses the basic time-delay neural network (TDNN) architecture [**?**], but it uses the proposed NiN as the nonlinearity instead of ReLU. This is described in detail in this section.

Feature extraction block

The feature extraction block used in the raw waveform setup is illustrated in Figure 3.5. We take M samples of the raw waveform and convolve them with N K-dimensional filters in the 1-d convolution layer. S is the step size taken along the input axis for computing the next dot-product of input and filters. Using the step size is equivalent to subsampling the output of convolution by a rate of S. This helps in reducing computation time. The output dimension of the convolution layer is $N \times D$, where $D = \frac{M-K}{S} + 1$. Next, we take the absolute value of the filter outputs and take the log.

The major difference in our CNN architecture compared to the conventional setups such as the ones in [**?**,**?**,**?**] is the use of a NiN nonlinearity. These setups use conventional pooling techniques such as max pooling over time to reduce the output of the convolutional layer to $N \times 1$ from $N \times D$. Our proposed NiN nonlinearity (see Subsection 3.1.4) takes the place of this pooling layer. The U_1 block of the first NiN nonlinearity is chosen to be of dimension $D \times k$, where k is the NiN hidden dimension that is typically around $5D$. A single *micro neural network block* is shared across all N filters. Each *micro neural network block* aggregates information over D samples.

HOW DOES NIN POOLING WORK?

The output y, for a time-convolution layer can be written as

$$y_{k,t} = \sum_{\tau=0}^{K} s_{t+\tau} \cdot h_{k,\tau} \tag{3.1}$$

s_t is the input signal, sampled at $8kHz$ for Switchboard experiments and $y_{k,t}$ is the subsampled output, where k is a filter index and t is subsampled every s frames. $h_{k,t}$ is the k^{th} FIR filter impulse response with a length of 31.25 ms and 250 samples. The nonlinearity function applied to the output of the convolution layer can be interpreted as envelope extraction, which contains rectification, low-pass filtering and sub-sampling. We subsampled the output of convolution $y_{k,t}$ after every 10 samples, which has a fixed 6.25 ms rate. The NiN layer generalizes the downsampling block and makes it a trainable block as follows:

$$x_{j,k,t} = f_3\Big(\sum_{i=0}^{M} f_2\Big(\sum_{\tau=0}^{N} f_1(y_{k,t+\tau}) \cdot l_{i,\tau,k} \Big) \cdot z_{i,j,k} \Big) \tag{3.2}$$

$l_{i,\tau}^{k}$ is the trainable low-pass filters with size M for filter k and the results show that sharing the low-pass filter envelopes across filters results in the best per-

formance. ReLU nonlinearity (i.e., $max(0, x)$) is used in most direct-from-signal models [**?**]. We used two different nonlinearity functions, absolute function $|x|$ and rectification $\max(0, x)$, as $f_1(x)$ on the output of pooling and the results show 0.5% absolute WER improvement using absolute function on the Hub5'00 Switchboard test set. f_3 is the ReLU function applied after the first nonlinearity block in the NiN block and $l_{i,k}$ is the i^{th} block in the NiN block. Figure 3.3 shows learned envelopes l_i with f_1 as $|x|$ learned on Switchboard datasets, where the NiN blocks with size 75×16 are shared across all filter outputs and 75 envelopes are trained as described in Equation 3.2.

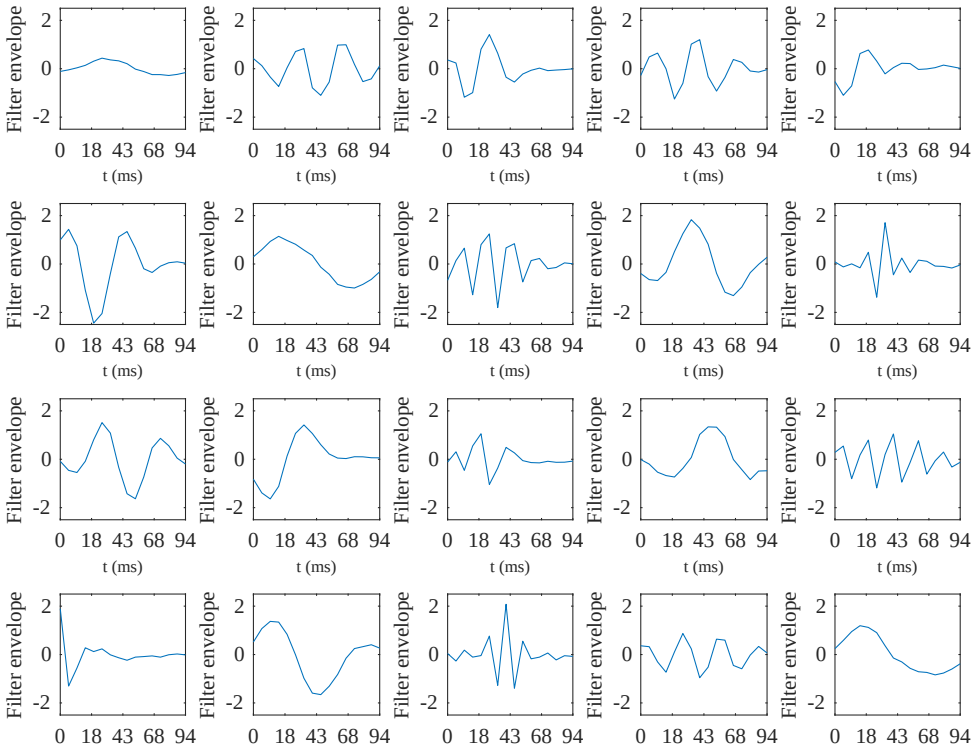

Figure 3.3: Learned envelopes l_i in the NiN pooling layer

COMPRESSION METHODS

The two common compression methods used to reduce dynamic range in feature extraction are as follows:

1. 10^{th}-root compression: It is shown in [**?**] to work better in time-filtered Gammatone features.

2. The log compression method: The two common approaches to tackle the

log singularity at 0 are stabilized log (i.e., $\log(x + \delta)$) and clipped log (i.e., $\log(\max(\delta, x))$).

Table 3.3 shows the results of a Switchboard subset of 100 hrs using different compression methods. As shown, two logarithmic approaches produce almost the same results in our raw waveform setup.

Table 3.3: WER (%) results of a 100 hours Switchboard LVCSR task using different compression methods

	Hub5'00	
Model	eval2000(Total/callhome)	rt03
No compression	16.0/21.0	21.2
Stabilized log, $\delta = 10^{-10}$	15.8/20.9	21.6
Stabilized log, $\delta = 10^{-2}$	15.9/21.3	21.6
Clipped log, $\delta = 10^{-4}$	16.3/21.6	21.7

We have two consecutive layers of the proposed NiN nonlinearity. The normalization layer, that we use after each nonlinearity, scales down the output and keeps its averaged norm in check.

In speaker adaptation experiments using a raw waveform setup, i-vectors are appended to the NiN output at this stage after first being passed through a separate affine component and a ReLU layer [**?**]. In the MFCC setup, we append and pass the i-vectors and MFCC through an LDA transformation [**?**].

Figure 3.6 compares the convergence rate using 3 different approaches to pool over time. In these experiments, we use the same classification block architecture, but different pooling methods on the outputs of the convolution layer in the feature extraction block. As shown, using NiN to aggregate out-

puts converges faster than both p-norm pooling and using no pooling. Our experiments also demonstrate that the NiN aggregation shows a 1% WER improvement over using p-norm pooling.

Classification block

Figure 3.5 shows the structure of a DNN layer used in the classification block. We use a TDNN architecture [**?**] to splice D_3 dimensional inputs at different time steps $t_1, t_2, \cdots, t_n$, but also append to this the moving statistics extracted using the statistics extraction layer (Subsection 3.1.5). Then, we rearrange the dimensions of the spliced input so that the L shared *micro neural network blocks* in the NiN nonlinearity are applied on $d = \frac{D_3}{L}$ dimensional patches of input data. The d-dim input patches are extracted from all the n time steps and they are appended with the statistics extracted over time steps $t_l, \cdots, t_r$ for the same d dimensions. The NiN nonlinearity is followed by a normalization layer and a full affine transformation to reduce the output dimension to D_3. We stack several layers of this type to form the classification block.

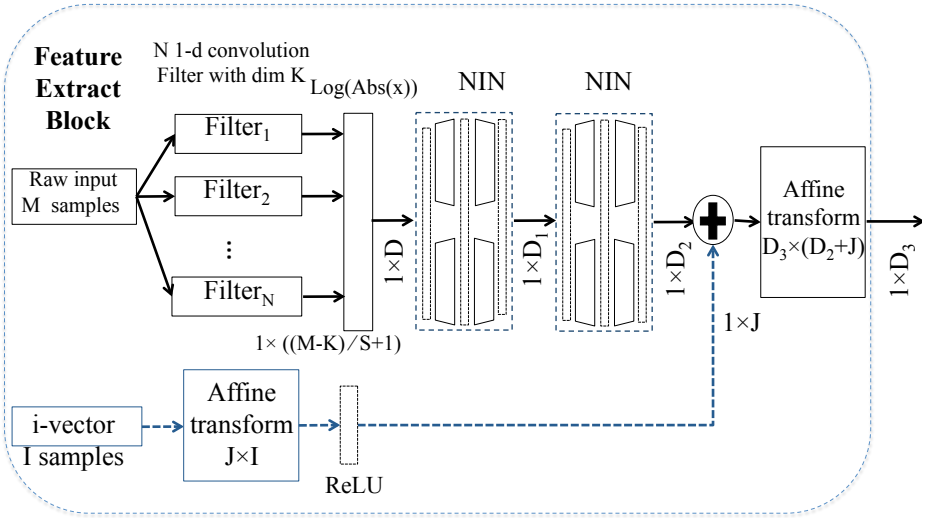

Figure 3.4: Raw waveform feature extraction block

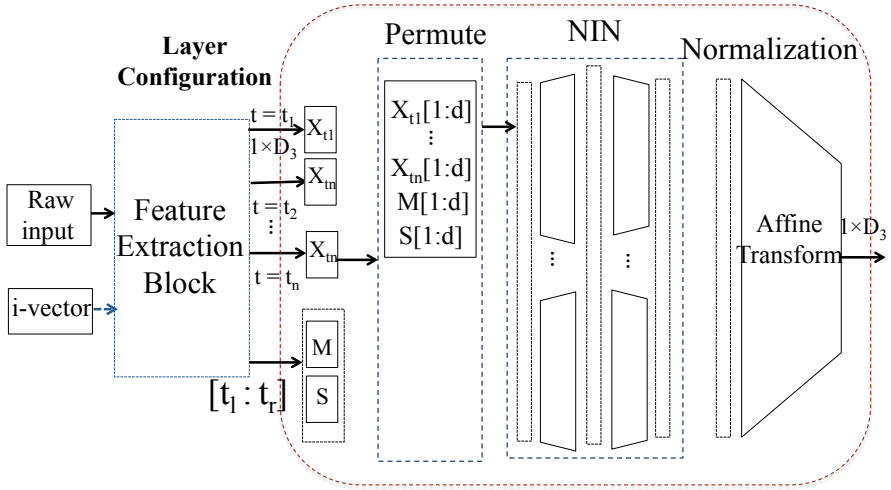

Figure 3.5: Layer configuration in raw waveform classification block

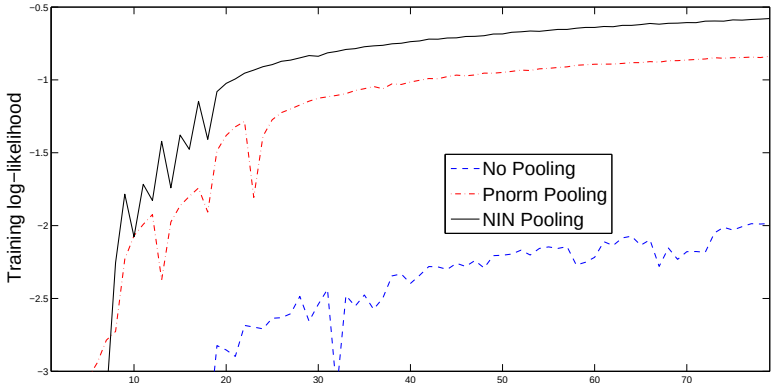

Figure 3.6: Training log likelihood for different pooling methods

Effect of regularization

Mel filters are sparse and narrowband in the frequency domain. One of the main challenges in learning time-domain filters is to learn narrowband filters, which are non-uniformly spaced across frequency. We investigate different regularization techniques such as l_1 regularization on the frequency domain transformation of time-domain filter weight, and l_2 regularization on filter weights. Dropout [**?**] is another regularization technique, which randomly sets some portion of activation to be zero during training and helps to improve model generalization and prevents over-fitting.

L_1 regularization

We minimize L_1-norm for convolution filter w_i trained in time-domain in FFT domain

$$L_1(Z_i) = \sum_{j=1}^{N} \left| Z_{ij} \right| = \sum_{j=1}^{N} Z_{ij}$$

$$Z_i = \left| \mathrm{FFT}(\mathrm{w_i}) \right|$$

L_2 regularization

We used a modified version of L_2 regularization, proportional shrink. It scales down parameters in the convolution layer with some scaling factors proportional to training iteration and learning rate. Figure 3.7 shows the learned filter banks using different regularization techniques. As can be seen, the proportional shrink as an extension to L_2-regularization produces a higher number of narrowband and sparse filter banks. Also, L_2-regularization demonstrates the best WER results compared to the others.

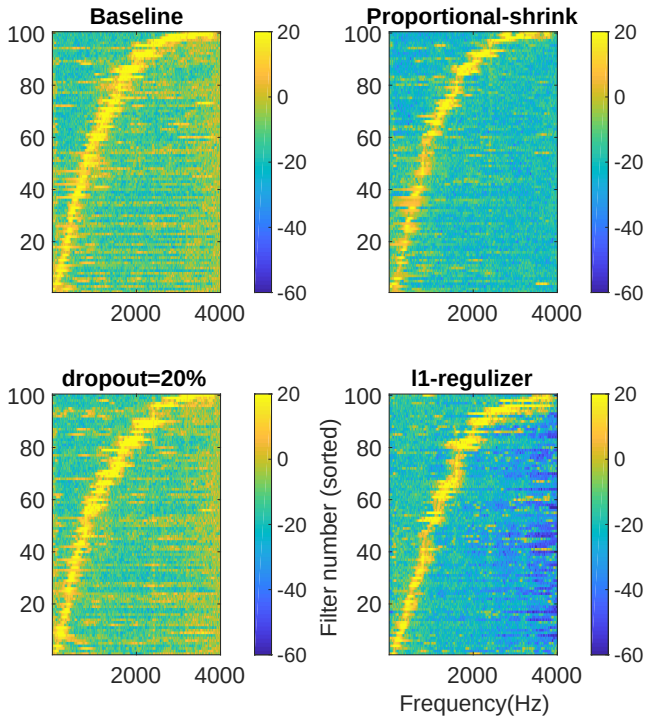

Figure 3.7: Learned filter banks using different regularization techniques

3.1.7 Results

We conduct our experiments on two corpora – 300 hours Switchboard conversational telephone speech corpus [**?**] and 80 hours WSJ continuous speech corpus [**?**]. All of our experiments are conducted using the Kaldi Speech Recognition Toolkit [**?**]. For the baseline, we use DNNs in time-delay neural network architecture with 40-dim MFCC features as input. For speaker-adapted systems, 100-dim i-vectors were appended to the input features. The reader is directed to [**?**] for the architectural details.

WSJ task

The raw waveform setup used in WSJ experiments is as described in Subsection 3.1.6, but we use p-norm pooling instead of the NiN nonlinearity, and the classification block uses a conventional TDNN layer. The networks here, including the MFCC baselines, are trained using a frame cross-entropy objective. The statistics extraction layer is not used in these experiments as the cross-entropy model trains on small chunks over which we cannot extract reliable statistics.

The CNN layer in the feature extraction block consists of $N = 40$ filters. The filter size used is 30 ms on a raw waveform signal that is sampled at 16 kHz. The filter step size used is 0.62 ms. The output of the convolution filters are pooled over time using p-norm pooling instead of using NiN nonlinearity. The classification block has six hidden layers, each with 750 ReLU units. Table 3.4 compares the results of our raw waveform setup and the MFCC-based TDNN system on the WSJ 5K vocabulary task. The first two rows represent the systems without speaker adaptation. We see that the raw waveform system performs more than 1% absolute better than the MFCC baseline.

From the next two rows, we see that adding i-vectors to the MFCC system improves the MFCC baseline but still does not beat the raw waveform setup without i-vectors. This may indicate that our raw waveform setup is less sensitive to speaker mismatches. Adding i-vectors to the raw waveform

setup degrades the result. In the final experiment, we tried adding the NiN nonlinearity but could not get any improvement over ReLU.

Table 3.4: WER (%) results on the WSJ LVCSR task

Model	*Nov'92 eval*	*Nov'93 dev*
MFCC	5.28	8.29
Raw	3.95	7.34
MFCC + i-vector	4.52	7.51
Raw + i-vector	4.06	7.80
Raw + i-vector + NiN	4.13	7.71

Switchboard task

Table 3.6 compares the results of the proposed raw waveform system and the MFCC-based TDNN system on the Switchboard task. The results are reported for both the *Hub5'00* and the *RT'03* evaluation sets.

In the raw waveform setup, the CNN layer consists of $N = 100$ filters. The filter size used is 31.25 ms on a raw waveform signal that is sampled at 8 kHz, and the filter step size is 1.25 ms. The feature extraction block uses NiN nonlinearity with 100 *micro neural network blocks* with input size $m = 16$ (same as the convolution filter output dimension), NiN hidden dimension $k = 120$ and output size $n = 18$. The output dimension of the feature extraction block is $D_3 = 500$. The classification block has 6 hidden layers, with either ReLU nonlinearity (600 hidden units) or NiN nonlinearity. The NiN nonlinearity has 100 *micro neural network blocks* with input size $m = 5$, NiN hidden dimension $k = 75$ and output size $n = 18$. The neural networks are trained using lattice-

free MMI [**?**]. In the experiments using the statistics extraction layer, mean and standard deviation of the hidden layer, activations are computed over the available frames on either side for up to a maximum of 99 frames.

Table 3.5 shows the effect of using NiN nonlinearity in the classification block in the raw waveform setup. Both the ReLU and NiN systems have the same TDNN structure regarding context [**?**] and use the same feature extraction block including i-vectors. We see that using NiN nonlinearity demonstrates 1% improvement over the conventional ReLU nonlinearity in the raw waveform setup. However, we found that the NiN nonlinearity does not show any improvement over ReLU on the MFCC setup.

Table 3.5: Effect of NiN nonlinearity

	Hub5'00		*RT'03*	
Model	Total	SWBD	Total	SWBD
ReLU	17.2	11.5	19.9	24.0
NiN	16.1	10.5	18.9	23.1

Table 3.6 compares the raw waveform setup with NiN nonlinearity and the MFCC setup with ReLU nonlinearity. The first two rows in the table are the results without speaker adaptation.

The next two rows show the effect of adding the statistics extraction layer. We see that the statistics extraction layer improves the performance of both MFCC and raw waveform setups. The raw waveform setup works slightly better than the MFCC setup, which may indicate that the statistics layer helps

the network to extract some speaker or channel dependent information directly from the raw waveform, which may be removed during the MFCC extraction process.

The last two rows show the effect of speaker adaptation by adding i-vectors. We see that i-vectors show much improvement in the MFCC setup, but only a little improvement in the raw waveform setup. We hypothesize that the raw waveform possesses more information than the MFCC features and the network can learn to account for the speaker and environment variability. However, we need to perform more experiments to verify this hypothesis.

Table 3.6: WER (%) results on the Switchboard LVCSR task

	Hub5'00		*RT'03*	
Model	Total	SWBD	Total	SWBD
MFCC	17.5	11.6	22.1	26.6
Raw	17.4	11.5	21.7	26.5
MFCC + Stats	16.4	11.0	20.0	24.3
Raw + Stats	16.3	10.6	19.1	23.3
MFCC + i-vector	15.7	10.4	19.2	23.5
Raw + i-vector	16.1	10.5	18.9	23.1

3.2 Frequency-domain joint feature extraction

In the work presented in this section, we simplify our previous approach [**?**] (i.e., time-domain feature learning) by operating in the frequency domain. That is, we include a Fourier transform layer in the network and let the network learn the filter banks in the frequency domain. Frequency-domain feature learning has been previously used in [**?**] and [**?**], however, we propose a new normalization layer which helps with stabilization and better convergence of the filters. Additionally, we employ a different weight constraint approach which further improves the results. We use the proposed frequency-domain layer in the state-of-the-art LF-MMI setup and show significant WER improvements on various well-known large vocabulary databases. Time-domain feature learning is explained in detail in Section 3.1. In Section 3.2, our proposed frequency-domain approach, as well as previous work on frequency-domain feature learning, is described. The experiments and results are presented in Subsection 3.2.4.

3.2.1 Prior work

Most of the data-driven feature learning approaches in recent years have attempted to do feature learning directly from the time-domain waveform. Tüske [**?**] trained a DNN acoustic model on waveforms and showed that auditory-like filters could be learned using fully connected DNNs. Other research studies usually use time convolution layers, which share weights across time shifts [**?**, **?**, **?**].

The first layer in a time-domain feature learning setup is usually a time-convolution layer, which is like a finite-impulse-response filter bank followed by a nonlinearity. This layer is expected to approximate the standard filter banks, which are often implemented as filters followed by rectification and averaging over a small window. The output of this layer can be referred to as time-frequency representation. Next, the rectification or absolute function is applied to the output of the convolution filters, and the log compression is used on the absolute value of the filter outputs to reduce the dynamic feature range. To the best of our knowledge, most of the reported results show performance degradation when using time-domain feature learning and [**?**] and [**?**] are the research studies where raw waveform setup slightly outperforms the conventional features. [**?**] proposed a new nonlinearity to aggregate filter outputs leading to results competitive with the state-of-the-art baseline systems.

In contrast to time-domain feature learning where the inputs to the CNN

and filter bank layers are raw speech samples, in the frequency-domain feature learning the samples are passed through a Fourier transform layer first [**?**,**?**,**?**].

In this study, we adopt a similar frequency-domain approach but with a few significant differences. Specifically, we use an extra normalization block, and constrain the weights in the filter bank layer to a short range. The details of our setup will be explained in the following subsections.

3.2.2 Proposed feature extraction block

The overall process of feature learning in our setup is shown in Figure 3.8. The input features of the neural network are non-overlapping 10 ms segments of the raw waveform signal. Each segment is represented by a vector of amplitude values (e.g., for 8kHz speech, the features will be 80-dimensional). Unlike acoustic modeling from time-domain [**?**], there is no need for input normalization in the frequency-domain setup. As shown in Figure 3.8, the input features are first passed through a pre-processing layer which performs pre-emphasis and DC-removal. Then they go through the Fourier transform layer which is implemented using sine/cosine transforms. L2-normalization is also applied to the output of the Fourier transform. The next step is the normalization block which is explained in Subsection 3.2.3. After normalization, there is the main filter bank layer. Implementation-wise, the filter bank layer is an NxM weight matrix (i.e., a linear transform), where each row represents an M-point

filter. The weights in this matrix constrained according to Equation 3.3 which is applied after updating the parameters of the filter bank for each mini-batch during training.

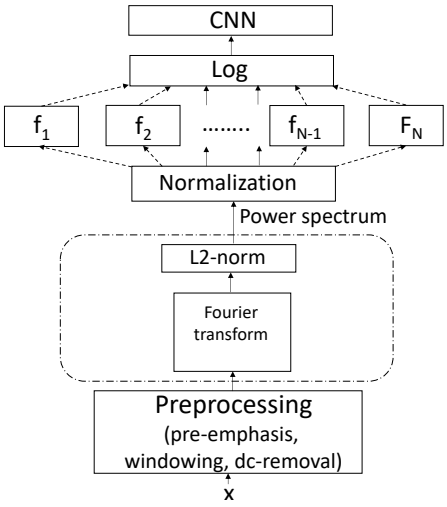

Figure 3.8: Frequency-domain feature extraction setup

$$\mathcal{W}_{ij}^{'} = \max(\alpha_1, \min(\mathcal{W}_{ij}, \alpha_2)) \quad \alpha_1 < \alpha_2 \tag{3.3}$$

We tried different values for α_1 and α_2 and determined that 0 and 1 produce the best results. Table 3.7 compares the different constraints we tried. In addition, we compared this method with the proposed method in [**?**], where the parameters are constrained to be positive by using exponentiation as $\exp(W_{ij})$ but found that our approach was more effective.

The filter bank layer is followed by log compression which is a common practice in DNN acoustic modeling, where the log compression helps to reduce the

dynamic range of the filter banks. We investigated two common log methods: (1) clipped log (i.e., $\log(\max(\delta, x))$) and (2) stabilized log (i.e., $\log(x+\delta)$) and found that clipped log was more effective which is what we use in this setup. Finally, the log filter bank features passed to a CNN layer. We use a 2-dim convolution layer with 32 filters with a size of 3×3, with time stride 2 instead of pooling with factor 2 in this setup.

Table 3.7: Effect of different filter bank constraint methods

Method		WER
Proposed weight constraint (α_1, α_2)	$(-\infty, \infty)$	15.9
	$(-\infty, 1)$	16.0
	$(0, \infty)$	14.5
	$(0, 1)$	**14.3**
exponential weights[1]		15.3

3.2.3 Normalization block

As suggested in [**?**], applying normalization before filter learning is beneficial. Distribution of the inputs can change during training and the first layer of the network is more sensitive to these changes which can slow down training or make it unstable. Therefore, we normalize the input power spectrum which helps to stabilize training and to better train narrowband filter banks. As shown in Figure 3.8, the inputs to the filter learning stage are normalized. This is shown in more detail in Figure 3.9. Specifically, we first transform the power spectrum features to log-space, where batch normalization is applied,

that is, normalizing the features over a mini-batch. We use batch normalization proposed in [**?**] which allows using much larger learning rates. After batch normalization, the outputs are normalized globally using mean and variance parameters that are jointly learned with other parameters during training. Finally, the parameters are transformed back into normal space using the exponential function.

We examine the effect of each component in the normalization block in Table 3.8. As shown, applying the normalization in log-space is crucial. Besides, batch-normalization has a significant effect on the final WER too.

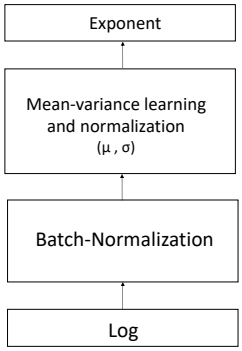

Figure 3.9: Normalization block

Table 3.8: Effect of different components in the normalization block

log-domain	batch-norm	global norm	WER
✓	✓	✓	14.3
✓	✓	✗	14.6
✗	✓	✓	17.2
✓	✗	✓	15.0

Scale and offset analysis

Figure 3.11 shows scale and offset values in a normalization block proposed in Subsection 3.2.3 on narrowband 8-kHz and wideband 16-kHz datasets. As shown, scales are larger in low-frequency FFT bins in log domain in $[0-1]kHz$ that is equivalent to applying a larger power in the normal domain. As shown, the smaller scale values applied in the high-frequency bins in the range $[6-8]kHz$ and the scale values are approximately 1 in the mid-frequency range $[1-6]kHz$. Smaller values for scales correspond to power 1 in the normal domain, which is equivalent to not changing any frequency values. The offset range on FFT bins in the log-domain is $[-0.5, 0.5]$, which is equivalent to per-dimension

FFT scaling in a normal domain with $[exp(-0.5), exp(0.5)] = [0.6, 1.6]$.

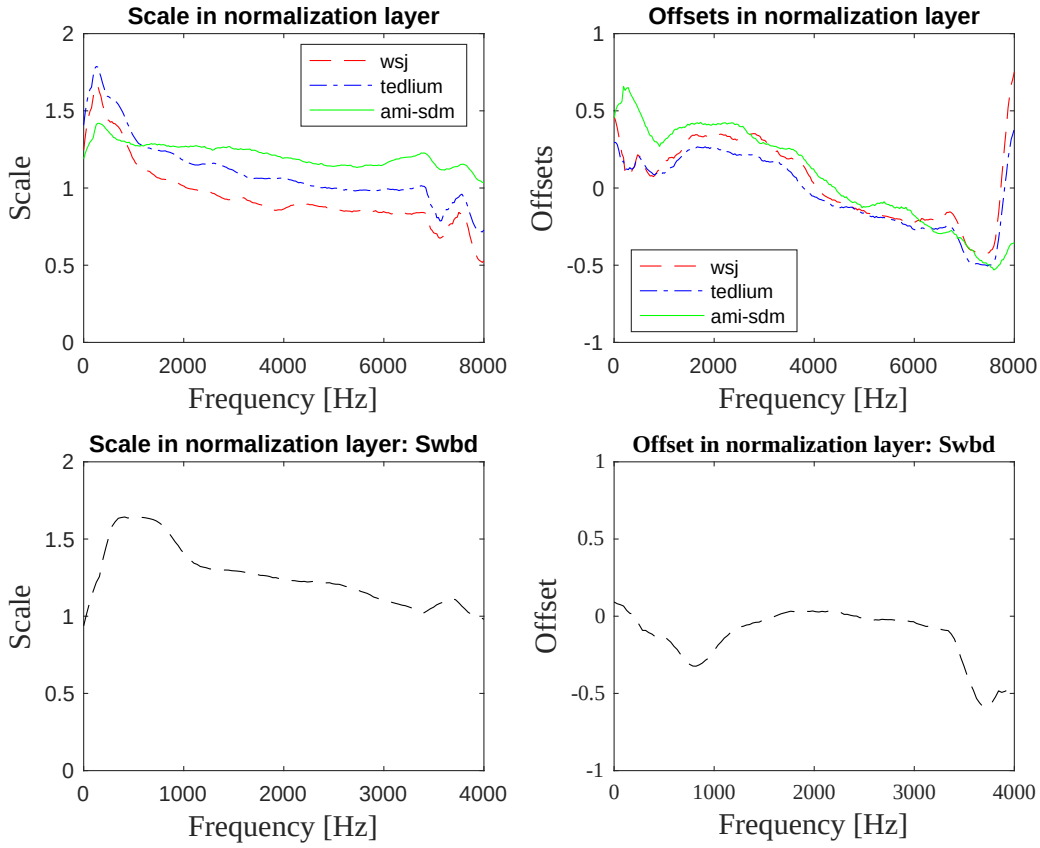

Figure 3.10: Scales and offsets vs. frequency for narrowband and wideband data in a normalization layer

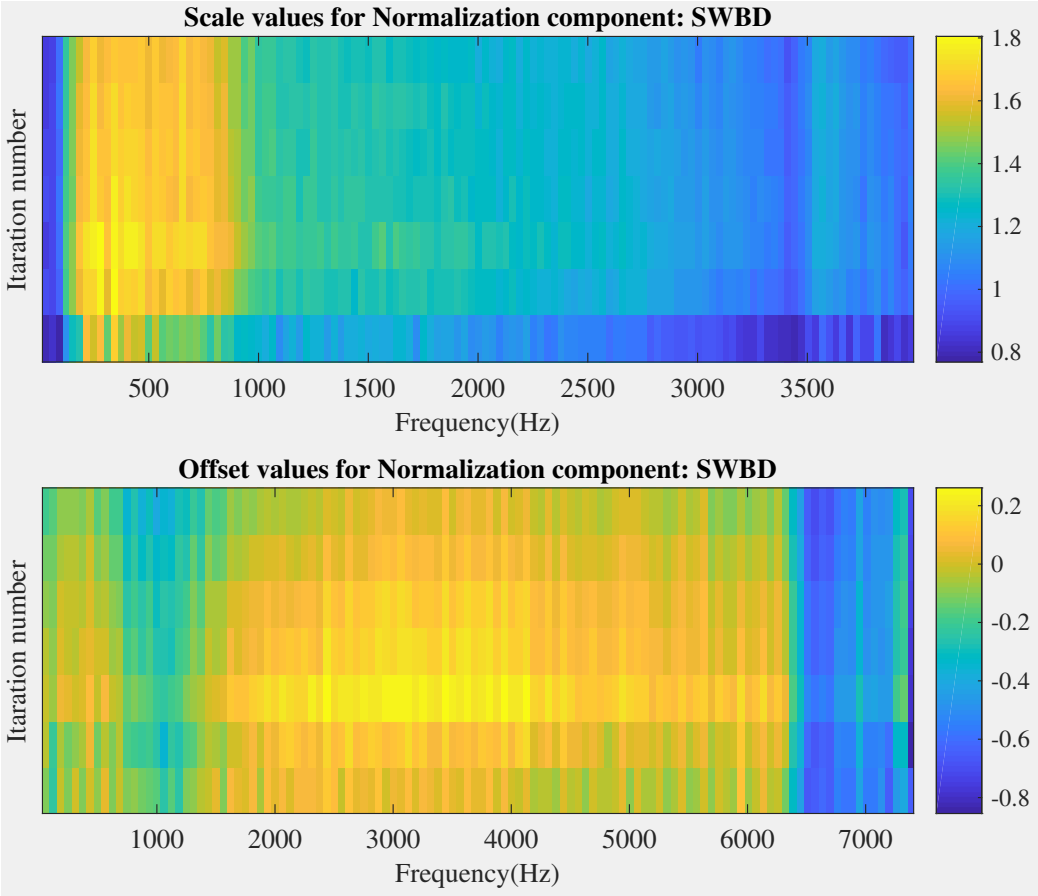

Figure 3.11: Convergence of scales and offsets in a normalization layer

3.2.4 Results

In this section, we compare our proposed frequency-domain setup with the time-domain setup proposed in [**?**] trained on the $300hrs$ Switchboard task. We evaluate the full Hub5 '00 set (also called "eval2000"). [2] We also compare two conventional baselines: MFCC and log-Mel filter bank features. The MFCC baseline system uses spliced 40-dimension MFCC feature vectors followed by

[2]We perform all the experiments using the Kaldi speech recognition toolkit [**?**].

an LDA layer. Note that the results for 40-dimension and 80-dimension MFCC features were the same (not shown). Mel features were generated by passing the power spectrum through a set of Mel filters, and log-Mel filter bank features were generated by applying a log compression on the Mel features. The log-Mel features – as well as all other feature learning layers we are comparing here – are followed by a CNN layer. The rest of the network structure is the same in all experiments (i.e., after the LDA or CNN layer). Specifically, we use blocks of TDNN layers [**?**] trailed by batch-normalization [**?**] and rectified linear units.

The results are shown in Table 3.9. The time-domain feature extraction setup used in the 3^{rd} row of this table is similar to [**?**]. We also show the results of training separate filters on real and imaginary parts of the Fourier transform as done in the complex linear projection (CLP) method proposed in [**?**]. Particularly, we train two separate filter banks, $\mathcal{W}_\mathcal{R}$ and $\mathcal{W}_\mathcal{I}$, on the real and imaginary parts of the signal's Fourier transform and the real and complex parts of the output computed as $\mathcal{W}_\mathcal{R}\mathcal{X}_\mathcal{R} - \mathcal{W}_\mathcal{I}\mathcal{X}_\mathcal{I}$ and $\mathcal{W}_\mathcal{R}\mathcal{X}_\mathcal{I} + \mathcal{W}_\mathcal{I}\mathcal{X}_\mathcal{R}$. $L2$-norm followed by log nonlinearity is also used to compute the log filter bank features. We can see that our proposed frequency-domain setup outperforms other frequency-domain, time-domain and conventional setups. In our setup, we used 40, 100, and 200 filters in the filter bank layer and all cases led to the same result shown in Table 3.9.

3.2.5 Filter analysis

Figure 3.12 shows the filter bank weights learned for the proposed frequency-domain setup with and without normalization. Clearly, normalization helps in learning less noisy filters.

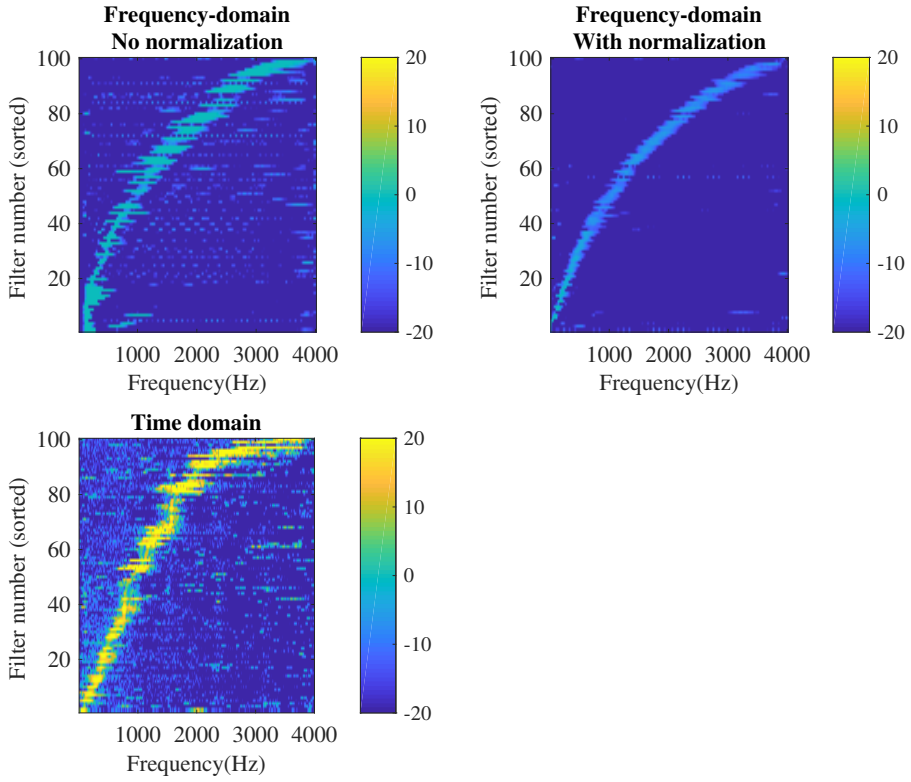

Figure 3.12: Magnitude response of learned filter ordered in center frequency

The filters learned in the filter bank layer are usually interpreted as a band-pass impulse response. One of the main issues in time-domain filter learning is that the filters are not usually narrowbanded and regularization is necessary. We use L_1 regularization on the Fourier transform of the filters learned in the time-domain setup which is helpful in learning narrowband filters. As can be

seen in Figure 3.12, this issue is alleviated in frequency-domain filter learning. The filter banks learned in this domain are narrowbanded, and few filters show multiple pass-bands. We apply L_2-regularization on filter bank weights in the frequency-domain and CLP setups.

Table 3.9: Frequency-domain vs. time-domain

Method	WER	
	eval2000	rt03
40-dim MFCC	14.9	17.8
log-Mel fbank*	15.1	18.5
Time-domain setup[1]	14.4	17.4
Time-domain setup[2]	15.2	18.2
Proposed frequency-domain setup*	**14.3**	**17.0**
CLP*	14.9	17.6

*: CNN layer added after log filter banks.

Finally, we evaluate our proposed frequency-domain setup on various databases, namely TedLium [**?**], AMI IHM and SDM [**?**], Wall Street Journal [**?**] and Librispeech [**?**]. The results are shown in Table 3.10. The amount of training data for filter learning varies from 80-1000 hours across these tasks. The baselines are the state-of-the-art TDNN models trained on standard 40-dimension MFCC features. We use 100 filters in 8kHz tasks and 200 filters for the 16kHz tasks. The results on Librispeech are obtained by rescoring with a $4-$gram language model. We use the same CNN layer as described in Subsection 3.2.2 in all the experiments. An average relative improvement of 1 to 7% was observed over the conventional state-of-the-art MFCC based models.

Table 3.10: Performance of the proposed frequency-domain setup on various databases

Database	Test set	Baseline	Proposed setup
Switchboard	eval2000	14.9	14.3
	rt03	17.8	17.0
Wall Street Journal	eval92	2.6	2.4
	dev93	4.7	4.6
TED-LIUM	dev	8.3	7.8
	test	8.8	8.4
AMI-IHM	eval	20.3	19.9
	dev	20.4	20.1
AMI-SDM	dev	37.3	36.3
	eval	40.9	40.2
Librispeech	dev-other	10.6	9.7
	test-other	10.9	10.2

3.3 Multiscale feature learning from frequency domain

For speech recognition, features, that can be extracted from both narrowband and wideband spectrograms, are important for good accuracy. ASR systems usually use single fixed frame lengths and the classification models are expected to learn key patterns with the objective of minimizing WER. However, it is not easy to detect important patterns from fixed length frames as some speech attributes require longer window lengths for a better estimate and some require smaller window lengths.

a narrowband spectrogram [**?**] is computed over a long segment of the time signal to capture the rapid change in amplitude at the time of vocal fold closure and used to estimate the fundamental frequency and intonation. On the other hand, a wideband spectrogram [**?**] is calculated over a short time window and captures rapid changes in amplitude and the timing of changes in vocal tract resonances more reliably. It is created with more coarse-grained frequency analysis to model the broad spectrum envelope peaks that correspond to vocal tract formants.

With the advent of DNNs in recent years, different parts of the ASR systems such as acoustic and language modeling have greatly improved. However, most commonly used ASR systems still use conventional hand-crafted features such

as MFCC [**?**], PLP [**?**] and log-Mel filter banks.

Data-driven feature extraction methods aim to jointly learn feature extraction and acoustic modeling. However, these methods have not significantly outperformed conventional features on LVCSR tasks [**?**, **?**, **?**].

Recent work on frequency-domain feature learning [**?**] shows promising results in the data-driven feature learning regime with low latency. Filter banks trained in this setup are constrained by their window size to a single scale. In this section, we explore jointly learning filter banks at multiple scales. Multiscale time-domain convolution filter learning has been investigated in different tasks such as ASR [**?**], image classification [**?**] and gesture detection [**?**]. The multiscale convolution systems split the spectrum into different filter banks using different strides and window sizes.

In this chapter, we propose a new multiscale feature learning using frequency domain setup by jointly learning multiple filter banks on signal frames with different lengths. We also find optimum sub-bands from different scales and show that combining these sub-bands results in the same improvement while requiring less computation and outperforms features learned on the single scale.

Also, we investigate the effect of increasing temporal resolution in multiscale filter bank learning by increasing the input frame rate while having a fixed output frame rate. The computational cost is optimized using a new

time delay network architecture with higher down-sampling in the intermediate layers of the network. In addition, we show that a higher input frame rate in multiscale feature learning outperforms a normal frame rate.

3.3.1 Prior work

Authors in [**?**, **?**] proposed learning features from raw waveforms using multiscale setups. In [**?**], the effect of temporal and frequency resolution by changing the frame rate and number of filters were presented in a connectionist temporal classification (CTC) framework and have shown significant improvements. In [**?**], authors investigated feature learning from the raw waveforms extracted at multiple downsampled rates in the context of the emotion recognition task.

Chan and Peddinti [**?**, **?**, **?**], whose work is based on hand-designed features, studied various wavelet based features to overcome the limitations of the Fourier transform in terms of temporal and frequency resolution. All of them reported improvements but their work was on a relatively clean dataset (i.e., TIMIT as a phone recognition task). Therefore, there is still a need to investigate effectiveness of these kinds of features on noisy datasets and LVCSR tasks.

In contrast to the studies of [**?**, **?**, **?**], authors in [**?**] have explored extracting features from longer time scales compared to the usual 10-30 ms windows

where LPC was used in a frequency domain to model temporal peaks.

3.3.2 Proposed method

In this section, we explain our multiscale feature extraction method from speech spectrograms. Figure 3.14 shows the process of feature learning for an LVCSR task from a speech spectrogram using our proposed 2-scale frequency domain setup. The raw waveform is passed through a pre-processing layer which performs pre-emphasis and DC removal on windowed speech segments with a 10 ms shift. The Fourier transform is applied on windowed segments to obtain a spectrogram. The "Povey" window, defined in Equation 3.4, is an extension to the Hamming window function, that goes to zero smoothly at the edges, to avoid large side-lobes. The input frame length can be adjusted by changing the power (ρ) in the "Povey" window. Larger power results in smaller effective window size and larger mainlobe.

The actual input window size used in both scales is 30 ms and the effective input window sizes used for learning filter banks are 30 ms (wide window) and 15 ms frame length (narrow window) modeled using ρ_1 and ρ_2 value of 4.0 and 0.85, respectively (i.e., Figure 3.14). We compared this setup with using actual window sizes 30 and 15 ms with default ρ value of 0.85 (i.e., rows 1 and 2 in Table 3.15). Figure 3.13 demonstrates the effect of different ρ values on the main lobe width and side lobe roll-off rate. The width of the main lobe in 30

ms window with ρ value of 6.0 is 280 Hz, while this width for 15 ms window and default ρ value of 0.85 is 120 Hz. The ρ value of 4.0 with window size 30ms gives closer main lobe width as 15 ms window with default ρ of 0.85. In spite of this fact, the results show the same performance in both setups. Also, we experimented using multiscale dbl setup with 3 different ρ values for smaller scale, 2.1, 4.0 and 6.0 with actual window size 30 ms. The results show ρ of 4.0 gives best performance.

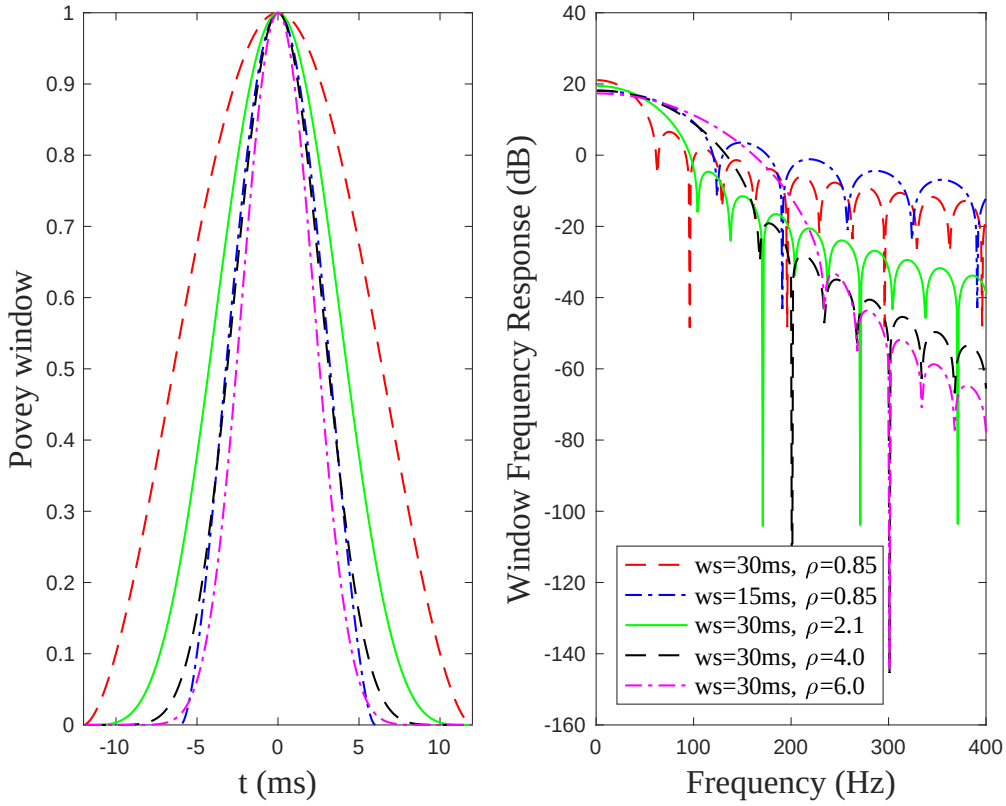

Figure 3.13: Effect of ρ on the main lobe width of "Povey" widnow

Then, L2-normalization is applied on the Fourier transform output followed by a normalization block. The normalization block consists of applying log, batch-normalization, mean-variance learning and normalization, and exponent, which is described in detail in [**?**].

After normalization, there is the main filter bank layer which is basically a linear layer. Each row in the weight matrix of this linear layer can be interpreted as a filter. The weights of the matrix were clipped to be between 0 and 1. Log compression is applied on this filter bank output to reduce the dynamic range which is a common practice in DNN acoustic modeling. The normalization block together with a filter bank and log compression is shown in Figure 3.14(a). Finally, these compressed features are passed to a 2D-CNN layer with 32 kernels. Convolution is performed with 3×3 filters and a stride of 2. For more detailed analysis, please refer to [**?**].

$$\mathbf{w}(n) = [0.5(1 - \cos(\frac{2n\pi}{N}))]^{\rho} \tag{3.4}$$

The full-band multiscale setup, explained in Figure 3.14(b), uses a full spectrum from both narrow and wide input frames, and two separate sets of filter banks with N_1 and N_2 filters trained separately on a normalized wideband and narrowband spectrum. Based on results from the setup shown in Fig-

ure 3.14(b), we also experimented with constraining the filter banks to a subset of frequencies which we call a sub-band multiscale model and Figure 3.14(c),(d) are the corresponding setups. With the setup in Figure 3.14(c), separate filter banks are learned for each window length. With the setup in Figure 3.14(d), sub-bands from different scales are appended and a common filter bank is learned for both window lengths.

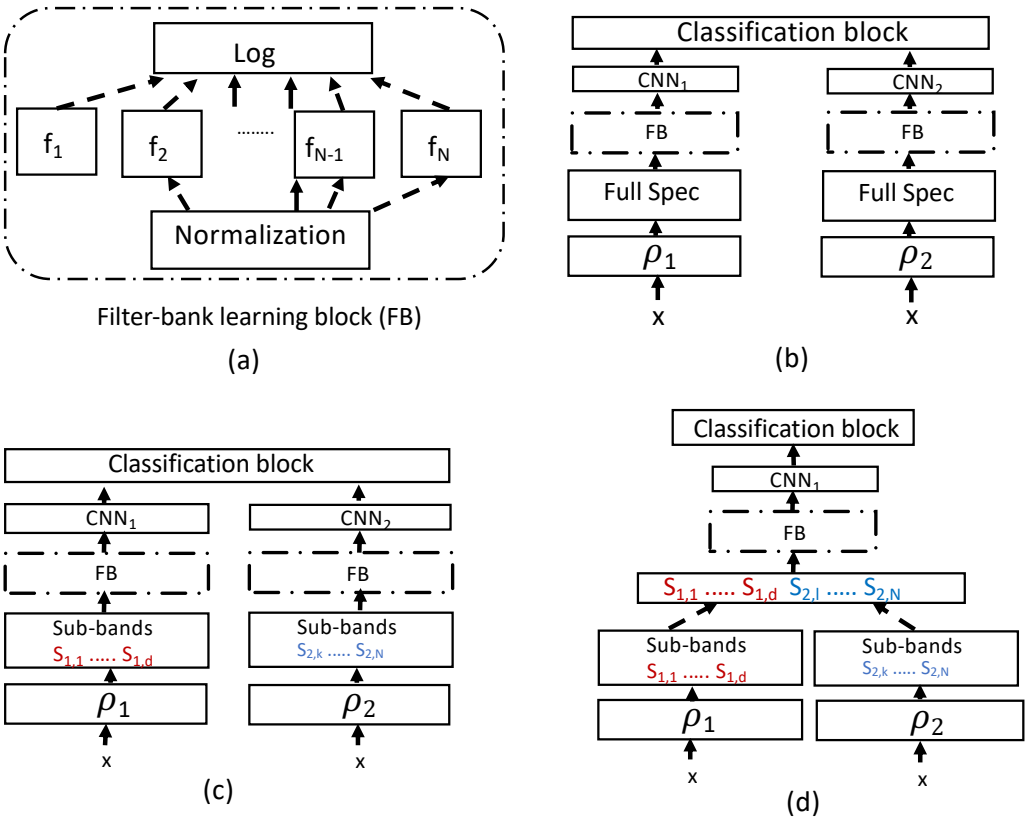

Figure 3.14: Multiscale frequency-domain feature extraction setup (a) Filter bank (i.e., FB) learning block, (b) Multiscale setup with separate FB and full spectrogram for each window length, (c) Multiscale setup with separate FB and only a few sub-bands for each window length, and (d) Multiscale setup with only a few sub-bands for each window length and common FB

We used a basic TDNN layer [**?**] for a classification block. To model a larger

temporal resolution, we increased the input frame rate to 200 Hz by reducing the frame shift to 5 ms, while the output frame rate is still 33 Hz. To reduce the computational cost of increasing the input frame rate, the explicit down-sampling is applied by multiplying the time-offset in the TDNN layer by a factor of 2 and the effective context is the same as the baseline model with a 10 ms frame-shift. The context in a normal and double frame-rate network (i.e., dbl) is shown in Table 3.11.

Table 3.11: Context specification in normal and double frame rate network

Layer	Normal input context	Dbl input context
1	[-1,1]	[-2,2]
2	conv([-1,1],[-1,1],2,32)	conv([-1,1],[-1,1],2,32)
3*	{-1,0,1}	{-2,0,2}
4	{-1,0,1}	{-2,0,2}
5	{-1,0,1}	{-2,0,2}
6	{-3,0,3}	{-6,0,6}
7	{-3,0,3}	{-6,0,6}
8	{-3,0,3}	{-6,0,6}
9	{-3,0,3}	{-6,0,6}
10	{0}	{0}

*: The i-vector appended at this layer for speaker adaptation.

We investigate the performance of our proposed multiscale feature learning using single and double input frame rates and compare the results with using multiscale MFCC setups. In addition, based on the filters learned for each window scale, we investigated constraining the model to learn a specific set of filters for each scale.

We evaluated our approach on various clean databases, namely Switch-

board [**?**], TedLium [**?**] and noisy datasets, AMI-IHM and AMI-SDM [**?**]. We performed our initial experiments on Switchboard and AMI-SDM to study the multiscale and multi-frame rate setup.

3.3.3 Effect of scale and frame rate with MFCC

In this section, we investigate extracting multiple sets of MFCC features using signal frames with different lengths. Window frames of 17 and 30 ms are used to obtain 40-dimension MFCC features (17 ms frame is selected to have the same FFT size for an 8kHz dataset). These features are concatenated and used as input to the network in a multiscale setup. We also investigate the effect of increasing the input frame rate from 100 to 200 Hz by reducing the frame shift. The TDNN context proposed in Table 3.11 is used in normal and double input frame rate setup. Table 3.12 shows the results using MFCC features extracted using 2 scales and 2 input frame rates. As shown, increasing the input frame rate degrades the performance, and only a slight improvement is observed with multiscale MFCC.

Table 3.12: Performance of MFCC features using different scales and input frame rates

		100Hz		200Hz	
Database	Test set	30 ms	$(17, 30)$	30 ms	$(17, 30)$
Switchboard	eval2000	14.8	14.7	15.2	14.8
	rt03	18.1	17.7	18.1	18.0

We also experimented with wavelet features as they provide the flexibility

on temporal and frequency resolution. To extract wavelet features, we experimented with different discrete wavelet functions on a frame length of 24 ms. Daubechies wavelet with 12-taps worked best among all. Experiments with both full tree split and Mel-scale based split did not give us any significant results compared to MFCC. We obtained a 19.4% WER on the eval2000 test set with a full tree split which is 4.6% absolute worse than MFCC.

3.3.4 Multiscale feature learning

Effect of number of scales

In the multiscale time convolution setup in [**?**], increasing the number of scales to 3 shows considerable improvement. To find the optimum number of scales, we experimented with learning features using a single scale, 2-scales, and 3-scales. For each scale, a separate set of filters are learned and the outputs of filter banks for all scales are concatenated at the end of the feature extraction block which will be used for classification (i.e., Figure 3.14(b)). We experimented with three scales with actual window sizes (15, 20, 30)ms and (15, 30, 60)ms and ρ value of 0.85. Also, we compared the results with single scale setup with window sizes 15, 30 and 60 ms. From Table 3.13, it is clear that using 2-scales is much better than a single scale and a slight degradation is observed with three scales on Switchboard datasets. On the other hand, 3-

scale setup gives some improvement on AMI-SDM. We decided to use only two scales for further experiments.

Table 3.13: Effect of number of scales in the multiscale dbl setup

Database	Test set	15	30	60	(15, 30)	(15, 20, 30)	(15, 30, 60)
Switchboard	eval2000	14.0	14.3	14.9	14.0	14.2	14.1
	rt03	16.6	17.0	17.0	16.2	16.5	16.4
AMI-SDM	dev	35.6	35.3	36.4	34.8	34.8	34.5
	eval	39.4	38.9	40.1	38.8	38.6	38.1

Figure 3.15 shows the filters learned on each window length. It can be observed in the case of Switchboard (first and third rows) that filters learned on a 30 ms (wide) window length mostly focused on high frequencies $[1 - 4]$kHz whereas filters learned on a 10 ms (narrow) window length are on low frequencies up to 1.5kHz. Surprisingly, filters learned on 60 ms are mainly on a low frequency sub-band $[0 - 1.5]$kHz.

Filters learned on a 30 ms window length on AMI-SDM (second row) show similar behavior to a 60 ms Switchboard and 80% of filters learned on a low-frequency sub-band $[0 - 1.5]$kHz. However, the filters learned on a wide window AMI-SDM don't seem to be specific to low-frequency sub-bands, and still, 20% of filters are learned on high frequency. The reason could be because this dataset is reverberant and noisy which makes the signal non-stationary within a window of 30 ms. We also observe similar behavior for filter banks learned on small 15 ms window lengths for AMI-SDM and 80% of filters learned on a high-frequency sub-band $[1.5 - 8]$kHz.

Based on this observation, we experimented with explicit constraints on the sub-bands that can be used in learning filters on each window length.

Frequency sub-band combination at different scales

As shown in Figure 3.15, there is a frequency overlap $[1 - 1.5]\text{kHz}$ between the filters learned from both window lengths. We hypothesize that constraining the model to learn dedicated low and high-frequency filters in each set will allow the model to perform better which also has the advantage of less computation.

Table 3.14 shows the results of sub-band combinations for different scales. The effective small and large window lengths are 15 and 30 ms, respectively, and their frequency sub-bands are denoted by an ordered pair ($[f_{s_1}, f_{s_2}]$,$[f_{l_1}, f_{l_2}]$). The Nyquist frequencies, denoted by f_{nq}, for Switchboard and AMI-SDM are 4kHz and 8kHz, respectively. The 200 Hz input frame-rate setup is used in all the experiments listed in this table.

Experiments in rows 1, 2 and 3 in Table 3.14 use 100 filters in the filter bank layer (refer to Figure 3.14(a)). We have observed a slight WER degradation when we used only 50 filters, as shown in row 4. The experiment in rows 5 and 6 use the setup proposed in Figure 3.14(d) with 100 and 200 filters in the filter bank layer and 32 and 64 kernels used in the CNN layer. As can be seen, increasing the number of filters does not improve the results.

Table 3.14 shows that combining a high-frequency sub-band from the smaller scale (i.e., wide spectrogram) and low-frequency sub-band from the larger scale (i.e., narrow spectrogram) results in the same performance improvements as combining full-bands. Also, comparing results in rows 3 and 5 shows that learning separate filter banks on different scales produces better results.

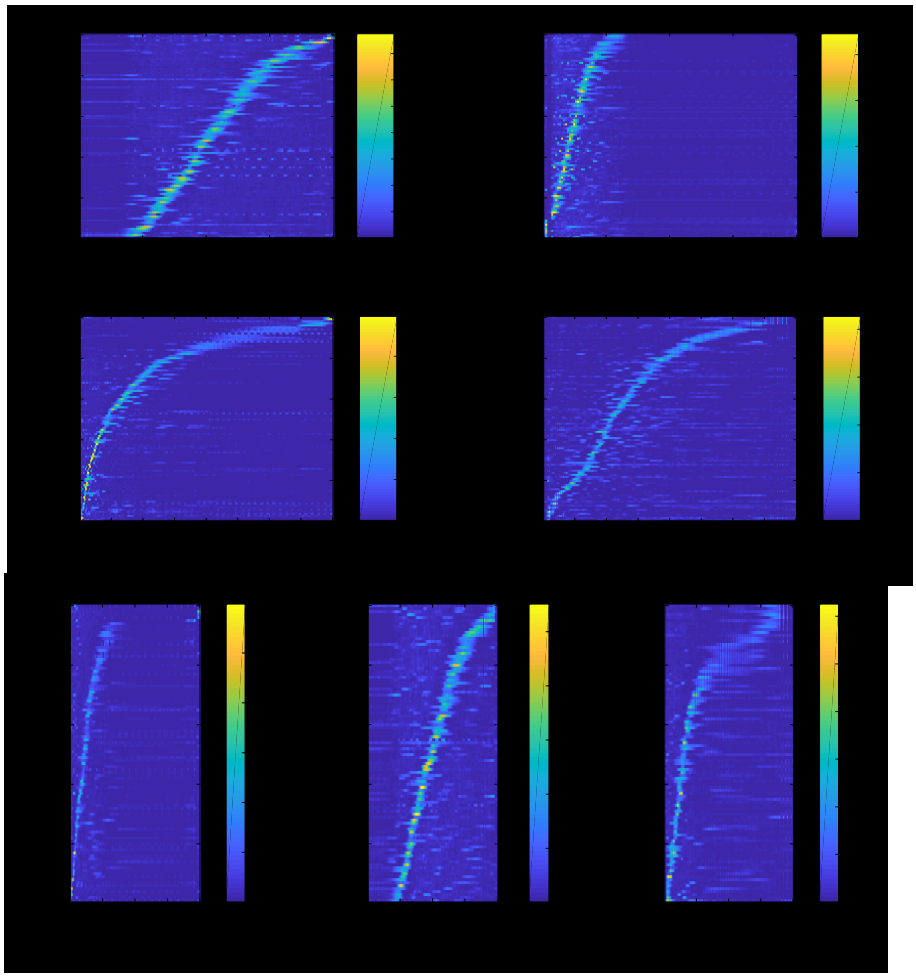

Figure 3.15: Demonstration of learned filters on Switchboard, AMI-SDM. First row and third row are on Switchboard using 2-scales and 3-scales, respectively. Second row is on AMI-SDM using 2-scales.

Table 3.14: Effect of sub-band combination from different window scales. ($[f_{s_1}, f_{s_2}]$,$[f_{l_1}, f_{l_2}]$) is an ordered sub-band pair selected from 15 and 30 ms, respectively.

Database		Switchboard		AMI-SDM	
Test set	(N_1, N_2)	eval2000	rt03	dev	eval
$([0 - f_{nq}]$,$[0 - f_{nq}])_{(b)}$	$(100, 100)$	14.0	16.2	**34.8**	**38.8**
$([0 - 1.5]$,$[1 - f_{nq}])_{(c)}$	$(100, 100)$	14.1	16.6	35.7	39.4
$([1 - f_{nq}], [0 - 1.5])_{(c)}$	$(100, 100)$	13.9	16.3	34.9	**38.8**
	$(50, 50)$	14.0	16.4	35.0	39.0
$([1 - f_{nq}], [0 - 1.5])_{(d)}$	$(200)^*$	14.1	16.3	35.2	39.2
	$(100)^*$	14.0	16.4	35.4	39.3

*: Single filter bank layer used on top of combined sub-bands.

Effect of input frame rate

Table 3.15 shows the results of a variety of clean and noisy datasets for normal and double frame rate networks (i.e., explained in Subsection 3.3.2) using the setup shown in Figure 3.14(b). We ensured that the input context used in the classification block for both frame rates is the same by doubling the time-offset in each TDNN layer (i.e., shown in Table 3.11). We can observe significant improvements with a double scale input compared to a single scale. Also, the double frame rate is beneficial in multiscale feature learning except for the TedLium database.

Table 3.15: Performance of the proposed frequency-domain setup on various databases using different scales and input frame rates.

		100Hz		200Hz	
Database	Test set	30 ms	(15, 30) ms	30 ms	(15, 30) ms
Switchboard	eval2000	14.3	14.4	14.1	14 (13.9*)
	rt03	17.0	16.6	16.5	16.2 (16.3*)
AMI-IHM	dev	20.1	20.1	19.8	19.7
	eval	19.9	19.9	19.6	19.4
AMI-SDM	dev	36.3	36.0	35.3	34.8
	eval	40.2	39.9	38.9	38.8
TedLium	dev	7.8	7.9	7.9	7.8
	test	8.4	8.3	8.2	8.4

*: Multiscale with separate input frames 15, 30 ms.

Chapter 4

Joint feature extraction application in emotion identification

Speech-based emotion classification has been gaining popularity for the development of emotionally sensitive human machine interaction (HMI) systems. In the evolving setups of intelligent commercial dialogue systems and smart call centers, emotion information obtained from speech can be used as metadata to understand speakers' psychology and response. There are a number of modalities that can be used to determine a human's emotions, which includes facial expression, body movement, physiological measures such as galvanic skin response and voice or speech. Automatic emotion classification sys-

tems have been developed using all of these modalities. However, designing a speech-based emotion detection system has some particular importance. Humans express emotional state related information through numerous subtle ways including low-level acoustic descriptors like pitch, voicing probability, energy, zero-crossing rate, Mel filter bank features, formant locations, bandwidths, harmonics-to-noise ratio, jitter, etc., that may or may not be directly represented by standard features such as Mel filter bank or formant locations, pitch, voicing, etc. Researchers have been primarily focused on deriving useful statistical feature sets from low-level acoustic cues as well as on developing an efficient machine learning based modeling strategy to learn the emotion dependent temporal and contextual variations of speech.

4.1 Prior work

The primary challenge in this chapter is that the feature set needs to be robust enough to capture the emotional content from various styles of speaking and complicated emotional states like happy and excited, angry and frustrated, etc. Machine learning based models have also been used to derive high-level features to represent the whole utterance from low-level acoustic features. Recently, deep learning approaches are becoming popular for modeling emotion-specific information from speech signals [**?**] [**?**] [**?**] [**?**] [**?**]. However,

there is a recent trend in deep speech-based system design which attempts to derive features of the input signal directly from raw, unprocessed speech waveforms excluding the necessity of hard-coded feature extraction outside the DNN. Such approaches have shown appreciable reliability and state-of-the-art performance in speech recognition tasks [**?**,**?**,**?**]. In the domain of a paralinguistic, Trigeorgis et al., 2016, used raw waveforms for speech emotion dimensional rating in a deep CNN framework [**?**]. Motivated by such success of raw waveforms, in this study we propose using raw waveform front-end layers to learn emotion-specific cues within the network and design an end to end DNN setup for categorical emotion identification tasks.

The raw waveform front end [**?**] used in this study attempts to learn a specific set of filters that are jointly optimized with the rest of the network and the filter bank is learned to optimize the emotion identification objective.

A challenging issue in an emotion identification task is effective modeling of the long temporal context. This is because emotion-specific information lies in the long span of time to a great extent. We explore multiple DNN architectures to appropriately model such long-term dependencies of emotion cues and provide a comparative analysis. We use temporal convolution in the form of TDNN layers [**?**] and unidirectional recurrent projected long short-term memory (LSTM) [**?**] layers individually with the raw waveform front end. We also experiment with an interleaving TDNN with unidirectional LSTM (TDNN-

LSTM) setup and time restricted attention mechanisms [**?**] which enables the DNN to be more attentive to emotionally sensitive portions of speech. We use such time restricted attention layers with both an LSTM and TDNN-LSTM setup, and we observe that attention improves the accuracy significantly in both of these setups as well as helps reduce confusion among individual categories.

We experiment with statistics extraction layers which were previously used with the xvector setup of the speaker and language identification [**?**, **?**, **?**]. Also, we experiment with all these temporal modeling setups individually with the frequency and time domain raw waveform front end and observe the best results using a TDNN-LSTM-attention setup with a time domain raw waveform front end.

All of the results have been reported on the categorical emotion identification problem of the interactive emotional motion capture (IEMOCAP) database [**?**]. We design a baseline DNN setup with TDNN layers using a high-resolution 23-dimensional MFCC. Experimental results confirm the improvement obtained from the proposed raw waveform based DNN setups which learn features within the network over the MFCC based DNN setup where hardcoded features were used. We also experiment separately with time and frequency domain data-driven filter learning approaches in the raw waveform setup. In addition, we include a few intermediate experimental comparisons regarding DNN training

time and decode time dependencies on seeing more or less context. Such dependencies play a very critical role for an emotion identification task. In our best DNN setup, we observe 8.31% improvement regarding weighted accuracy (WA) and 4.37% improvement in terms of unweighted accuracy (UA) over a 257-dimensional magnitude FFT vectors based DNN setup reported in [**?**].

4.2 Feature extraction in emotion identification

Emotions influence both the voice characteristics as well as linguistic content of speech. Most previous research studies in speech emotion recognition have been focused on the search for speech features that are indicative of different emotions [**?**] and used suprasegmental/prosodic features as their acoustic cues. Pitch, energy and rhythm are prosodic features and suprasegmental features are computed on a whole sentence or emotion instance. These cues are important indicators of emotional states [**?**] and used in many emotion recognition systems [**?**,**?**,**?**]. The spectral information of speech is another important feature for representing emotional states, which has been found to be useful for emotion classification [**?**,**?**].

It is still unclear which features are more informative about emotions. The traditional approach is to extract a large number of statistical features, de-

scribed in Table 4.1, and perform classification using standard machine learning algorithms. Some acoustic features are usually extracted from short frames of 20 to 50 ms, and are called low-level descriptors (LLD). Then, various statistical aggregation functions such as mean, max and variance are applied to LLDs over the whole utterance to extract long utterance level feature vectors. Some of these features are mentioned in Table 4.1. The high-level statistical functions describe the temporal variations of LLDs during the utterance, and the assumption is that emotional content correlates with temporal variations, rather than short-term LLDs.

Table 4.1: Low-level descriptors (LLDs) and high-level statistical functions (HSFs) for speech emotion recognition

LLDs	pitch, voicing probability, energy, zero-crossing, Mel filter banks, MFCCs, formant locations/bandwidths, harmonics-to-noise ratio, jitter
HSF	mean, variance, min, max, range, median, quartiles, higher order moments (skewness, kurtosis), linear regression coefficients.

Here, we describe the baseline MFCC based DNN setup and two raw waveform feature extraction front end setups (Sections 3.1 and 3.2) for the emotion identification task. Table 4.2 presents the results obtained from all three experiments. The neural network setup used in all these experiments is explained in detail in Section 4.4.

Most speech systems use short-term hand-crafted spectral and cepstral features based on fixed filters, such as MFCC or Mel filter banks. In the first

experiment, we use 23-dimensional MFCC features as input to DNN. The determination of parameters from the raw signal compared with the classical representations in the time or frequency domain seems to have more advantages.

However, using a fixed filter may not be the most appropriate for a final objective of minimizing emotional states classification errors. In the next experiment, row 2 of Table 4.2, we use a direct-from-signal setup described in [**?**] which attempts to learn filters within the DNN. We refer to this as time domain raw waveform front end in the rest of the description. The input frames are 40 ms long segments of raw waveform signal with 10 ms overlap. This raw waveform front end has a $1-d$ time convolution layer, which operates on a 40 ms raw signal with a step size of 1.25 ms and the filter outputs are aggregated using two trainable NiN nonlinearity layers introduced in [**?**]. We also used a setup proposed in [**?**], where the signal is first transformed into the frequency domain and a trainable filter bank layer, which is modeled using linear transformation, is jointly trained with the rest of the network.

We observe that using a direct-from-signal setup can improve the performance significantly compared to the baseline MFCC. Also, we observe that the results of the time domain raw waveform front end are better than learning features from the frequency domain. We need to do more experiments using complex domain filter learning to model phase information, which can be use-

ful in learning emotional states. In all results reported hereafter, we use a time-domain feature extraction block.

Table 4.2: Effect of different feature extraction methods

Feature extraction method	WA
MFCC	59.9
Time-domain	**65.5**
Frequency-domain	63.4

Figure 4.1 shows the learned filter banks in ASR and age identification tasks using proposed time-domain and frequency domain setups. As shown, filter banks are wider, especially in the high-frequency region in the time-domain setup. Also, there are more filter banks in the frequency sub-band $[0, 200]$ Hz in the frequency-domain setup.

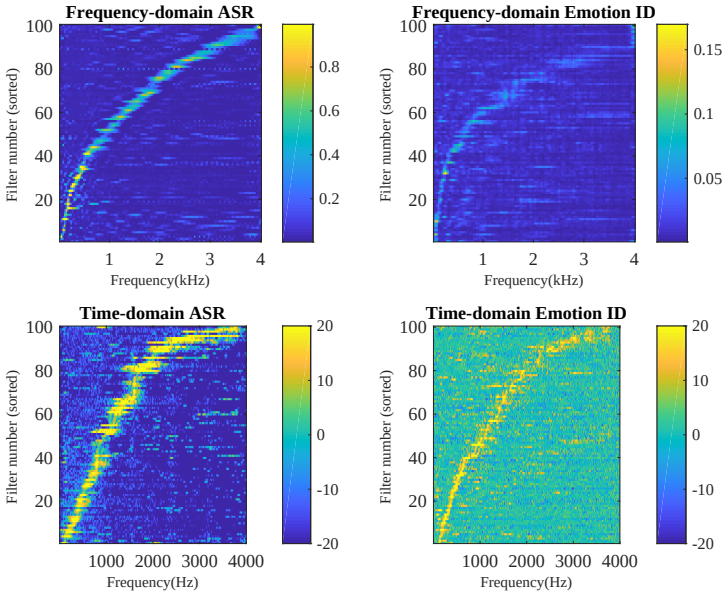

Figure 4.1: Learned filter banks for ASR and age identification tasks using proposed time-domain and frequency-domain setups

4.3 Results

This section describes the experimental details and results. All of our experiments are done using the Kaldi toolkit [**?**]. All of DNN based emotion identification setups described in this section have two common structures as shown in Figure 4.2 (a) and (b). The initial block contains raw waveform front-end layers as described in Section 4.2. The temporal modeling layers are either TDNN or LSTM or a combination of the two along with an attention layer. We use a statistics pooling layer before a softmax layer as in Figure 4.2(a) to get segment-level emotion class output. We also use an attention layer in the temporal modeling block. We do post-processing as an averaging of posteriors over frames outside the network to get the segment-level emotion class output as in Figure 4.2(b).

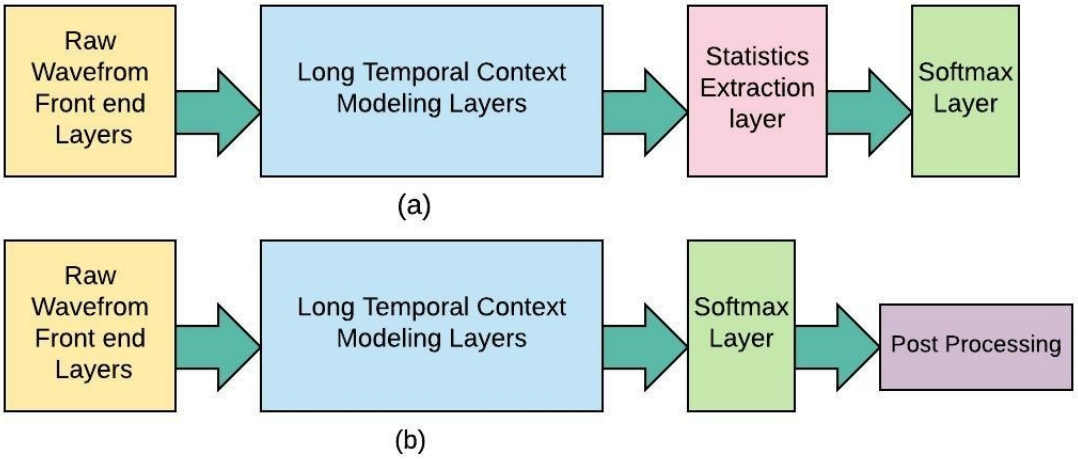

Figure 4.2: Layout of the proposed end to end DNNs for an emotion identification task

We have used four emotion categories (neutral, angry, sad and happy) from the interactive emotional dyadic motion capture (IEMOCAP) database [**?**]. The database consists of about 12 hours of audiovisual data (speech, video, and facial motion capture) from five mixed gender pairs of male and female actors, in two recording scenarios: scripted and improvised speech. It is organized in five sessions, four of which are used for training and one is used for testing. Each wave file has a segment-level emotion category label annotated by human annotators.

The performance of the emotion identification DNNs are reported using two parameters, WA which is the overall classification accuracy and UA which is the average recall over the emotion categories.

To increase the amount of data in the training set, we perform data augmen-

tation utilizing amplitude and speed perturbation. For each speech signal, five different amplitude modulated versions are created initially. Then, speed perturbation [**?**] is applied on the amplitude modulated signals with speed factors of 0.9, 1.0, and 1.1. The effect of data perturbation on the emotion identification task can be seen in Table 4.3.

Table 4.3: Effect of data perturbation on emotion identification in our best setup without tuning decode time parameters

Perturbation	WA	UA
No	60.27	48.84
Yes	66.07	57.215

4.4 Modeling long temporal context

The DNN is trained to classify different emotional states. Training examples consist of chunks of speech features ranging from 0.6 to 35 seconds with a single emotional state label. We use the softmax layer at the end of the network to give the network freedom to model any distribution over output and each emotional state modeled as a separate output class.

One of the main issues in predicting emotional state is that the emotion cues cannot be easily estimated over a small span of time and we need to preserve the temporal context or use long examples to estimate the emotional state of speakers. In this section, we compare different approaches in modeling temporal context. We use TDNN architecture which models long-term temporal de-

pendencies in models described in this section. One disadvantage of temporal modeling using TDNN is the linear increase in parameters and computation with an increase in the temporal context; a non-uniform sub-sampling method helps to mitigate this issue. We also use only LSTM layers with and without attention for temporal modeling in a recurrent way.

Table 4.4: Effect of long temporal modeling layers

Temporal Modeling	WA	UA
TDNN-Statistics Pooling	65.5	55.3
TDNN-LSTM	59.5	56.4
TDNN-LSTM-Attention	66.3	60.3
LSTM	59.9	53.7
LSTM-Attention	63.4	56.2

Statistic pooling layer

We use the TDNN layer as a temporal convolution in this setup, and the context used in the TDNN layer is similar to the setup in [**?**]. The statistic pooling layer [**?**, **?**] is used in this setup, which aggregates all available frame level inputs for the intermediate layer in the network and outputs their mean and standard deviation. This layer operates on the entire segment and the mean and standard deviation are concatenated together and passed through a feedforward layer, and finally, a softmax layer is applied on them. Despite the TDNN-LSTM setup, we use a single emotional state label for the entire example in this setup. One disadvantage of this setup is that the speech and

non-speech frame weights are similar in computing the mean and standard deviation in the statistic pooling layer and the error is back-propagated uniformly from this layer across all time frames. This requires the use of energy based SAD to filter out non-speech frames. The results of not filtering the non-speech frames is a large degradation in this setup. This issue is solved using a TDNN+LSTM+Attention setup, in which the non-speech frames are not removed

TDNN-LSTM

In this setup, we use temporal convolution in the form of TDNN layers along with LSTM layers. We use interleaving of temporal convolution with unidirectional LSTM, which has been reported to outperform bidirectional LSTM [**?**]. Also, we use per-frame objective, where all frames have the same emotional state label for each utterance. We use a higher frame rate at lower layers of LSTM, and a TDNN layer in the network and layer frame rate decreased with layer depth. The layer-wise context of a temporal modeling block is similar to $config\ 1$ of Table 4.5 except that this setup does not have an attention layer. The results of this setup are shown in row 2 of Table 4.4.

TDNN-LSTM with time-restricted attention

We exploit a time-restricted self-attention mechanism, where the input and output sequence lengths are the same, and it attends at a particular frame with a limited number of frames to its left and right. A time-restricted attention layer [**?**] is used as the last layer along with TDNN and a unidirectional LSTM layer. The architecture of a TDNN-LSTM-Attention setup contains interleaving TDNNs and LSTMs with an attention layer after the last LSTM layer. The layer-wise context of the temporal modeling block of this setup is shown in $config\ 1$ of Table 4.5. The dimensions of projection and the recurrence are one quarter the cell dimension. We found that a cell dimension of 128 is optimal for the current emotion identification task and with the recurrence of dimension 32 and the dimension 64's output of LSTM. The LSTMs operate with a recurrence that spans 3 time steps. The attention layer used has 12 heads, a context of $[-5, 2]$, a key-dimension of 40 and a value dimension of 60. In this setup, we use a per frame dropout using the dropout schedule method described in [**?**] where the entire vector is forced to be zero or one. The dropout schedule is expressed as a piecewise linear function on the interval $[0, 1]$, where $f(0)$ gives the dropout proportion at the start of training and $f(1)$ gives the dropout proportion after seeing all the data. A dropout schedule of the form $0, 0@0.20, p@0.5, 0@0.75, 0$ is used in this setup, where p is 0.3 in the results reported here. Thus, the dropout probability is 0 at $f(0)$, 0 at $f(0.2)$, 0.3 at $f(0.5)$, 0 at $f(0.75)$ and 0 at

$f(1)$. In this setup, we average frame posteriors outside the network to get a segment level aggregate from the frame level posteriors. The performance of this setup is shown in row 3 of Table 4.4. We add some extra left context at the time of decoding which provides flexibility to the network regarding the number of frames it sees in addition to what is provided during training. We evaluate the model several times to tune this length of decode time context. Also, we observe improvement by using a longer training chunk by using fixed length examples during training. Details of fixed length versus variable length training example experiments are reported in Section 4.5.

Table 4.5: Layer wise context of a temporal modeling block for TDNN-LSTMP-Attention setup

	Config1		Config2	
Layer	Context	Layer-type	Context	Layer-type
1	[-1, 0, 1]	TDNN	[-1,0,1,2]	TDNN
2	[0]	LSTM[1]	[-3,0,3,6]	TDNN
3	[-3, 0, 3]	TDNN	[0]	LSTM[1]
4	[-3, 0, 3]	TDNN	[-6,0,6,12]	TDNN
5	[0]	LSTM[1]	[0]	LSTM[2]
6	[-3, 0, 3]	TDNN	[-12,0,12, 24]	TDNN
7	[-3, 0, 3]	TDNN	[-5,2]	Attention
8	[0]	LSTM[1]	[-12,0,12]	TDNN
9	[-5, 2]	Attention		

LSTM[1]: delay time=-3
LSTM[2]: delay time=-6

LSTM with time restricted attention

We use three unidirectional LSTM layers in this setup with a cell dimension of 128 and a recurrent and nonrecurrent projection dimension of 32. In prior work on a language identification task [**?**], it was suggested to perform pooling over a time recurrent layer to reduce redundancy. We experiment using a max pooling layer after the last LSTM layer and observe improvement in the accuracy. A comparison of results of LSTM with and without time pooling is shown in Table 4.6. We also add time restricted attention to this LSTM only setup. We observe significant improvement using a time restricted attention layer as a final layer in the LSTM only setup as shown in rows 4 and 5 of Table 4.4. Also, we use a similar dropout schedule as in Section 4.4.

Table 4.6: Effect of time pooling in the LSTM setup

Temporal Modeling	WA	UA
LSTM	54.5	48.9
LSTM with max pooling	59.9	53.7

4.5 Variable-length vs. fixed length training

The training and test utterances used in our emotion identification setup have variable lengths in the range of 0.6 to 10 seconds, and we need high accu-

racy on short segments during test time. It is challenging to get utterance-level representation, which is normalized over different lengths. Minimizing network sensitivity to speech duration is important. One solution is to train the network on chunks of different durations. In this section, we investigate the effect of training with variable length chunks, where the output is generated from the entire utterance (if it is shorter than 6 seconds). We compare the results with dividing the utterance into fixed-length chunks and randomize and use them for training. Table 4.7 presents the results using models trained on the fixed and variable-length examples. The model configurations used in all experiments are described in Table 4.5.

In the experiment in row 1, the length of example chunks varied from 1 to 6 seconds, and the entire utterance is used as an example if it is shorter than 6 seconds. The size of mini-batches is a function of example length (e.g., mini-batch sizes used for examples with lengths of 100 and 200 are 128 and 64, respectively) and the total number of frames is almost equal in different mini-batches. The network configuration, $Config2$, is used in this experiment that is described in Table 4.5. The use of future context information in unidirectional LSTM is accomplished using delayed prediction of the output label. We use delay time 3 and 6 for LSTM layers in our experiments. For the LSTM layer, we use an effective temporal context and decay time, and it helps to generalize unseen sequence length which is equivalent to a maximum number of frames

that are remembered via the LSTM layer. We use a decay time of 100 frames in this experiment, and the error is back-propagated through 100 effective frames. In this experiment, we use a longer decay time to remember longer frames for extended example chunks. The network learns emotional states on longer chunks more efficiently. The frame-level cross entropy objectives are improved with a faster rate using variable length utterances, and the model converges within 30 epochs.

In the experiments in row 2, we used a fixed length chunk with 50 frames, and the network trained for the same number of epochs. As expected, it is harder to learn emotional states over 0.5 seconds. The network converges slower, and it needs to train for a longer time. The interesting point about this setup is higher randomization. Long chunks in the variable length setup are segmented into subsegments with a smaller size (e.g., 600 frames are segmented into 6 subsegments with 100 frames.). The network uses these subsegments randomly in different mini-batches during training, which results in more randomness during training. It can aid in better convergence in stochastic gradient descent (SGD) and can be the reason for accuracy improvement for the fixed chunk length setup.

The results in rows 3 and 4 are trained on fixed chunks with lengths of 50 and 100 frames, respectively, and the training epoch is increased to 100. The result shows that the network needs longer training time to learn emotional

states using fixed length chunks. The results reported within brackets in rows 3 and 4 are the WA obtained by providing an effectively unlimited left context during decode time. It means the network is allowed to reuse hidden state activations from the previously computed chunk. As can be seen from Table 4.7, unlimited left context helps only with smaller training chunks.

Table 4.7: Effect of training example chunk length. The numbers inside parenthesis are results using looped decoding.

chunk length	epoch	WA	UA
$100 - 600^1$	30	65	53.0
50^2	30	60.78	53.93
50^2	100	66.4 (67.2)	60.3
100^2	100	**70.1** (66)	**60.7**

1: Config2 is used in this setup.
2: Config1 is used in this setup.

Chapter 5

Universal feature extraction

The DNN can be used as a complex feature extractor and act as the language-universal feature extractor to learn the universal structure of speech that is common across different languages. It is shown that DNN is highly effective in learning representation, which is invariant to different variations in data such as speakers, environment, and channels. Researchers used BNFs to leverage out-of-domain resources that can be multilingual or cross-lingual sources. Using data from other sources, the BNFs learn the structure of speech and help to improve ASR performance. Therefore, we can compensate for the lack of training data in the target language. In Section 5.1, we explore multilingual BNF extraction.

In addition to what was described in Chapter 3, one of the main issues in the direct-from-signal setup is the filter bank over-training especially in low

resource conditions. As shown in Subsection 5.2.1, transferring filter banks across English datasets with different channels, recording and noise conditions degrades the ASR performance. Multilingual feature extraction can help solve data scarcity, mismatch issues and close the gap between resource-rich and resource-scarce languages or datasets. Section 5.2 uses a proposed direct-from-signal structure (Section 3.2) to learn a set of universal filters that are shared across different datasets and languages.

5.1 Multilingual feature extraction

We explored training the multilingual DNN and its use for both tandem and hybrid systems. In the hybrid system, the posterior probability for the target language in the multilingual model is directly used for decoding. In this system, a few hours of the target language are required to train the language-specific output layer. In a traditional tandem system [**?**], a DNN is trained to classify context-independent states and the outputs from the DNN are projected down to a low-dimensional space. The BNF, in our tandem system, is extracted from this model and a separate model is trained on the target language using stacked BN and MFCC features.

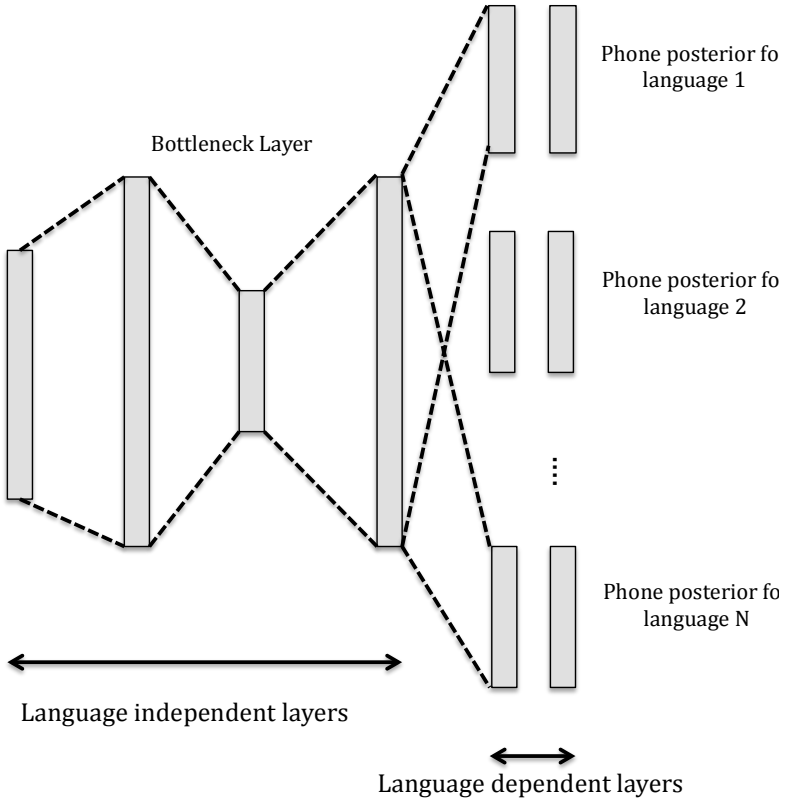

Figure 5.1: Proposed multilingual structure

Data selection

We used language identification on 25 available Babel languages to pick the most similar languages to the target language. In this setup, the closest languages are selected by computing the average language class posterior over all frames for a given target language. We tried two different data selection approaches. In the first approach, 10 randomly chosen languages are selected and in the second approach, the multilingual TDNN model is trained on the

most similar languages. In the second approach, we used the confusion matrix generated from the language-identification system to select 10 languages which are most confusable with Georgian. The language identification system is a neural network which is trained to classify 2-10 second utterances into one of the 25 languages. It uses the TDNN [**?**] and a statistic pooling layer [**?**]. The closest languages are selected by computing the average language class posterior over all frames from a given language. As can be seen, data selection is important in multilingual training and random selection of 10 languages shows less improvement. This shows that selecting languages with a higher similarity to the original language helps learning better multilingual BNFs.

Table 5.1: WER vs. different data selection methods.

Data selection	WER
Baseline	49.7
Random Selection	49.2
LID-Based Selection	47.3

Multilingual setup

In this setup, we used 80 hrs of data from 10 languages identified as the closest ones to Georgian (Lithuanian, Mongolian, Turkish, Kazakh, Kurmanji, Pashto, Swahili, Tok Pisin, Igbo, and Dholuo) to train a multilingual acoustic model for estimating several language-specific senone posteriors. The multilingual acoustic model is an HMM-TDNN hybrid system where the TDNN stucture shares all layers, other than the final affine layer, among different lan-

guages. As shown in [**?**], activations from the higher layers of a DNN are more robust to variations and distortions from the speech signal and the BN layer at a higher layer generates discriminative and invariant feature representation. The TDNN acoustic model has 6 layers where the fifth layer has a 42 dimensional BN. In each epoch of training, the TDNN is trained using mini-batches of data sampled from all 11 languages. The sampling of the mini-batches is based on the relative frequency of data from these languages The high-resolution MFCC (dimension of 40), pitch features (dimension of 3, see [**?**]), are appended as input for the DNN acoustic model. The outputs of the BN layer are termed as multilingual bottleneck features (MLBNFs).

Table 5.2 shows results using MLBNFs. The BLSTM acoustic model trained on 80 hours of Georgian. The result shows 1% absolute WER improvement over baseline setup.

Table 5.2: WER using different features

Acoustic model	Features	WER
BLSTM	MFCC+i-vector	45.9
BLSTM	MFCC+i-vector+pitch	45.0
BLSTM	MFCC+i-vector+pitch+MLBNF	44.0

5.2 Learning universal filter banks

As shown in Chapter 3, the proposed frequency-domain setup produces the best results on various LVCSR tasks. In this section, we investigate the effec-

tiveness of the joint feature extraction method, proposed in Section 3.2, over hand-designed features on the multi-English and multi-language datasets. In Subsection 5.2.1, we investigate the transferability of the filter banks across different datasets. In Subsection 5.2.2, we investigate the effect of learning filter banks using multiple English corpora with various recording and noise conditions. In Subsection 5.2.3, we investigate the effect of language diversity on learning a universal set of filter banks using a frequency-domain setup and multiple language corpora.

5.2.1 Filter bank universality

In this section, we investigate transferability of the filter banks learned on different datasets and whether the filter banks trained on one dataset are applicable to other datasets and can be exploited without degrading the performance. Table 5.3 shows results of 4 English language datasets. The filter banks used in the filter bank layer are the ones trained on different datasets using the proposed frequency setup (i.e., the filter banks for experiments in Table 3.10) and the transferred filter bank layer (i.e., as shown in Figure 3.8) is fixed in all experiments.

The filter bank matrix W is applied on the normalized FFT-bin vector x for a 16kHz input signal. The FFT vector x can be divided into x_L and x_H w.r.t FFT-bins in frequency sub-bands $[0-8]$kHz and $[8-16]$kHz. The filter bank matrix

$\mathbf{W}$ is also split into 4 non-overlapping matrix blocks as shown in Equation 5.1. The main assumption in transferring the filter banks from 8 to 16 kHz datasets is that the matrix norm for submatrices W_{LH} and W_{HL} are close to zero and we can neglect the correlation between low-frequency and high-frequency sub-bands. $\mathbf{W_{HL}}x_L$ and $\mathbf{W_{LH}}x_H$ are negligible.

In transferring the 8kHz filter banks $\mathbf{W_L}$ to the 16kHz dataset, a new randomly initialized set of filters, $\mathbf{W_H}$, for frequency range $[8-16]$ kHz, x_H, is added to the filter banks layer. The filter bank for the 8kHz dataset, $\mathbf{W_L}$, used for frequency bins corresponds to $[0-8]$ kHz, x_L, and the output is the same as in Equation 5.1.

In transferring the 16kHz filter banks to 8kHz datasets, the subset of the filter banks for frequency range $[0-8]$kHz, $\begin{bmatrix}\mathbf{W_L}\mathbf{W_{HL}}\end{bmatrix}^T$ is used as a fixed set of filter banks in 8kHz datasets.

The filter banks trained on Switchboard, TedLium, AMI IHM, and AMI SDM (i.e., experiments in Table 3.10) are used in the feature extraction block (i.e., Figure 3.8) and are fixed. The scale and offset layer in the normalization

block is transferred and retrained on the target datasets.

$$\begin{bmatrix} \mathbf{W_L} & \mathbf{W_{LH}} \\ \mathbf{W_{HL}} & \mathbf{W_H} \end{bmatrix} \begin{bmatrix} x_L \\ x_H \end{bmatrix} = \begin{bmatrix} \mathbf{W_L} x_L + \mathbf{W_{LH}} x_H \\ \mathbf{W_H} x_H + \mathbf{W_{HL}} x_L \end{bmatrix} \approx \begin{bmatrix} \mathbf{W_L} x_L \\ \mathbf{W_H} x_H \end{bmatrix} \tag{5.1}$$

As shown in Table 5.3, transferring filter banks between AMI-IHM and SWBD datasets results in $0.3 - 0.4\%$ WER degradation on both datasets. However, transferring the filter banks between SWBD and AMI-SDM as a distant microphone dataset with reverberation results in larger WER degradation around 1%. The interesting finding is that transferring the filter banks from TedLium to AMI-SDM shows $0.2 - 0.3\%$ WER improvement compared to using an in-domain AMI-SDM filter bank. Also, the results show that TedLium is more robust in transferring a filter bank from other datasets and the WER degradation, by using out-of-domain filter banks, is negligible.

Table 5.3: Effect of transferring filter banks from other datasets

Database	Test Set	Filter Bank			
		8kHz SWDB	16kHz AMI-IHM	16kHz AMI-SDM	16kHz TedLium
SWBD	eval2000	**14.3**	14.7	15.5	15.0
	rt03	**17.0**	17.3	18.3	17.7
AMI-IHM	dev	20.4	20.1	**20.0**	20.1
	eval	20.3	19.9	19.9	**19.8**
AMI-SDM	dev	37.3	36.3	36.3	**36.1**
	eval	41.2	40.5	40.2	**39.9**
TedLium	dev	8.1	8.1	8	**7.8**
	eval	8.5	8.5	8.4	**8.4**

5.2.2 Multi-English dataset

In this section, we combine multiple English corpora, WSJ [**?**], Switchboard [**?**], HUB4, TedLium and Fisher [**?**]. Speed perturbation [**?**] is performed on the amplitude modulated signals with speed factors of 0.9, 1.0 and 1.1 to augment the dataset for DNN training. Switchboard and Fisher are $8kHz$ and WSJ, TedLium and HUB4 are $16kHz$ datasets. To handle different sampling rates for different corpora, all datasets are downsampled to $8kHz$ for narrowband experiments and upsampled to $16kHz$ for wideband experiments. The basic dictionary is prepared on a combination of Switchboard, CMU and TedLium lexicons and a G2P model is trained using the combined lexicon. The pronunciations are synthesized for out-of-vocabulary (OOV) words across all training transcripts using the G2P model and the final lexicon is produced. The speaker adapted training (SAT) is trained on fMLLR adapted features on a subset of

combined datasets. The DNN model contains 7 layers of TDNN with a size of 1024 and despite the model proposed in Section 5.1, a shared output layer is used across all datasets. A 100-dim i-vector is trained on a subset of combined corpora and is added to input features for speaker adaptation.

Learning filter banks using multiple English datasets is more difficult due to higher channel, noise and volume variations. The normalization block in the frequency-domain setup (i.e., Section 3.2) performs global normalization over different datasets, and the scale values in the scale and offset layer can be varied among different datasets which can result in training instability. The filter banks learned on the multi-English dataset are noisy and have higher entropy and multiple peaks. Figure 5.2 compares the center frequency versus the filter index for filter banks trained on multi-English and Switchboard datasets. The noisy filter banks are removed using the entropy threshold. It also compares these center frequencies with the ones computed using DNN-c method (i.e., Section 6.2). As can be seen, there are more filters in some specific frequency sub-bands (e.g., $[0 - 1200]$ Hz) on Switchboard compared to multi-English datasets.

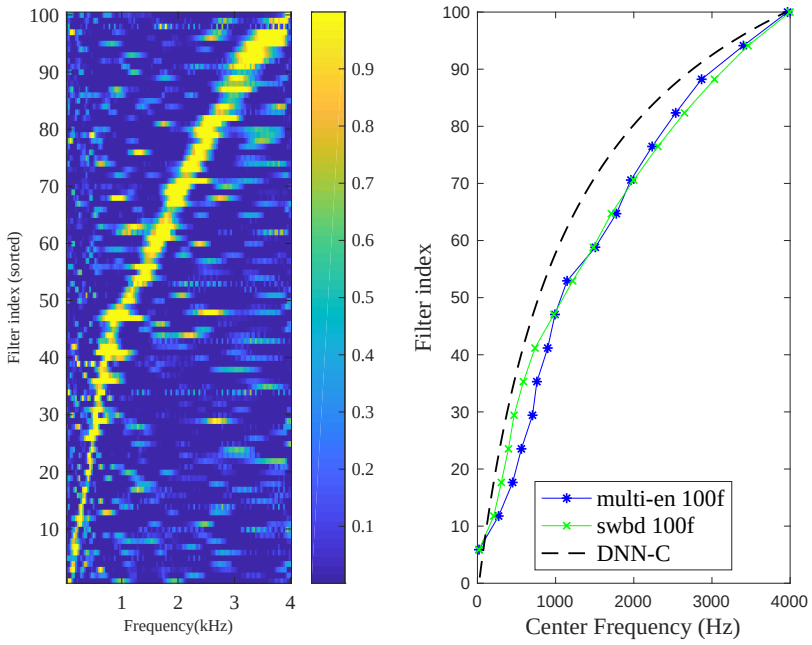

Figure 5.2: Comparison of center frequency vs. filter index for multi-English and Switchboard datasets

Table 5.4 compares the results using conventional 40-dim MFCC and the frequency domain setup proposed in Section 3.2.

Table 5.4: WER (%) frequency-domain feature extraction setup vs. MFCC

Test set	MFCC	Frequency-domain setup
SWBD eval2000	14.1	13.7
SWBD rt03	15.8	14.8
TedLium test	10.9	10.5

5.2.3 Multilingual dataset

Developing a new set of universal filters, that can be applicable to all languages without performance degradation, is challenging. A popular approach

(i.e., used in Section 5.1) is to train multi-language BNFs by training a multi-language acoustic model. In this section, we consider a different scenario, where the target language is unseen during training. To do this, a consistent phone set is required across all languages, and should have good phone set coverage for unseen languages. We train an acoustic model on multiple languages by sharing a common phonemic representation as a universal phone set of ASR. Our approach is similar to that of [**?**]. We use a selection of 21 mixed bandwidth Babel languages that are released by the IARPA Babel program (each LLP consists of 13 hours of transcribed data; 90% of Babel datasets are $8kHz$ and 10% are 44100Hz), $16kHz$ Spanish Hub4-NE and $16kHz$ French and Russian languages from Voxforge. Diphthongs and riphthongs are split into their constituent phones to increase cross-language phone coverage and enforce sharing of the phoneme. Also, we standardize the representation of tone (tonal trajectory) across all training languages. Eleven factorized TDNN layers with dimensions of 256 are used in the network architecture. The dropout-probability technique in [**?**] is used during training. The dropout schedule is in the form of $'0, 0@0.20, 0.5@0.50, 0'$, where it applies no dropout in 20% of the training and the dropout proportion is linearly increased to 50% until reaching 50% training and reduced to 0 at the end of training. The network is trained using LF-MMI criterion.

We also bootstrapped the lexicon using a G2P from provided resources. The

vocabulary is generated from the provided text. In the first set of experiments, the datasets are downsampled to a bandwidth of $8kHz$, and the raw frequency domain block (i.e., proposed in Section 3.2) is used in the raw frequency domain experiments. In the next set of experiments, all datasets are upsampled to a bandwidth of $16kHz$, and have the same topology as the $8kHz$ bandwidth experiments.

Lorelei [**?**] $16kHz$ IL9 from the incident language pack and Haitian Creole (L201) from the IARPA Babel language project are used for evaluation. No data adaptation is done to adapt the universal model to new languages in Table 5.5. None of these target languages are involved during training. Database IL9 is considered as an almost-zero-resource target language with 15-minute data, and the Haitian Creole dev set (L201) contains 10 hours. The results in rows 1 and 2 are evaluated on an unadapted universal model.

Table 5.5: Performance of the universal frequency-domain setup on unseen target datasets

Database	8kHz		16kHz	
	MFCC	Universal filters	MFCC	Universal filters
L201	69	69.6	69.6	69.9
IL9	63.6	62.8	63.3	62.3

Figure 5.3 shows filter banks that are trained on 8kHz and 16kHz multi-language datasets. As shown, 50% of filters are trained on frequency range $[0-1]\text{kHz}$ and the filter bandwidth in this sub-band is increased by increasing the center frequency. Twenty percent of filters are in the high-frequency sub-

band $[4-8]$kHz and the filters in this sub-band are noisy.

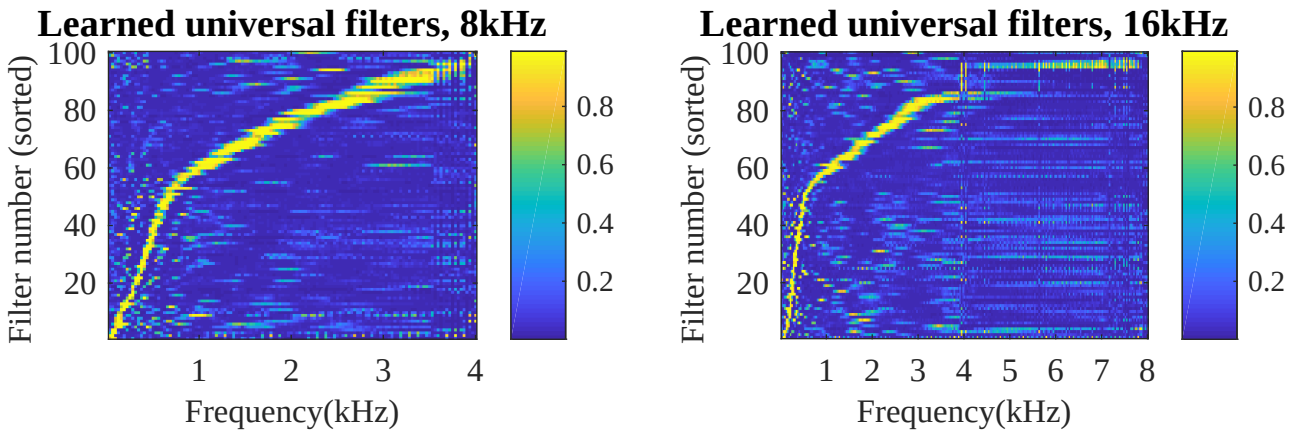

Figure 5.3: Learned universal filter banks on 8kHz and 16kHz multi-language datasets

Chapter 6

Data-driven based feature learning

Mel scale is the most commonly used warping function in extracting features for ASR. It is known to be very effective, however, it is not specifically designed for the current ASR models which are based on DNNs. In this chapter, we propose 3 new features based on filter banks learned in the frequency domain setup (Section 3.2) on different datasets. We first introduce a new frequency warping function which is simple, scalable and invertible. This warping function is parameterized using 3 parameters and we use it to propose a new set of features called DNN coefficients (DNN-c), which uses cosine-shaped filters. The bandwidths are computed using a piece-wise linear function. Additionally, we propose an alternative set of features called formant

DNN-c (fDNN-c) in which we use Gaussian mixture models (GMM) to indirectly capture the formant information. Finally, we propose a modified version of Mel filters, which uses a modified version of Mel scale function and whose bandwidths are computed as a combination of piece-wise linear bandwidth and overlap-based bandwidth. By evaluating the proposed features on a variety of databases, we see consistent improvements over Mel filter bank features.

The Mel scale is a perceptual scale of frequencies which is handcrafted based on physiological models of the human auditory system. This scale is used in the MFCC and log-Mel features, which are perhaps the most commonly used features for ASR. However, they are not guaranteed to work well with the latest ASR models which are all based on DNN.

One approach for extracting features that are more suited to DNNs is to train the ASR model from the signal domain and let the network craft its own features in a data-driven scheme. This is also known as joint feature extraction and acoustic modeling and has been investigated in a few studies [**?**, **?**, **?**].

In particular, in Chapter 3, we proposed a data-driven feature learning layer that can be trained jointly with ASR [**?**]. We used that to learn new filter banks, outperforming the MFCC-based models. The main drawback was that the network learned data dependent filter banks and could overfit to the training data.

To address this issue, we proposed a new analytic filter bank which we es-

timated using the learned data-driven filter banks. We successfully obtained similar results as the learned filter bank using the analytic filters.

In this chapter, we propose 3 new alternatives for MFCC features: DNN-c, fDNN-c and modified MFCC. These features are based on the filter banks directly learned on different narrowband and wideband datasets using the joint feature learning setup proposed in [**?**].

At the core of DNN-c features is an invertible frequency warping function that is used for finding the center frequencies for the filters. This warping function is analogous to the Mel scale. The bandwidth for filters is computed using a piece-wise linear function of the center frequencies.

Formant DNN-c features (fDNN-c) are variants of DNN-c, where the center frequency is computed using piece-wise linear function and the filter bandwidth is a function of filter overlap, where the filter overlaps are approximated on the learned filter banks and depend on the center frequency of the filters.

6.1 Prior work

The main baselines in this chapter are MFCC and Mel filter bank features which are both well known and use the Mel scale [**?**]. The bark scale is another well-known frequency warping function, on which distances correspond with preceptually equal distances [**?**]. A novel warping function based on high-

energy portions of speech signals is proposed in [**?**]. Other data-driven approaches comparable to our work include [**?**] which uses linear discriminant analysis to maximize the separability between linguistic classes and [**?**] which uses the discriminative feature extraction for designing warping functions.

Filter shape

We use cosine-shaped filters for filter shape in 3 new features. The formula used for filter estimation is shown in Equation 6.1, where each filter is specified using a center frequency f_c and a bandwidth w. As shown, the filters estimated using this formula have the same energy.

$$\begin{cases} \frac{2.0}{w}\cos(\frac{\pi(x-f_c)}{w}) & f_c - \frac{w}{2} \leq x \leq f_c + \frac{w}{2} \\ 0 & else \end{cases} \tag{6.1}$$

It can be demonstrated mathematically that the proposed cosine-shaped filters have the bandwidth of $\frac{w}{2}$ according to the noise-equivalent formula.

6.2 DNN-c features

Center frequency approximation

In this approach, we create a new frequency space – called DNN-c space – in which the center frequencies for the filters are linearly spaced. The warping function to transform a normal frequency to DNN-c space is defined as $g(f)$ in Equation 6.2. Using this warping function along with the low and high frequencies f_l, f_h (in the normal frequency space), we can compute the center frequency for the i^{th} filter $f_c^{'}(i)$ as $f_c^{'}(i) = \frac{[g(f_h)-g(f_l)]i}{M-1} + g(f_l)$ in DNN-c space, where M is the number of filters (i.e., bands). Finally, $f_c^{'}(i)$ can be transformed back to normal space using $g^{-1}(f^{'})$.

$$g(f) = \gamma[(f + f_0)^{\alpha} - f_0^{\alpha}] \tag{6.2}$$

The parameters in Equation 6.2 (i.e., f_0, γ and α) are estimated using mini-batch SGD. The training data used to estimate the parameters are the center frequencies for the filter banks trained on narrowband datasets (8kHz SWBD using 40, 100 and 200 filters), wideband datasets (16-kHz AMI-IHM, AMI-SDM, TedLium, and WSJ using 100 filters) and mix-band datasets (SWBD,

TedLium, HUB4 are combined, and down-sampled to 8kHz and up-sampled to 16kHz). This results in the following values for these parameters:

$$f_0 = 900 \tag{6.3}$$

$$\gamma = -0.695$$

$$\alpha = 0.001$$

Figure 6.1 shows the Mel, data-driven and DNN-c center frequencies using the proposed formula for different datasets.

We also compared a new invertible function with the approach proposed in [**?**], which uses a polynomial function to approximate the center frequencies. In the polynomial approach, a new set of filters are approximated using the filter banks learned on the 8kHz Switchboard, which is trained (separately) using 40, 100 and 200 filters. The center frequencies f_c are estimated using a 4th order polynomial which is, in turn, approximated using least-square error minimization on the center frequencies for the 40, 100 and 200 learned filters. The approximated polynomial is shown in Equation 6.4. Nyquist and f are in Hz, and M is the number of filters.

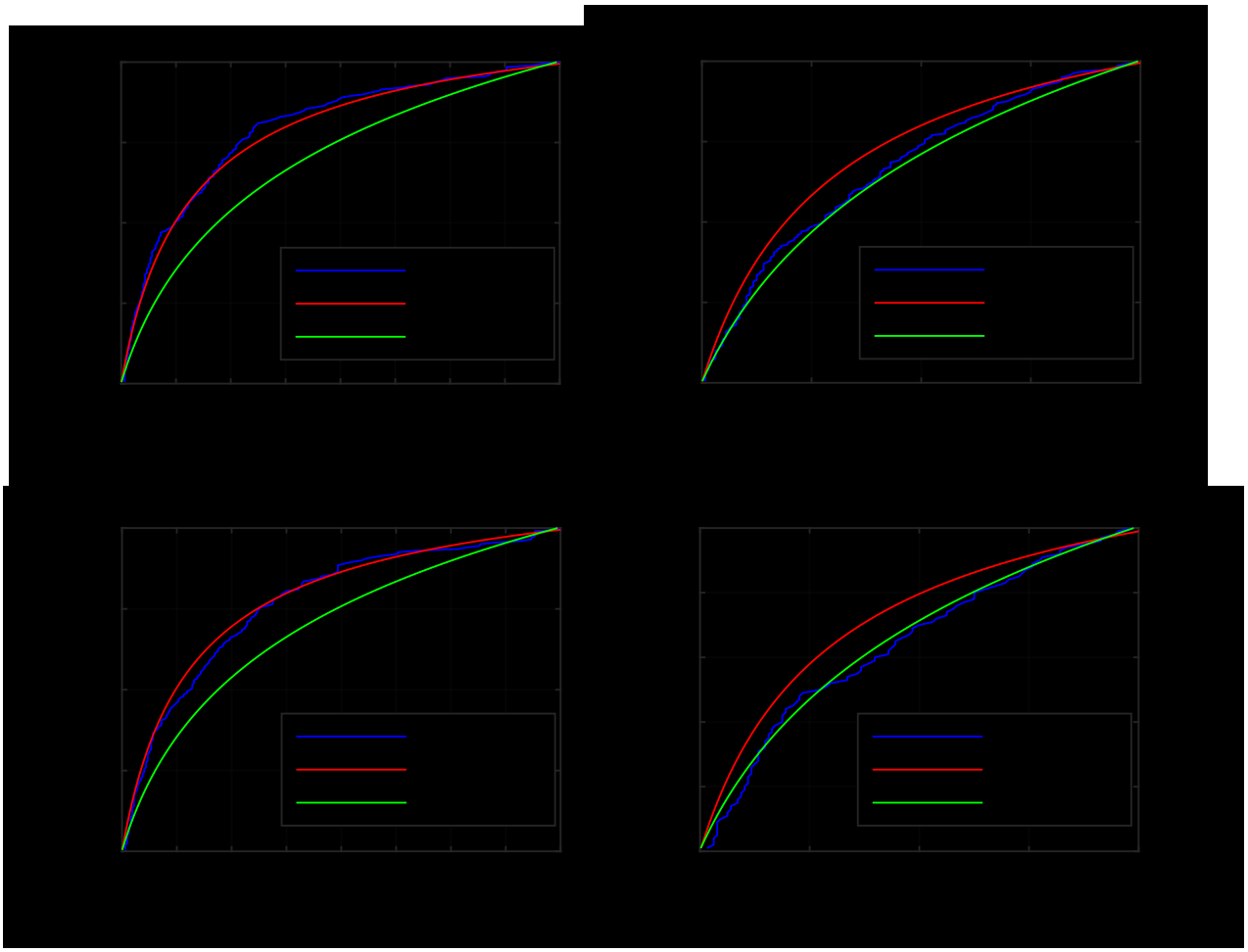

Figure 6.1: Center frequency vs. filter index

$$f_c(i) = a_1 f^4 + a_2 f^3 + a_3 f^2 + a_4 f + a_5 \tag{6.4}$$

$$f = \frac{i \times Nyquist}{M}$$

$$(a_1, a_2, a_3, a_4) = (1.6e^{-11}, -7.4e^{-8}, 2.2e^{-4}, 0.23, 0)$$

We approximate the warping function using center frequency for the learned

filters on narrowband, wideband, and mixed-band datasets. The main issue with the proposed formula in Equation 6.4 is that it is only applicable in the narrowband dataset.

To evaluate the proposed analytic filters (in Section 6.2), we set the filters in the filter bank layer (in the DNN acoustic model) using the proposed analytic set of filters and train the DNN while the filters are fixed. The results are presented in Table 6.1. We can see the proposed analytic filters have outperformed the proposed frequency-domain filters based on which they are approximated. This might be because they are fixed during the training.

Table 6.1: Frequency-domain setup vs. proposed analytic filters

Database	Test set	40-dim MFCC	F-domain setup[1]	Approx. filters Pol. warping[2]	Approx. filters Inv warping[3]
Switchboard	eval2000	14.9	14.3	**14.2**	14.3
	rt03	17.8	17.0	**16.8**	17.1
AMI-SDM	dev	37.3	**36.3**	36.6	36.6
	eval	41.2	**40.1**	40.4	40.5

1: Subsection 3.2.2
2: Eq. 6.4
3: Eq. 6.2

Bandwidth approximation

To measure the filter bandwidth for the learned filters, we considered noise equivalent bandwidth estimation, in which the bandwidth for filter u is defined as $\sum_i u_i^2/(\max_j u_j)^2\delta_f$, where $\delta_f = \frac{Nyquist}{N}$ and N is the number of FFT bins.

We estimate the bandwidth for the learned filters as a piece-wise linear

function of the center frequencies. The plot of the filter bandwidth vs. center frequency for the learned and approximated filters is shown in Figure 6.2. The important observation is that the optimal filter bandwidth stays constant as the number of filters is increased especially in certain frequency sub-bands around formant frequencies; this is not how triangular Mel filter banks are set up. As shown in Figure 6.2, the network learns filters with the same bandwidth in certain frequency sub-bands, for a different number of filters. In other frequency sub-bands, the network with a larger number of filters learns more filters with lower frequency bandwidth.

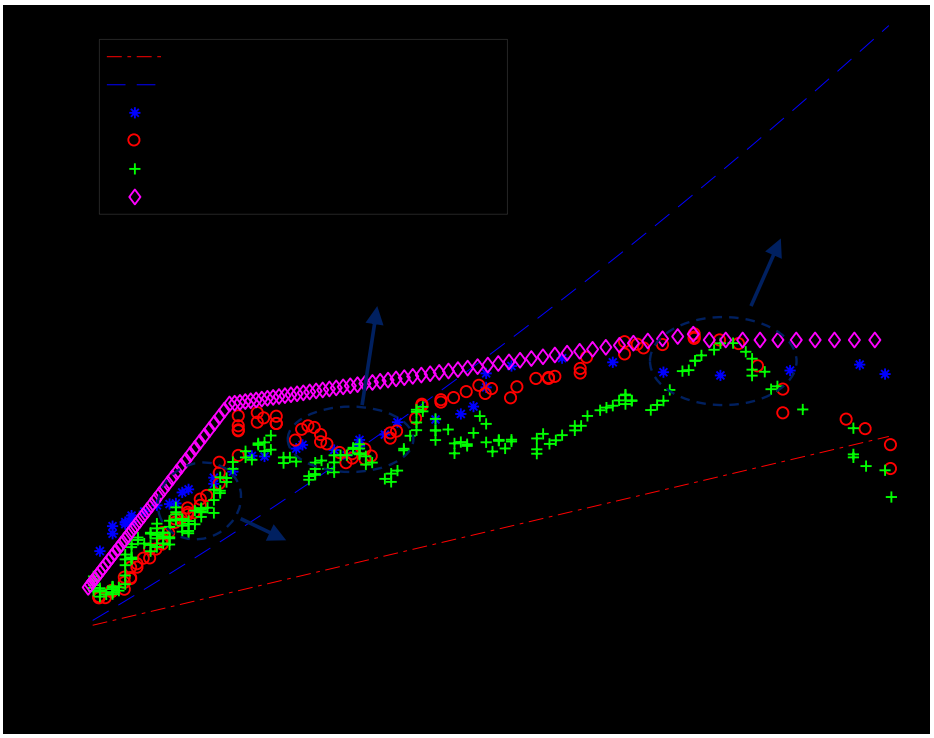

Figure 6.2: Filter bandwidth vs. center frequency for different filter banks

6.3 fDNN-c features

The primary assumption in center frequency approximation in Section 6.2 is that the learned filters are single peak and the center frequency and bandwidth are approximated using this assumption. However, this is not always the case with the filters trained using the frequency-domain setup. As can be seen in Figure 6.3, some of the learned filters in the frequency-domain setup have multiple peaks. The center frequency method used in Equation 6.2 selects the peak with the maximum value as the center frequency. Also, having multiple peaks can affect bandwidth computation in Section 6.2.

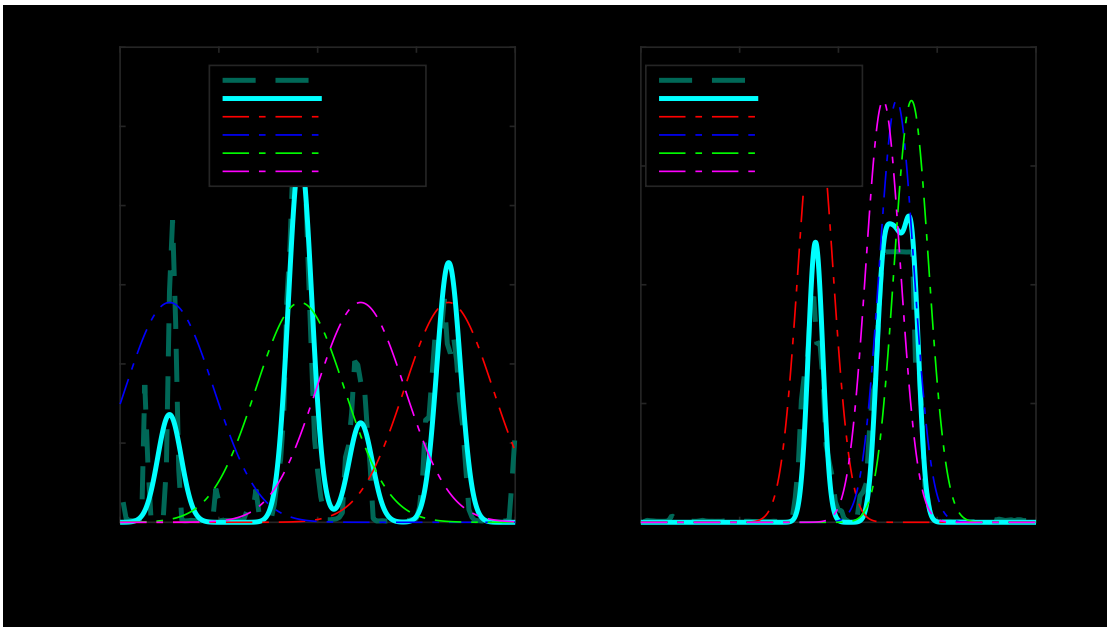

Figure 6.3: Original vs. GMM-based approximation of filters and unweighted GMM components (The ordered weights of GMM components are shown on top of figures).

Another issue is regarding the bandwidth estimation method described in Section 6.2, where we proposed a piece-wise linear function for bandwidth approximation. It is possible that filters do not overlap in some frequency bands especially in high-frequency bands when a smaller number of filters are used in the DNN-c method.

To address these issues, we approximate learned filter banks using GMMs, and we propose a new warping function using the mean distribution of the learned GMMs. Also, we propose a new method for approximating bandwidth for the filters in different regions to solve the issue where some regions were not covered.

Formants are the peaks in the spectrum caused by the resonance of the vocal tract. It has been shown that formant has a smoother trajectory, which is more consistent for a given phone class than MFCC parameters [**?**]. We call this method fDNN-c (formant DNN-c) because the intuition is based on formants and as shown in Figure 6.4, there is a higher filter probability distribution around the first, second and third formants.

Note that in this method, we still use the same cosine-shaped filters that are used in DNN-c, but we use a different warping function as well as a different bandwidth function.

Center frequency approximation

To get the center frequencies for the cosine-shaped filters in this method, we fit a separate GMM to each filter in the learned filter banks. Specifically, for a 100-band filter bank we fit 100 GMMs, where each one can have 1 to 5 components depending on the number of peaks in the corresponding filter. Here, we think of the probabilities in GMMs as weights in the filters.

Once the GMMs are estimated, we compute the weights $p(f_i)$ for N frequencies $f_1, f_2, ..., f_N$ linearly spanning the whole frequency band (e.g., 0 to 4000), as the sum of the probabilities of f_i according to all the GMMs. The weights are normalized so that they add up to one. A plot of these weights (as a function of frequency) is shown in Figure 6.4.

Then we split the whole frequency band into M sub-bands so that in each sub-band the area under the curve shown in Figure 6.1 is the same.

Finally, the center frequencies are determined using the estimated sub-bands and N filters are uniformly distributed among M sub-bands, where each sub-band contains $\frac{N}{M}$ filters (linearly spaced in the sub-band). Figure 6.5 shows center frequency vs. filter index for $8kHz$ and $16kHz$ datasets. The 'DNN-c' line shows the center frequencies computed using the DNN-c method, proposed in Equation 6.2.

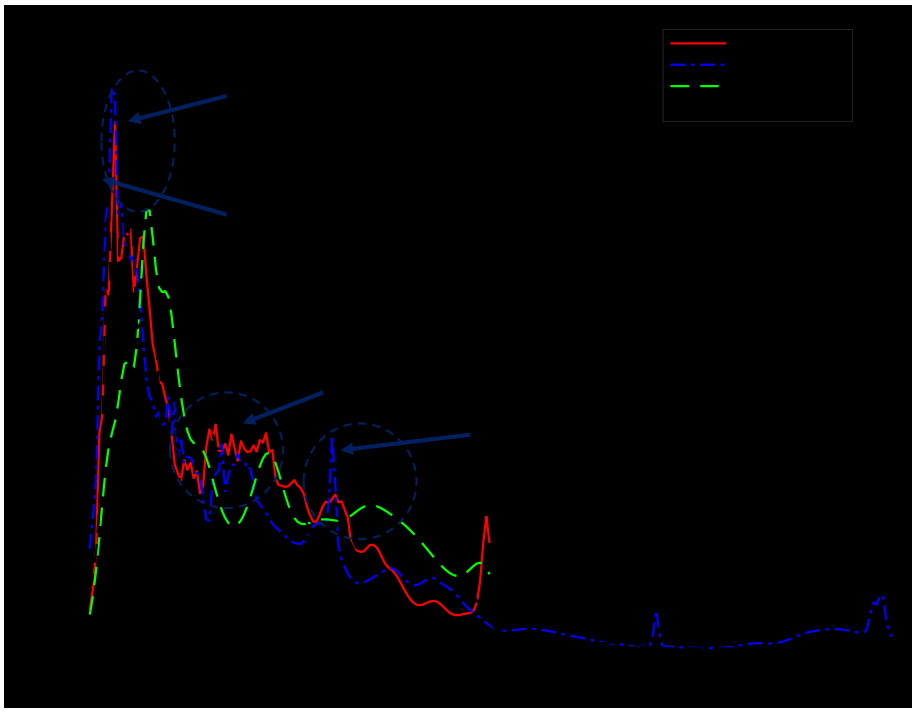

Figure 6.4: GMM estimated center frequency distribution for different datasets

Bandwidth approximation

In this method, instead of approximating the bandwidths based on the frequency-domain learned filters, we determine the bandwidth for each cosine-shaped filter based on the overlaps. Specifically, we compute the overlap in the learned filters as an average of normalized correlations between adjacent filters in each frequency sub-band (i.e., frequency sub-bands computation described in Section 6.3).

Figure 6.6 shows the filter overlap between two cosine-shaped filters that is a function of the ratio of their center distance, d, and the filter bandwidth w_f

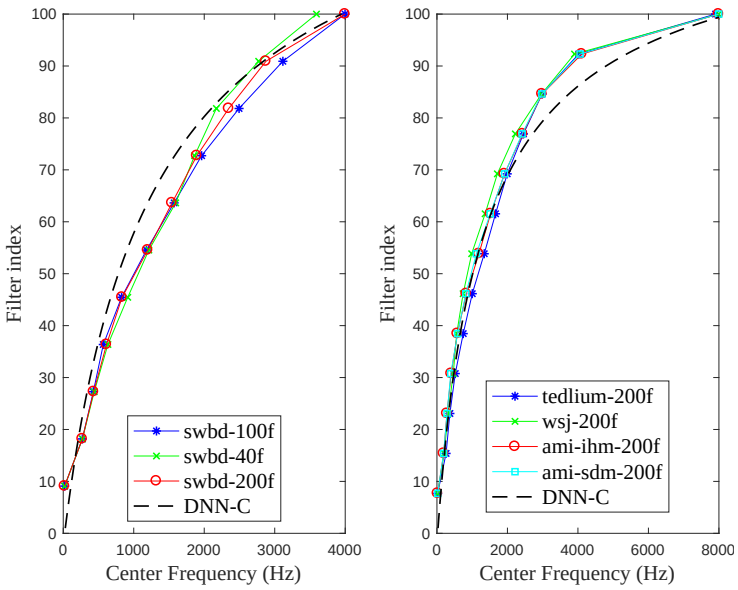

Figure 6.5: Weighted GMM based center frequency vs. filter index for different datasets. Left figure is a $8kHz$ Switchboard dataset and right figure corresponds to $16kHz$ datasets.

(i.e., $r = \frac{d}{w_f}$).

$$f_{op}(r) = -\frac{\sin(\pi(r-1))}{\pi} + (r-1)\cos(\pi r) \tag{6.5}$$

We approximate r as a function of filter overlap op, where $r = (0.42\exp^{2.68op} + 0.9)^{-1}$ if $op \leq 0.8$ and otherwise $r = (0.009\exp^{9.26op} + 0.3)^{-1}$. Then, the filter bandwidth

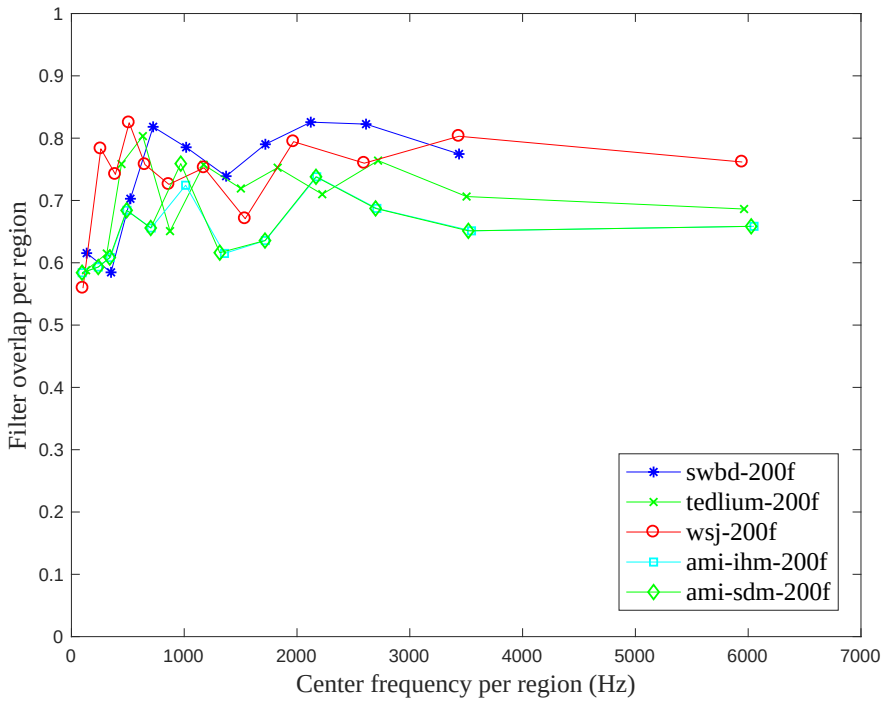

Figure 6.6: Sub-band filter overlap vs. sub-band center frequency

w_f is computed as $\frac{d}{r}$, where d is $\frac{w_s}{M}$ (where w_s is the sub-band bandwidth in Hz and M is the number of filters per sub-band) as defined in Section 6.3.

6.4 Modified Mel filter bank

Center frequency approximation

In this section, a modified Mel scale function is proposed to estimate the center frequencies based on filters learned in the filter bank layer in DNN. As shown in Figure 6.4, the center frequency distribution probability is higher

around the average of the first and second formant frequencies. The original Mel scale function has a higher distribution around 700 Hz and the center frequency distribution is gradually decreased after 700 Hz. Equation 6.6 is a new warping function, that is a modified version of Mel scale function. f_{b_1} and f_{b_2} are the parameters in the modified Mel warping function.

$$g(f) = \log\left(f_{b_1} + f_{b_2}\log(1 + \frac{f}{f_{b_2}})\right) \tag{6.6}$$

$$f_{b_1} \in [300, 900]\, Hz$$

$$f_{b_2} \in [1500, 3500]\, Hz$$

Figure 6.7 compares center frequency for DNN-c, fDNN-c, original and modified Mel warping functions. As can be seen, the modified Mel warping function shows closer center frequency approximation to DNN-c and fDNN-c. f_{b_1} and f_{b2} used in this figure are 300 Hz and 1500 Hz, respectively.

Table 6.2 shows the effect of two cutoffs f_{b_1} and f_{b_2} in modified Mel warping function on Switchboard, AMI-SDM and CHiME5. The bandwidth in the filter banks is approximated using methods proposed in Equation 6.7, where bw_{min} and s_{bw} are at 30 and 60 Hz. Log and DCT transform are applied on the modified Mel filter banks. 80-dim modified MFCC are used in all experiments.

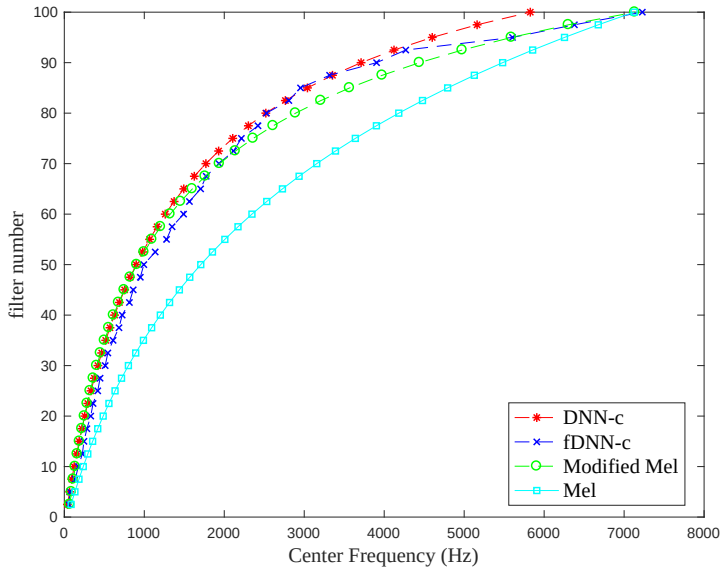

Figure 6.7: Center frequency vs. filter index ($f_{b_1} = 300$ Hz, $f_{b_2} = 1500$ Hz in modified Mel)

Table 6.2: Effect of f_{b_1} and f_{b_2}

(f_{b_1}, f_{b_2})	SWBD		AMI-SDM		CHiME5	
	rt03	eval2000	dev	eval	devworn	devbeamformit
$(300, 1500)$	17.3	14.6	36.2	40.5	45.6	79.7
$(500, 1500)$	17.4	14.6	36.5	40.6	-	-
$(300, 2000)$	17.4	14.6	36.4	40.5	-	-
$(300, 2000)^1$	17.6	14.6	36.6	40.6	-	-
$(300, 3500)$	17.5	14.5	36.6	40.5	-	-
$(500, 3500)$	17.4	14.5	36.6	40.5	54.8	88.1
$(900, 3500)$	17.4	14.7	36.3	40.3	-	-

1: 40-dim modified MFCC

Bandwidth approximation

Sections 6.2 and 6.3 propose two different approaches for filter bandwidth approximation in DNN-c and fDNN-c features. The first approach (i.e., band-

width approximation in Section 6.2) estimates the bandwidth, which is a function of center frequency, using the piece-wise linear function. In this approach, the bandwidth does not depend on the number of filters and the main drawback is that some frequency regions, especially in the high-frequency sub-band, may not cover with any filters. In the second approach (i.e., bandwidth approximation in Section 6.3), the filter bandwidth is approximated to satisfy specific overlap between neighboring filters. Using this approach, the bandwidth strongly depends on the number of filters. However, this behavior is not seen in low-frequency sub-bands for filters learned in the filter bank layer in DNN. In this section, the goal is to combine the benefit of both methods in low and high-frequency sub-bands and combine the bandwidth using a bandwidth estimation formula. Equation 6.7 shows the formula used to approximate bandwidth in the modified Mel filter banks. The bandwidth $bw(i)$ for filter i is a combination of $bw_{lin}(i)$ and $bw_{op}(i)$. $bw_{lin}(i)$ is estimated using linear function and is not a function of number of filters (i.e. $bw_{min} \leq bw_{lin}(i) \leq bw_{min} + s_{bw}$). Also, $bw_{op}(i)$ is computed based on an overlap op to satisfy minimum overlap between adjacent filters.

$$bw_{lin}(i) = bw_{min} + s_{bw}(\frac{f_c(i)}{f_c(i) + f_{b_1}}) \tag{6.7}$$

$$bw_{min} \in [30, 100]\, Hz$$

$$s_{bw} \in [30, 100]\, Hz$$

$$bw_{op}(i) = (f_c(i) - f_c(i-1))(1 + op)$$

$$op \in [0.0, 0.5]$$

$$bw(i) = \sqrt{bw_{lin}(i)^2 + bw_{op}(i)^2}$$

Table 6.3 compares the results using different values of bw_{min} and s_{bw}. The result shows that the WER is less sensitive to these parameters. The LDA layer is applied on the first layer of the network but a CNN layer is not. In this table, f_{b_1} and f_{b_2} at 300 and 1500 Hz, are used in all experiments.

Table 6.3: Effect of bw_{min} and s_{bw} for bandwidth approximation in modified Mel filter banks

(bw_{min}, s_{bw})	SWBD		AMI-SDM		AMI-IHM	
	eval2000	rt03	dev	eval	dev	eval
$(30, 60)$	14.6	17.3	36.2	40.5	20.4	20.6
$(80, 30)$	14.5	17.2	36.2	40.3	20.4	20.6
$(50, 50)$	14.6	17.2	36.4	40.4	20.4	20.6
$(60, 50)$	14.6	17.1	36.4	40.6	20.3	20.4
$(80, 100)$	14.5	17.4	36.4	40.4	20.5	20.7

We also used two different methods to combine linear bandwidth bw_{lin} and

overlap-based bandwidth bw_{op}. To combine bandwidths, two functions were used: $g_1 = \sqrt{bw_{lin}^2 + bw_{op}^2}$ and $g_2 = \sqrt{bw_{lin} bw_{op}}$, where two bandwidth values bw_{lin} and bw_{op} are lower bounds for g_1 and upper and lower bounds for g_2. Table 6.4 compares the results using two combination methods, and bw_{min} and s_{bw} are at 30 and 60 Hz in these experiments. Also, we experimented with different overlap values, 0.0, 0.1, 0.2 and 0.5 and op value of 0.1 gives best performance on 300 hours Switchboard.

Table 6.4: Effect of linear and overlap-based bandwidth combination methods

Method	SWBD	AMI-SDM		
	eval2000	rt03	dev	eval
g_1	14.6	17.4	36.6	40.5
g_2	14.7	17.5	36.8	40.8

Effect of modified Mel filter bank parameters in different noise conditions

In this section, we investigate the effect of different parameters in the modified MFCC features in different noise conditions. We evaluate the effect of 4 different paramters in the modified Mel filter bank, $(f_{b_1}, f_{b_2}, bw_{min}, s_{bw})$ (i.e., see Equations 6.6 and 6.7), in different noise conditions. Through data augmentation, we increase the amount and diversity of the training data. We employ additive noise and reverberation. Reverberation involves convolving room impulse responses (RIR) with audio. For additive noise, we use the MU-

SAN dataset, which consists of over 900 noises, 42 hours of music from various genres and 60 hours of speech from twelve languages. We experiment using 3 different training datasets; 1) clean: 300 hours clean Switchboard dataset (no speed perturbation), 2) add-noise: 300 hours clean is combined with 300 hours data perturbed with MUSAN noises, 3) rvb + add-noise: 300 hours clean combined with 300 hours data perturbed with additive noise and 300 hours data reverberated with RIRs. The training data augmentation details used in this setup are described in [**?**]. We evaluate the setup on noise-added and reverberated copies of rt03 and eval2000. The test datasets are perturbed with different additive noises with various SNR levels (i.e., babble, music and noise with 0-20 dB SNR) and are also artificially reverberated via convolution with simulated RIRs.

Figure 6.8 shows the effect of different parameters for the models trained using 3 different training datasets and evaluated on clean and reverberated test sets. As shown, parameter sets $(1500, 300, 80, 30)$ and $(3500, 900, 80, 30)$ result in the best performance in most cases. Besides, increasing the initial bandwidth bw_{min} degrades the performance. Also, increasing s_{bw} which results in a larger bandwidth for the filter banks in a mid-frequency region degrades the performance. In addition, (bw_{min}, s_{bw}) of $(80, 100)$ degrades the best results using rvb+add-noise training datasets for the reverberated test sets. It shows that increasing bandwidth hurts modified MFCC performace. Consider that

the bandwidth for the frequency response of the whole system is constrained by a frequency response of the window function. The frequency response of a window function can be estimated as a width of the main sidelobe. We used a 25 ms "Povey" window which has a main sidelobe is 60 Hz.

Figures 6.9, 6.10 and 6.11 show the effect of modified MFCC parameters on different types of additive noises. The results show that the test data perturbed with additive noises are less sensitive to the modified Mel filter bank parameters, especially for the models trained on training dataset (2). Overall, parameter sets $(1500, 300, 80, 30)$ and $(3500, 900, 80, 30)$ show the best performance in most cases, especially in rvb+add-noise training condition. The interesting point is that the WER results on test data perturbed with "babble" noise are less sensitive to the parameters, especially on low SNR conditions.

6.5 Results

Performance on clean datasets

We evaluate the proposed DNN-c, fDNN-c, and modified MFCC features on various clean databases, namely TedLium [**?**], Heroico and Switchboard [**?**]. The results are shown in Table 6.5. The amount of training data varies from 10 to 300 hours across these tasks. The baseline is the state-of-the-art TDNN models trained on standard 40-dim MFCC features. We used 69-dim fDNN-c features with a filter overlap of 0.6, that is linearly increased to 0.7 for

$f \leq 3kHz$ and constant overlap of 0.7 for $f \geq 3kHz$. In DNN-c experiments, 60-dim features are used. A 1-byte compression is used to compress the features in both DNN-c and fDNN-c experiments. We used 80-dim modified MFCC features (i.e., Section 6.4), where 300 and 1500 Hz were used as f_{b_1} and f_{b_2} in computing warping function and 80 and 30 Hz represented function g_1 with op value of 0.1 used to compute bandwidth.

Table 6.5: Performance of the proposed features on various databases

Database	Test set	MFCC	DNN-c	fDNN-c	Modified MFCC
Switchboard	eval2000	14.9	14.3	14.4	14.5
	rt03	17.8	17.1	17.1	17.2
Hereico	test	52.4	51.4	51.3	-
	non-native	55.6	54.6	54.3	-
TED-LIUM	test	7.8	7.9	7.6	7.8
	dev	7.4	7.4	7.2	7.4

Performance on low resource dataset

Table 6.6 compares results using Mel filter banks, fDNN-c, DNN-c and modified Mel filter bank features. We can see that DNN-c has achieved the same results as Mel filter banks, while fDNN-c has slightly improved the results.

Performance on far-field datasets

The ASpIRE task data is released as part of the ASpIRE far-field recognition challenge by IARPA [**?**] and it uses the English portion of the Fisher

Table 6.6: Performance of the proposed DNN-c and fDNN-c features on a low resource Vietnamese dataset

Feature	dev 10hrs
Mel filter banks	51.5
DNN-c	51.6
fDNN-c	51.0
fDNN-c[1]	51.2
Modified Mel filter banks	51.4

1: 100-dim fDNNC

dataset [**?**] for acoustic and language model training. Two datasets (5hrs dev and 10hrs dev-test) are provided as part of the ASpIRE challenge for evaluation. A major challenge in this dataset is the severe mismatch between the training data: The training data is telephony speech but the test data is recorded reverberant speech. To overcome this issue, the mismatched reverberation is simulated, and each utterance reverberates 3 times with 3 different impulse responses. In these experiments, we use a 300 hrs subset of a Fisher dataset, which is randomly selected. To investigate the effect of noise and reverberation, two subsets of Fisher datasets are selected (test and dev sets), and the reverberation is applied to both (i.e., test-rvb, dev-rvb) and the performance is reported on both clean and reverberated subsets. The training and validation objective during training of the ASpIRE acoustic model shows that the training is more stable using DNN-c features. The reason can be related to a larger bandwidth in DNN-c features.

The results on far-field data are shown in Table 6.7. As shown, there is more

improvement for AMI-SDM and no improvement for AMI-IHM. Also, there is a small amount of degradation on reverberated test sets for ASpIRE using fDNN-c and the modified Mel filter banks does not degrade the performance. The reason might be that the bandwidth in fDNN-c is larger spacially in a low-frequency sub-band.

Table 6.7: Performance of the proposed fDNN-c and modified Mel filter bank features on far-field databases

Database	Test set	Mel fBanks	fDNN-c	Modified Mel fBanks
AMI-IHM	dev	20.4	20.1	20.2
	eval	20.3	20.2	20.2
AMI-SDM	dev	37.3	36.4	36.2
	eval	41.2	40.4	40.3
ASpIRE	dev	17.0	17.2	16.8
	dev-rvb	24.7	24.8	24.6
	test	17.3 17.3	17.5	
	test-rvb	22.2	23	22.4
	Aspire-dev	65.8	66.1	65.9

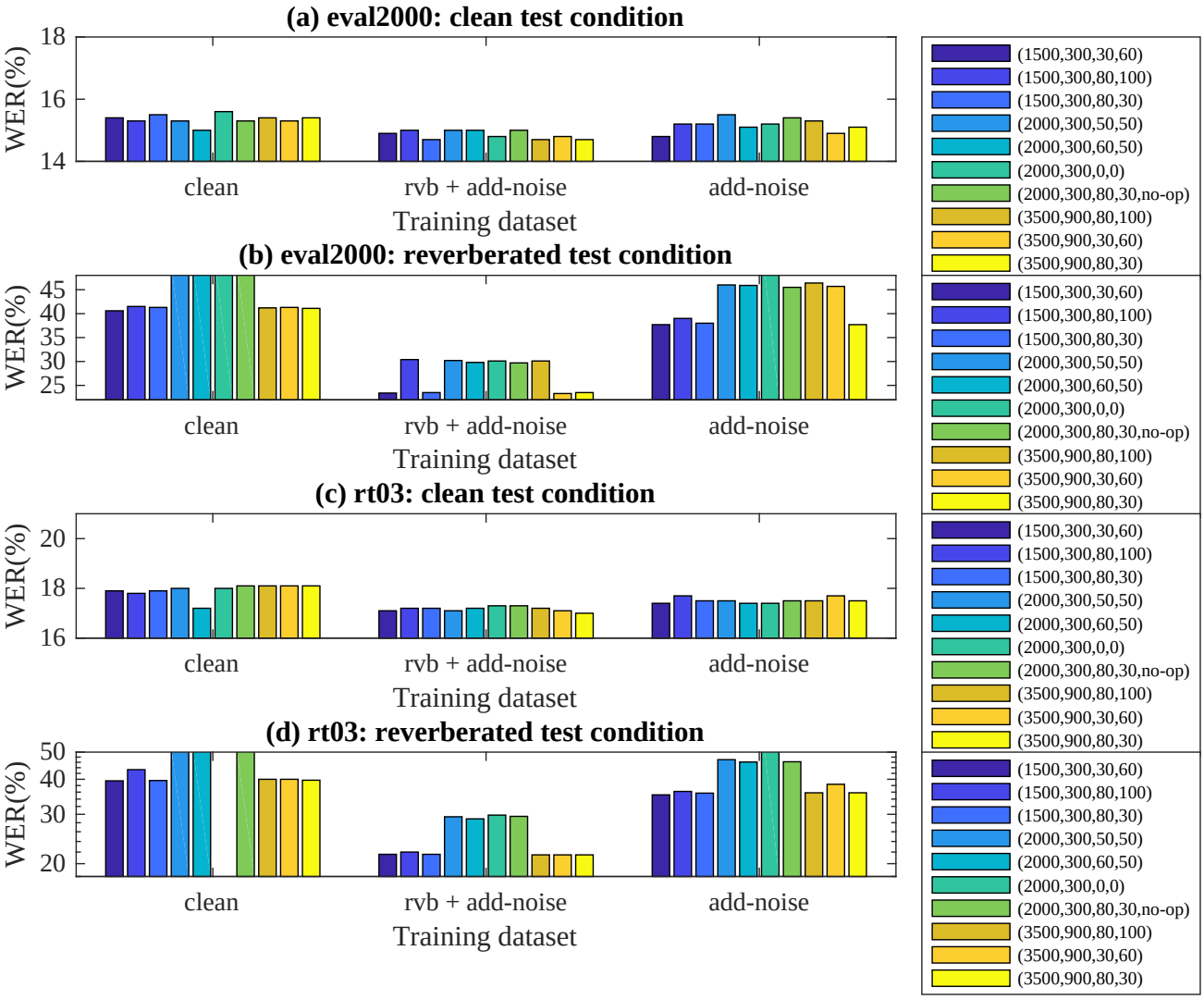

Figure 6.8: Effect of $(f_{b_1}, f_{b_2}, bw_{min}, s_{bw})$ for modified MFCC trained on Switchboard on reverberated test sets.

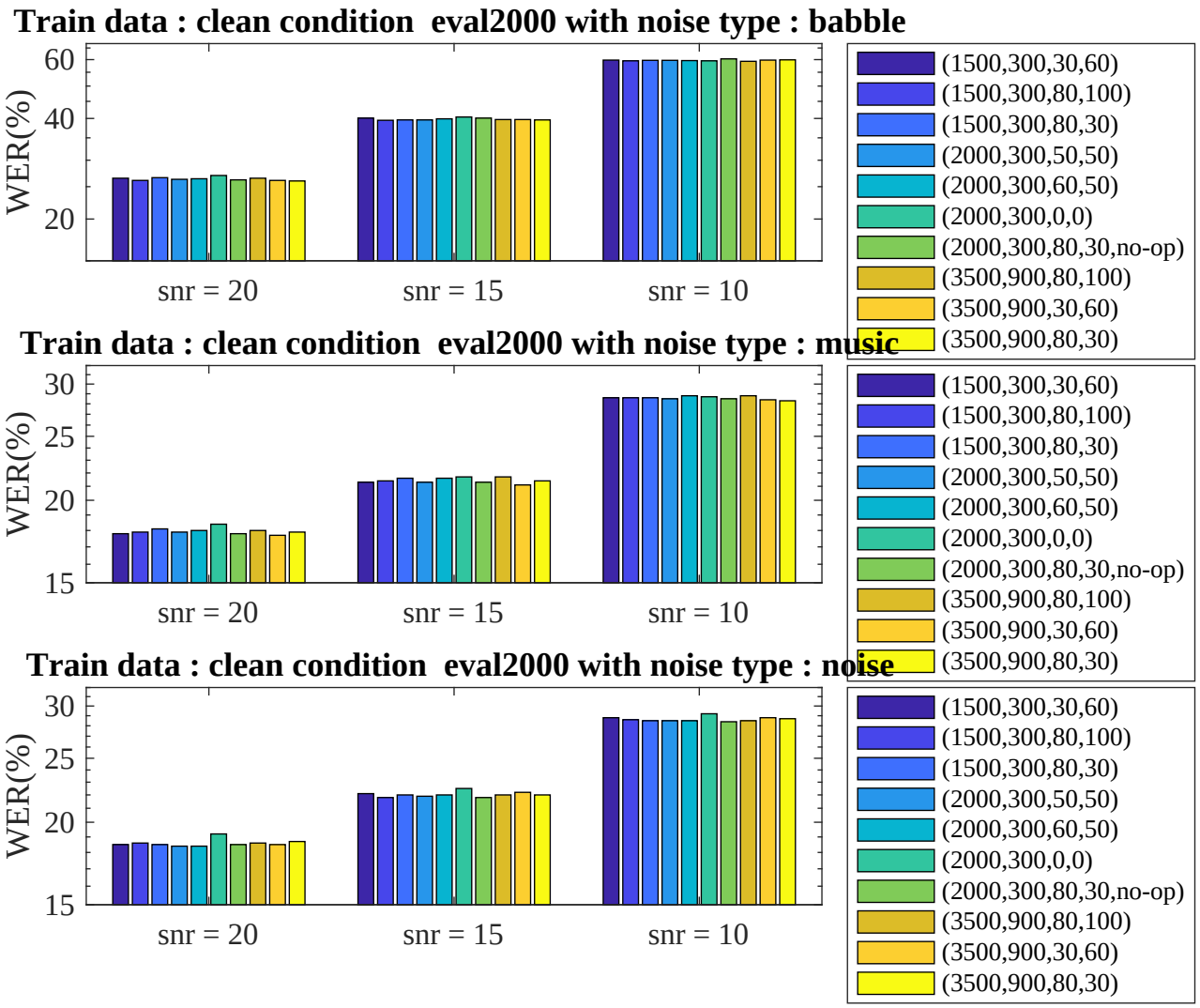

Figure 6.9: Effect of $(f_{b_1}, f_{b_2}, bw_{min}, s_{bw})$ for modified MFCC trained on clean Switchboard on additive noise test sets

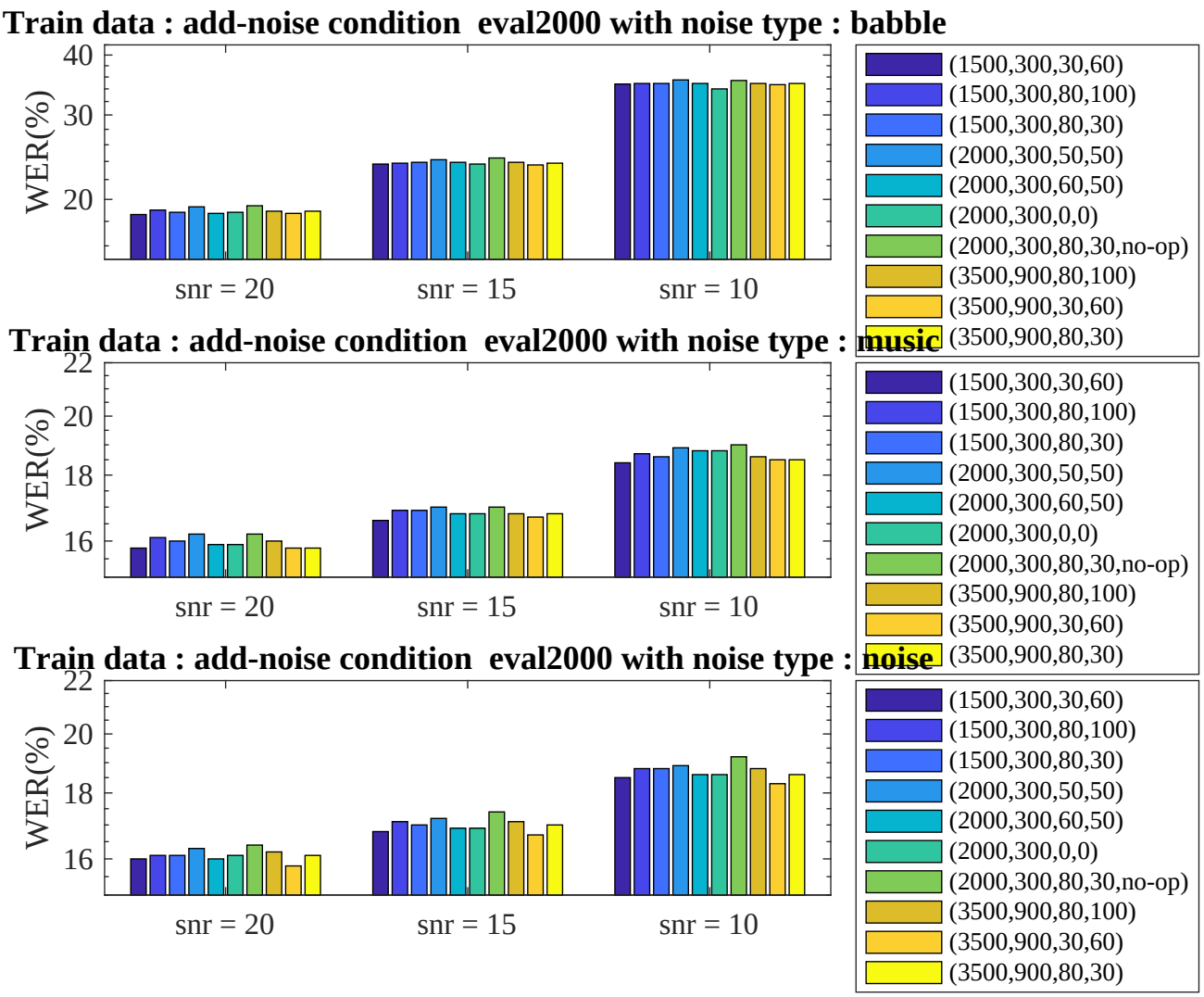

Figure 6.10: Effect of $(f_{b_1}, f_{b_2}, bw_{min}, s_{bw})$ for modified MFCC trained on additive-noise Switchboard on additive noise test sets

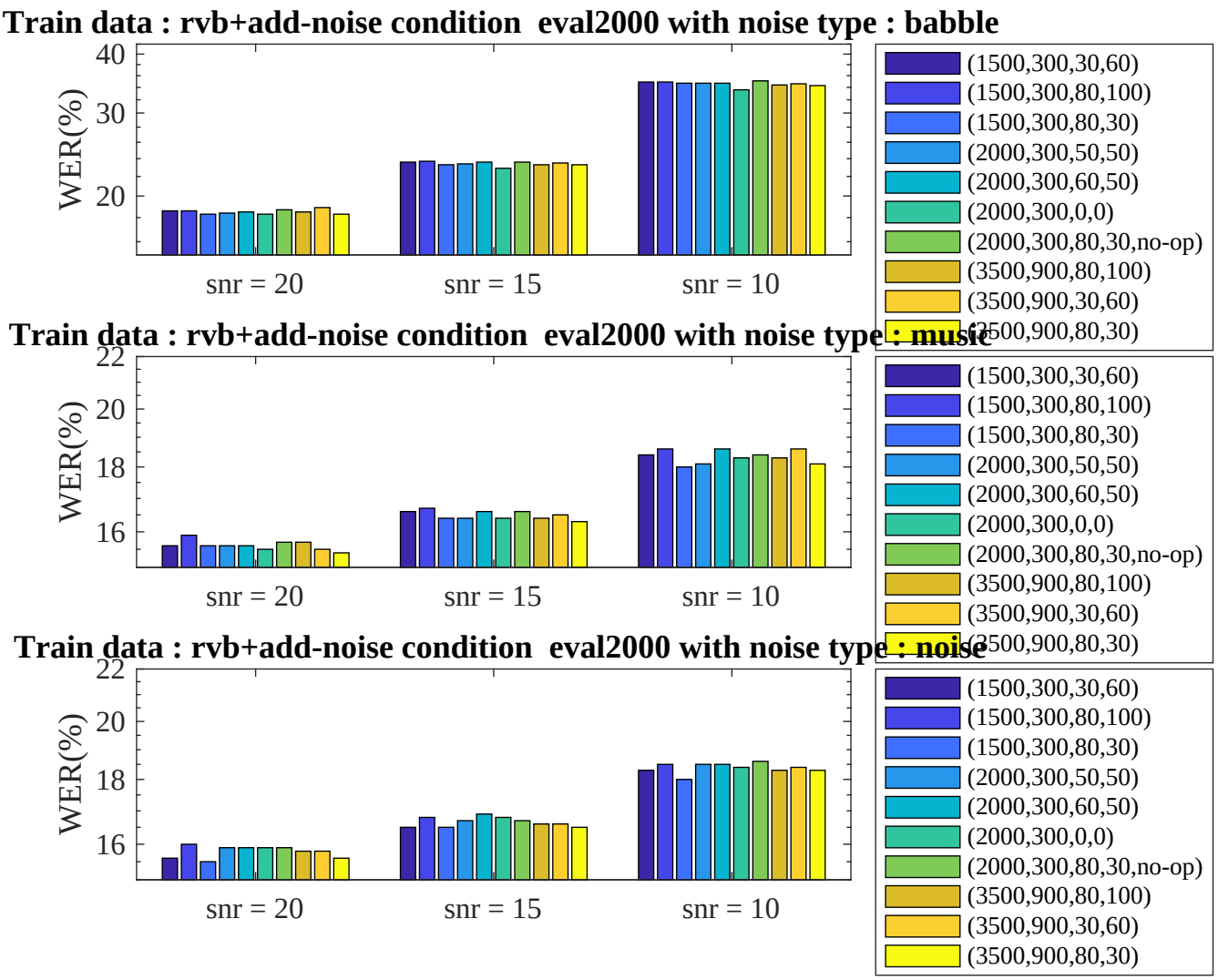

Figure 6.11: Effect of $(f_{b_1}, f_{b_2}, bw_{min}, s_{bw})$ for modified MFCC trained on rvb+additive-noise Switchboard on additive noise test sets

Chapter 7

Deep feature representation transfer across domains

7.1 Prior work

Transfer learning is the general machine learning approach of transferring knowledge from one model to another, and can be used in a different, but related task or domain and can be regarded as a superset of unsupervised adaptation, domain adaptation, model compression, and other related problems. There is a rich survey of transfer learning methods in the literature [**?**, **?**, **?**]. In the case where we have to learn a smaller model on the same domain, the approach is called "model compression". The "domain adaptation" approach is utilized where we have to learn a model in a different domain. In this study,

we investigate the use of transfer learning in ASR tasks for adapting neural networks to different domains or datasets.

One of the advantages of deep learning is to learn a hierarchy of feature representations from low-level to more abstract higher-level features [**?**, **?**]; consequently, it can be useful in transfer learning. Multi-task learning [**?**] has been adopted to explicitly learn intermediate-level features in the neural network that are useful for several different tasks. In an alternate paradigm, pretraining [**?**,**?**] has been used to learn intermediate representations that are useful for different tasks implicitly. The intermediate layers in neural networks that are trained on speech data appear to not be specific to any particular task, while the higher layers are task-specific [**?**]. This has been demonstrated in [**?**], where unsupervised pretraining using deep belief networks (DBN) has been shown to learn representations useful for phoneme recognition and audio classification tasks. Unsupervised pretraining has also been applied to multilingual speech recognition [**?**]. Supervised training using out-of-domain data is also a form of pretraining and has been used to learn multilingual BNFs in [**?**, **?**].

Similar ideas of using learned representations from one model as "guides" to train better or more complex models have shown good success. In [**?**], averaged posteriors from an ensemble of networks are used to guide the training of the networks in the ensemble. In [**?**], a DNN is used to regularize the training of a complex RNN for ASR. In the *FitNet* approach [**?**], networks deeper than

the parent network are trained using regressors in intermediate layers. Deep adaptation network (DAN), proposed in [**?**], relies on a similar idea for domain adaptation by matching mean-embeddings of hidden layer representations.

Transfer learning methods have been applied to speech processing in various settings. Wang et al. [**?**] give a good overall survey of methods used in speech processing. Domain adaptation by adapting network parameters, and in particular speaker adaptation, has been attempted using simple linear input network (LIN) [**?**]. This has inspired more advanced methods like fDLR [**?**] and linear transforms at various stages of the network [**?**] using Liner Hidden Network (LHN). The weight transfer method described in this research is similar to LHN-based adaptation, but we re-initialize an affine layer instead of training a newly added layer. LHN-based adaptation is compared with multitask learning in [**?**]. A speaker adaptive training (SAT) type approach is investigated for speaker adaptation of DNN by learning hidden unit contributions (LHUC) [**?**]. In [**?**], various transfer learning approaches for speaker adaptation are compared including multi-task learning. Multi-task architectures with hidden layers shared across languages have been used successfully for multilingual training [**?**, **?**]. Not only is the amount of data found to be necessary for effective transfer learning, but also the similarity of the languages, i.e., the relatedness of the task [**?**, **?**].

For the experiments, we use TDNN [**?**] with i-vectors [**?**] for speaker adapta-

tion [**?**]. For details about the training of TDNN with the lattice-free maximum mutual information (LF-MMI) objective, the reader is directed to [**?**]. Sequence discriminative training using maximum mutual information (MMI) [**?**] has been shown to improve the performance of frame-level cross-entropy trained neural networks. However, neural networks are trained from scratch using objectives like CTC [**?**] and LF-MMI [**?**], and these usually outperform the frame-level trained ones.

We train the network with an LF-MMI objective and cross-entropy regularization. For our experiments, we use several different corpora – Switchboard, Librispeech [**?**], WSJ and AMI [**?**] in both individual headset microphone (IHM) and single distance microphone (SDM) conditions.

In this chapter, we investigate 3 different approaches to transfer knowledge between networks. For this study, we circumvent the side effect of language similarity (or dissimilarity) seen in multilingual training and focus only on English datasets, albeit in different language domains and environments (channels). Section 7.2 describes a joint multi-task approach for transfer learning.

Section 7.3 describes the weight transfer approach and discusses various training strategies in the weight transfer approach. Section 7.4 investigates the teacher-student (T-S) learning as a transfer learning approach.

7.2 Joint multi-task learning

In this approach, we use the setup where the initial layers of the network are shared across all tasks, and each task has a specific final layer. This approach has been previously used in several studies including [**?**, **?**, **?**]. If tasks are known to have different importance, then they can be weighted proportionally as in [**?**]. Unlike [**?**], which uses model averaging (typically after training over 400,000 frames), we train for different tasks in different mini-batches, which averages over a mini-batch (typically 10,000 frames). This can reduce optimization difficulty due to co-adaptation during training the network. Another issue is over-training to a specific task, which might degrade performance in other tasks as seen in [**?**] when transferring from Fisher English to other languages. To reduce such an over-training effect, the gradients are scaled for each task by a factor inversely proportional to the square root of the number of training samples in that task.

7.3 Weight transfer

The main idea here is that the internal layers of DNN learn intermediate-level representations of input, which can be pre-trained on one dataset (or task) and re-used on the other tasks. A typical weight transfer approach is to first train the model on a large dataset, retain only n layers and add new task-

specific adaptation layers over those.

The usual strategy is to perform two-stage training by freezing the transferred layers and training task-specific layers in the 1^{st} training stage and then fine-tuning the whole network in the 2^{nd} stage of training using a smaller learning rate [**?**]. However, we show in Section 7.3 that it is better to perform single-stage training – train the transferred layers with a smaller learning rate while training the task-specific layers with a more significant learning rate.

In this section, we investigate the effectiveness of weight transfer under various source and target conditions. Further, the effectiveness of the weight transfer approach is investigated as a function of the amount of data in the source and target domains. In addition, the performance of the weight transfer as a function of the performance of the source model on the source domain is investigated.

Single-stage vs. two-stage training

Table 7.1 shows results using two different weight transfer strategies. In these experiments, as shown in Figure 7.1, 5 layers of the source model, that are trained on the Switchboard dataset, are transferred to the AMI-SDM dataset and 2 randomly initialized layers added on top of the transferred layers. The global learning rate is the same in all stages of experiments, and the learning rate for each layer is the global learning-rate scaled by its learning rate

factor. In two-stage training, the transferred layers are fixed, and only the task-specific layers are trained in the first stage with a learning rate factor α for s_1 epochs. Then, in the second stage, the whole network is fine-tuned using a smaller learning rate factor β for s_2 epochs. In single-stage training, the transferred layers are trained with a learning rate factor α, while newly added layers are trained with the global learning rate, all for s_1 epochs.

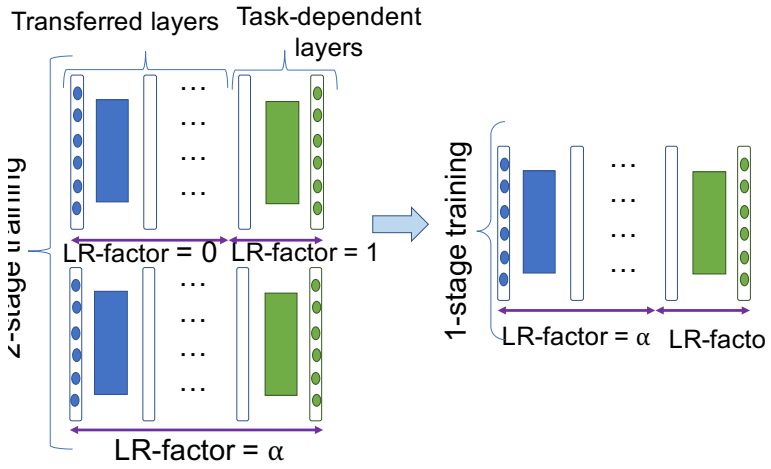

Figure 7.1: Overall single-stage vs. two-stage weight transfer training architecture

The i-vector used in all experiments in Table 7.1 is extracted using the SWBD extractor. The "baseline" row of Table 7.1 reports the results using the model trained on just a 8kHz AMI-SDM dataset for 4 epochs with no weight transfer. As shown in the table, single-stage training shows better results than the conventional two-stage training with a smaller number of epochs. The single-stage results improve as we increase the learning rate factor α.

Besides, fine-tuning the single-stage trained model by training the whole

model with a smaller learning rate does not improve the results as shown in experiment 6 in Table 7.1. In a single-stage* model, the single-stage trained model is fine-tuned for $s_2 = 1$ epoch.

Table 7.1: Single-stage vs. two-stage WER results on SWBD→AMI-SDM.

#	Model	LR factors		# epochs		WER%	
		α	β	s_1	s_2	dev	eval
1	Baseline	1	-	4		45.3	50.0
2	two-stage (s1)	0.25	1	4	2	50.6	55.0
	two-stage (s2)					46.5	51.2
3	two-stage (s1)	0.25	1	2	2	51.8	56.3
	two-stage (s2)					46.4	51.5
4	single-stage	0.02	-	4	-	45.4	50.3
5	single-stage	0.1	-	4	-	44.5	49.7
6	single-stage	0.1	-	2	1	44.5	49.4
	single-stage*					44.3	49.5
7	single-stage	0.25	-	2	-	44.0	48.9

*: fine-tune whole net

Number of transferred layers

The initial layers in the DNNs are "generic" and final layers are task-specific; so there must be a transition boundary from generic to specific in some layers. To investigate this, we conduct two weight transfer experiments to target corpus AMI in IHM condition, one with Librispeech as the source corpus and the other with SWBD as the source corpus. The number of layers in the Librispeech and SWBD neural networks were 6 and 7, respectively. The neural networks all had TDNN architectures with the same overall input context.

In the first case with Librispeech as the source, the results in Figure 7.2

show that the most considerable WER reduction was achieved by transferring half of the layers (3 or 4 layers out of 7). On the other hand, for the case of SWBD as the source corpus, the most substantial WER reduction was achieved by transferring a larger proportion of layers (5 out of 6 layers).

In the case of SWBD, a larger proportion of transferred layers might be better because SWBD and AMI-IHM senones are more similar compared to Librispeech and AMI-IHM. This might be expected because SWBD and AMI-IHM are both spontaneous speech corpora, while Librispeech is a read speech corpus.

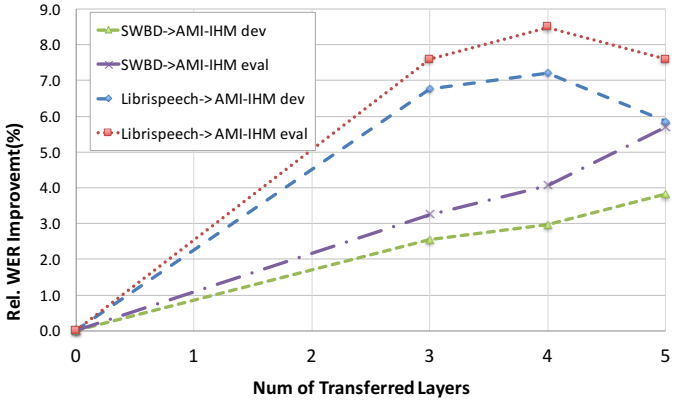

Figure 7.2: WER(%) vs. number of transferred layers for Switchboard to AMI

In the weight transfer approach described above, the last layer is not usually transferred since the phone set, the tree in the source and the target domain are different. However, in some cases, we can use the same phone set for both the source and the target data, and hence share the tree and the senones. In these situations, we can transfer the whole network, including the last layer.

This is particularly useful in cases where the target corpus is very small compared to the source corpus, as found in the case of the MGB-3 challenge [**?**].

We share the phone sets for Librispeech and WSJ and do weight transfer from Librispeech to WSJ by transferring all the layers. Figure 7.3 shows the results of weight transfer for a different number of transferred layers including the whole network transfer of 7 layers. Here, transferring all the layers shows the best WER performance.

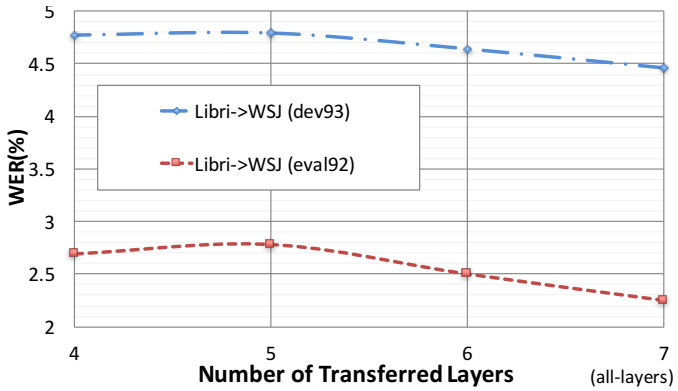

Figure 7.3: WER(%) vs. number of transferred layers for Librispeech to WSJ

Amount of target data

To investigate the effect of the amount of data in the target corpus in transfer learning, we experiment with transfer learning from 1000 hours Librispeech corpus to 80 hours WSJ. The amount of data in the target WSJ corpus varied by using subsets containing 84 (15h), 144 (40h) and 284 (80h) speakers, respectively. For comparison, the results of training directly on WSJ data subsets, i.e.,

without transfer learning, are shown as "baseline". In all these experiments, the i-vector extractor trained on only Librispeech data. Figure 7.4 shows the WER results for baseline WSJ and transferred model trained using the WSJ subsets. As seen in the figure, using transfer learning with 84 speakers shows the greatest improvement and it decreases as we add more data to the target corpus. The results show that weight transfer is most effective when the amount of data in the target corpus is small and insufficient to train a good model.

Interestingly, Figure 7.4 shows that the improvement to the 84-speaker WSJ model due to weight transfer is greater than that obtained by using the rest of the 200 in-domain speakers in WSJ data (i.e., extra 65 hours in-domain data).

As explained before, the phone sets and lexicons in Librispeech and WSJ are similar which allows us to transfer the final layer too. Therefore, we tried a full network transfer with varying amounts of target data. Since the final layer in DNNs is usually a large transformation with the dimensionality of hidden layer sizes by the number of senones, training this layer from scratch can be more difficult when there is a less amount of training data in the target domain. As a result, we expect the already-trained final layer transfer to be more helpful in such cases. This can be observed in Figure 7.4. The "baseline" results in the figure are the models trained using only the WSJ dataset with

a different number of speakers and the initial model in the "baseline" results is randomly initialized. The i-vector for the target WSJ in all experiments was extracted using source extractors.

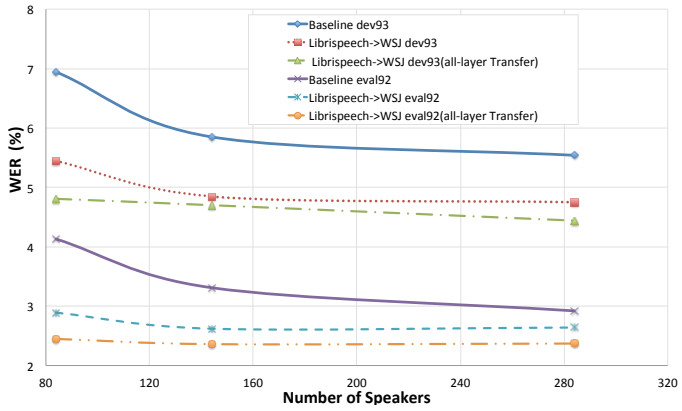

Figure 7.4: WER(%) vs. size of target WSJ corpus (in number of speakers) for baseline and transferred model from Librispeech

Power of source model

In this section, we investigate the effectiveness of the weight transfer method as a function of the number of parameters used in the source model and the number of epochs used to train it. We answer whether a weak source model (as measured on source test set performance) can still be useful as a seed for weight transfer. Table 7.2 shows the WER results for weight transfer from SWBD to AMI in both IHM and SDM conditions. The baseline source model in SWBD is trained for four epochs. The first column of the table shows WER results on the *eval2000* test set for SWBD using a different source model. The results, using the baseline model for weight transfer, are shown in row 1 of the

table. If, instead, we train a model that has only 30% of the number of parameters as the baseline model, the performance on the *eval2000* test set drops by 0.8 to 17.8%. Using this as a seed for weight transfer, we get a WER performance that is worse as shown in row 2. However, if we train the same model as in the baseline, but for fewer epochs like 1 or 2 instead of 4, and use it as a seed for weight transfer, the WER performance is closer to that of the fully-trained baseline model. This is despite the seed model having a WER (17.9%) on the source domain as weak as the model that has 30% of the number of parameters (17.8%).

These results suggest that the source model can learn some generic features from the source data in the initial stages of training that are useful for the target data. However, the later stages of training will learn features specific to the source data that are not very useful for the target data.

Table 7.2: WER(%) results for different source models: SWBD $\rightarrow$ AMI.

	Source	Target corpus			
	SWBD	AMI-SDM		AMI-IHM	
Model	eval2000	dev	eval	dev	eval
4 epochs	17.0	43.5	48.7	22.5	22.8
4 epochs 30% Params	17.8	45.6	50.3	23.3	23.6
1 epoch	17.9	43.8	48.9	22.8	23.3
2 epochs	17.1	43.5	48.8	22.6	23.1

Effect of i-vector extractor

We use i-vector based speaker adaptation of DNNs. In the weight transfer learning approach, we have to use the same i-vector extractor for both pieces of training on source data as well as adaptation to target data. In this section, we investigate the effect of using different i-vector extractors. We conduct weight transfer experiments from SWBD to AMI in SDM condition (down-sampled to 8kHz) using i-vector extractors trained in 3 different ways – one trained only on the source (SWBD) data, one trained on 25% of the subset of data from source and target, and one trained only on the target (AMI) data. The "baseline" columns of Table 7.3 report results on directly training the neural network on the AMI data, i.e., without transfer learning. Comparing the last three rows with the first row showing WER without i-vector adaptation, we can see that any of the three extractors show 3-4% absolute improvement. This suggests that even an out-of-domain i-vector extractor is suitable for speaker adaptation in ASR. These are also 1-2% better than the CMVN normalization results shown in row 2 of the table. The "weight transfer" columns in Table 7.3 show that using the extractor trained on combined data (row 4) shows more than 1% absolute improvement over using an i-vector extractor trained only on the source data (row 3).

Table 7.3: Speaker adaptation: 8kHz SWBD → 8kHz AMI-SDM WER(%) results

Extractor	Baseline		Weight transfer	
	dev	eval	dev	eval
No adaptation	49.5	53.7	45.7	50.0
CMVN adaptation	48.0	52.7	45.8	50.7
i-vector adaptation				
SWBD extractor	46.2	51.1	44.6	49.5
Combined extractor	45.8	50.8	43.5	48.7
AMI-SDM extractor	45.3	50	-	-

7.4 Teacher-student transfer learning

Teacher-student (T-S) learning is a transfer learning approach, where a teacher network is used to "teach" a student network to make the same predictions as the teacher. Originally formulated for model compression, this approach has also been used for domain adaptation. It is particularly effective when parallel data is available in source and target domains [**?**]. The standard approach uses a frame-level objective of minimizing the KL-divergence between the frame-level posteriors of the teacher and student networks. However, this may not apply to state-of-the-art speech recognition models that are trained at the sequence-level, and in particular using LF-MMI.

A sequence-level objective for T-S learning was introduced in [**?**] for model compression from an ensemble. In that study, the KL divergence between the sequence-level posteriors of the teacher and student network was interpolated

with an MMI objective for sequence-discriminative training. However, it was applied in the lattice-based discriminative training framework. Unlike that research, we apply sequence-level KL divergence in the lattice-free training framework and domain adaptation scenario. In [**?**], a lattice-free sequence-level KL divergence objective was introduced for model compression. Our research in this section differs in how the supervision for training the student is generated. In particular, we make use of the lattice supervision used for semi-supervised training in [**?**]. An alternative view to the KL divergence objective is that of a regularizer which prevents the model from diverging too much from what the original model predicts. Thus, the KL divergence was used as a regularizer in supervised adaptation in [**?**] to prevent the model from over-fitting to a small adaptation set. A sequence-level KL version of this idea was used to regularize LF-MMI training in [**?**] for supervised adaptation to small adaptation sets. On the other hand, our work in this paper focuses on unsupervised domain adaptation, i.e., when we have unsupervised target-domain data. In this context, we can view the sequence-level KL objective to be regularizing semi-supervised LF-MMI training to prevent the model from over-fitting to the unsupervised data.

Sequence-KL objective

We propose using the KL divergence between sequence-level posteriors of the teacher and student network as the objective for T-S learning as shown in Equation (7.1). This objective is similar to the ones in [**?**, **?**], but our approach differs in the implementation. We describe our method and the differences in the rest of this section.

$$\mathcal{F}_{\text{KL}} = -\sum_{r}\sum_{\pi\in\mathcal{L}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*) \log \left[\frac{\mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*)}{\mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda)}\right] \tag{7.1}$$

$$\propto \sum_{r}\sum_{\pi\in\mathcal{L}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*) \log \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda)$$

$$= \sum_{r}\sum_{\pi\in\mathcal{L}} \mathbb{P}(\pi \mid \mathbf{O}^{(r)}; \lambda^*) \log \Big[\mathbb{P}(\mathbf{O}^{(r)} \mid \pi; \lambda)\mathbb{P}(\pi)\Big]$$

$$- \log \mathbb{P}(\mathbf{O}^{(r)}; \lambda) \tag{7.2}$$

In Equation (7.1), the probability distributions corresponding to the teacher and student models are parameterized by their neural network parameters λ^* and λ, respectively.

The second term in Equation (7.2) i.e., the log-likelihood under the student network, is entirely independent of the teacher network and is the same as the denominator term in the MMI objective. In [**?**], this term was computed using a denominator lattice created using a weak LM. However, we compute this by

considering all the paths in a fixed denominator graph $\mathcal{G}_{\text{DEN}}$, just as in the case of LF-MMI. The denominator graph is created using a 4-gram phone LM [**?**] trained using interpolated counts from source and target domains [**?**].

We compute the first term in Equation (7.2) as a summation over HMM state sequences $\pi = s_1 \dots s_T$ in the lattice supervision $\mathcal{L}$ created by decoding the utterance using the teacher network. We use a strong 3-gram or 4-gram word LM, preferably in the target domain to do the decoding. This results in sharper target posteriors for training the student network compared to computing the summation over HMM state sequences in a weak denominator graph $\mathcal{G}_{\text{DEN}}$ as done in [**?**]. We give more details about creating the lattice supervision in Section 7.4.

Domain adaptation

In this section, we describe how we use T-S learning for domain adaptation.

Teacher-student learning is useful for domain adaptation when parallel data is available or can be artificially created [**?**]. For this, we first use a trained teacher network to decode the data in the source domain and dump the lattices. The performance is expected to be better if the teacher network was trained on data matched to the source domain. These lattices are converted to supervision using the smart splitting method in [**?**]. What this means is that when we create the supervision, we split the lattice into chunks, keeping the relative scores

of different paths in the chunk the same as they would in the original lattice. This allows us to use the same supervision for training the student network using either LF-MMI or sequence-KL.

We then train the student network from scratch using parallel data (parallel to the data in the source domain) in the target domain and the supervision created as described above. Note that if we do not have parallel data, we can still use the same data to propagate through the teacher network and to train the student network.

One distinction between the supervision used for sequence-KL and the one used for LF-MMI is that the former uses a frame tolerance of 0. In [**?**], frame tolerance of $\pm 30ms$ was used for LF-MMI to allow senones to appear slightly ahead or behind where they appeared in the lattice. For sequence-KL, we had to force the senones to appear precisely where they did in the lattice. The reason for this is that if we used a higher frame tolerance, we would have to recompute the acoustic scores in the supervision by propagating through the teacher network. However, for a frame tolerance of 0, we can use the existing acoustic scores in the lattice and dump the numerator posteriors.

7.5 Transfer learning in sampling rate mismatch condition

One of the main difficulties in transfer learning for ASR is the sampling rate mismatch of the recordings (e.g., 8kHz, 16kHz, etc.). Transferring information across datasets with different sampling rates requires down-sampling of data, which results in some loss of high-frequency information and degrades ASR performance. To verify this, we trained two separate TDNNs – one using MFCC features extracted from 16kHz data and the other from down-sampled 8kHz data– on AMI corpus in SDM condition. The results in the first two rows of Table 7.4 show that removing high-frequency information results in 3 to 4% WER degradation. The deletion error is decreased in the down-sampled experiment, while the substitution error is increased considerably in the far-field scenario.

We used the transfer learning approach to adapt the 8kHz trained network to the 16kHz features on the AMI-SDM dataset. Here, we retrain the first affine transform, W, after input features and fine-tune the rest of the network with a much smaller learning rate. In the experiment indicated by $(*)$ described in Figure 7.5, a new affine transform W is added before the LDA layer of the 8kHz trained network and this transform is initialized to regress the 16kHz features to 8kHz features. The results in rows 3 and 4 of Table 7.4 show that

the information loss due to pretraining on down-sampled data as opposed to the full-band 16kHz data can only be partially recovered by learning a simple new adaptation layer at the initial layer.

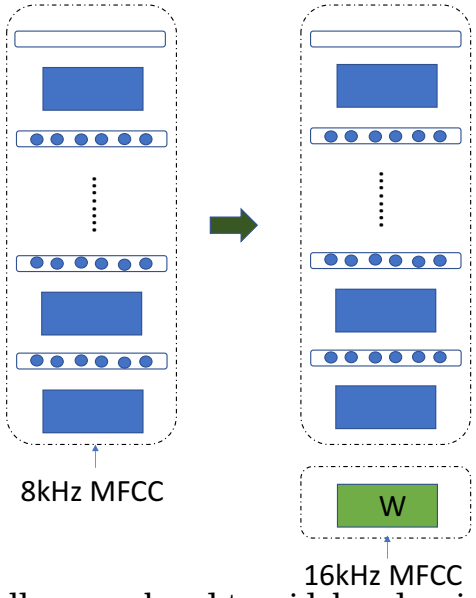

Figure 7.5: Overall narrowband to wideband weight transfer architecture.

Table 7.4: WER (%) results on AMI-SDM

Model	WER	
	dev	eval
16kHz	40.5	44.4
8kHz	43.6	48.6
8 ⇒16 kHz transfer	41.9	46.2
8 ⇒16 kHz transfer*	42.4	47.2
16kHz SWBD⇒AMI TS learning	39.4	43.9
8kHz SWBD⇒ 16kHz AMI 2-stage transfer[1]	40.1	44.5

1:1^{st} stage:8kHz⇒16kHz SWBD, 2^{nd} stage:16kHz SWBD⇒16kHz AMI.

Mixed-bandwidth ASR training, which combines narrowband (inserting ze-

ros for high-frequency bands) and wideband speech signal can improve ASR performance on a narrowband test domain [**?**]. The second transfer learning approach is to train the source model using MFCC features extracted from up-sampled 16kHz data and transfer the layers from the up-sampled source model using the weight transfer approach in Section 7.3. We tried this by using a source model trained on up-sampled 16kHz SWBD data as the seed model for weight transfer to AMI-SDM. As shown in row 4 of Table 7.4, this approach shows 1% absolute WER improvement over the baseline that uses only 16kHz AMI-SDM data.

The last experiment in Table 7.4 shows T-S transfer learning results from $8kHz$ SWBD to $16kHz$ AMI. We tried two approaches as follows:

(a) We use 8kHz, and 16kHz AMI as parallel data and the teacher model is trained on 8kHz SWBD. The student model is trained using weighted objectives as sequence-KL using lattice posterior extracted for 8kHz AMI using the teacher network and LF-MMI using 16kHz AMI.

(b) In the second approach, we first use the T-S approach to transfer 8kHz SWBD to 16kHz SWBD. Then, we use the 16kHz network as a teacher to extract lattice posterior for 16kHz AMI.

7.6 Transfer learning in environment mismatch

In this section, we discuss experiments showing transfer learning from Librispeech to AMI in IHM and SDM conditions. The results are in Table 7.5. The "baseline" row of the results in this table is trained with only the 80 hours target datasets. The i-vector in all experiments in Table 7.5 was extracted using i-vector extractor trained on Librispeech. The weight transfer model shows 1% absolute improvement in WER in the case of SDM condition, and 2% absolute improvement in WER in the case of IHM condition. This might suggest that having the source and target data from a similar environment condition (like Librispeech and AMI IHM) is better for the weight transfer scenario.

Table 7.5: WER results: Librispeech to AMI transfer

		WER(%)	
Target Data	System	dev	eval
AMI-SDM	Baseline	41.0	45.2
	Weight transfer	39.9	44.2
AMI-IHM	Baseline	22.2	22.4
	Weight transfer	20.6	20.5

7.7 Weight transfer vs. multi-task training

In this section, two transfer learning approaches are investigated for transferring information from the 300 hrs SWBD dataset to the AMI-IHM, AMI-SDM and WSJ datasets. The i-vector extractor in these experiments is trained on 25% of pooled speed-perturbed data across all datasets. The "baseline" system is trained only on the target dataset. As discussed in Section 7.5, downsampling the data degrades performance on the AMI dataset. So we report results on the 8kHz baseline for all the corpora. The results in Table 7.6 show good improvement over baseline using both weight transfer and multi-task training.

We also tried a multi-task approach, where data pooled from all three target datasets, and all layers except the last layer are shared across all datasets. This is reported in the multi-task-pool row and shows a slight improvement over multi-task using just source and target data in the cases of AMI-SDM and AMI-IHM.

Table 7.6: WER results: SWBD to AMI and WSJ transfer

	WER		Rel. WER(%)	
AMI-SDM	*dev*	*eval*	*dev*	*eval*
Baseline	45.3	50	-	-
Weight transfer	43.9	49.3	3.1	1.4
Multi-task	45	49.2	0.66	1.6
Multi-task-pool	44.9	49.6	0.9	0.8
AMI-IHM	*dev*	*eval*	*dev*	*eval*
Baseline	23.6	24.6	-	-
Weight transfer	22.7	23.2	3.8	5.7
Multi-task	22.4	22.7	5.1	7.7
Multi-task-pool	22.1	22.6	6.4	8.2
WSJ	*dev93*	*eval92*	*dev93*	*eval92*
Baseline	5.49	3.15	-	-
Weight transfer	5.32	2.84	3.1	9.8
Multi-task	4.8	2.57	12.5	18.5
Multi-task-pool	4.99	2.53	9.1	19.7

Multi-task-pool: Trained on pooled speed-perturbed SWBD, AMI-SDM, AMI-IHM, and WSJ datasets.

7.8 Transfer learning using different objectives

The state-of-the-art neural networks in ASR are trained with sequence-level objectives like LF-MMI [**?**]. Frame-level objectives used in model transfer such as using soft-targets [**?**, **?**] are not naively applicable to the LF-MMI objective as the neural network outputs are not frame-level posteriors. Regressing information too close to the output may not be applicable as the outputs are

not specifically trained for good frame-wise predictions. Furthermore, output nodes in the LF-MMI networks operate at a lower (one-third) frame rate.

Table 7.7 shows transfer learning results using frame-level cross entropy vs. the sequence-level LF-MMI objective. The i-vector extractor is trained on 25% of the combined data from all the datasets for all the experiments, and all datasets are down-sampled to $8kHz$. In the multi-task training experiments, the TDNN models are trained on pooled speed-perturbed datasets of SWBD, AMI-SDM, AMI-IHM, and WSJ using cross-entropy and LF-MMI objectives. In weight transfer experiments, both source and target models are trained using the same objective function i.e., both cross-entropy or both LF-MMI. SWBD is used as the source dataset for the weight transfer experiments. The results show that transfer learning is as useful for the LF-MMI objective as it is for frame-level cross-entropy objective.

Table 7.7: Transfer learning for frame-level CE vs. sequence-level LF-MMI objective

	Cross-Entropy		*LF-MMI*	
WSJ	*dev93*	*eval92*	*dev93*	*eval92*
Baseline	6.38	3.38	5.49	3.15
Multi-task*	5.85	3.47	4.99	2.53
AMI-IHM	*dev*	*eval*	*dev*	*eval*
Baseline	26	27.6	23.6	24.6
Multi-task*	23.7	25.1	22.1	22.6
Weight transfer	25.2	26.3	22.7	23.2

*: Trained on pooled speed-perturbed SWBD, AMI-SDM, AMI-IHM, and WSJ datasets.

Chapter 8

Conclusion and future work

8.1 Conclusion

In this thesis, we investigated learning new feature representations at different levels for ASR. In the first part of this thesis, we focused on learning features at the signal level. We investigated learning new features; the first was to learn pitch as a complementary feature to conventional features for ASR and the second was to exploit joint feature extraction to learn data-driven based sets of filter banks.

In the second part of this thesis, we focusd on learning new feature representations at intermediate and higher-levels using DNNs. This can be viewed as transferring feature representations learned on other tasks. We investigated 3 different transfer learning approaches in ASR in more detail.

We proposed a robust pitch tracking algorithm based on NCCF, that showed performance improvement compared to off-the-shelf pitch tracking methods. Also, it showed 6% absolute WER improvement on tonal and 2% on atonal languages (Figure 2.2).

In the next part, we investigated joint feature extraction in more detail using new time-domain and frequency-domain setups. We first presented time-domain CNN-TDNN based architecture for the raw waveform setup. This model contained "NiN" pooling that showed a 5-8% relative improvement compared to other pooling methods (Table 3.2). In addition, we proposed a new statistic pooling layer as an alternative to the i-vector adaptation method, that showed more improvement on a raw setup compared to a baseline MFCC setup (Table 3.6). Next, it showed competitive WER results with state-of-the-art networks based on conventional features (Table 3.6). We also introduced a new frequency-domain feature learning setup that included a new normalization block and short-range weight constraint. It showed consistent 1-7% relative WER improvement on various wideband and narrowband databases (Table 3.10). Finally, feature learning from the spectrogram at multiple scales with a different frequency and time resolution was investigated in more detail using a proposed frequency-domain feature learning setup. We demonstrated that the optimum number of scales in the multiscale frequency-domain setup is 2. In addition, the multi-scale 200Hz setup showed a 3-5% relative WER im-

provement over a 100 Hz system with a single scale in different datasets (Table 3.15). Combining a low-frequency sub-band from the larger scale and a high-frequency sub-band from the smaller scale showed similar improvement with less computation (Table 3.14).

We extended this work to other speech tasks and investigated the effect of direct-from-signal learning in the emotion task. The time-domain joint feature learning showed an 8% WA improvement and 4.4% UA improvement in emotion identification (Table 4.2).

We also designed 3 new sets of features for DNN based ASR: DNN-c, fDNN-c and modified Mel filter banks, based on the learned filter banks in the filter bank layer in frequency-domain setup for different datasets. DNN-c features exploited a new invertible warping function for center frequency approximation and the bandwidth was approximated using the piece-wise linear function. Some filters learned in the filter bank layer contain multiple peaks. In fDNN-c feature, GMM is used to approximate the center frequency in warping function approximation. Also, the filter bandwidth is approximated based on the overlap between adjacent filters. We also design a modified version of Mel filter banks. The warping function was modified to show better center frequency approximation based on filters learned in the filter bank layer in different datasets. The bandwidth was also computed using the combination of the overlap-based bandwidth and piece-wise linear function. The new

proposed features, especially modified versions of Mel filter banks, showed consistent WER improvements ranging from $0.2 - 0.4\%$ on various clean and noisy databases, compared to Mel filter bank features (Tables 6.5 and 6.7).

Finally, we investigated weight transfer and multi-task training in ASR using a sequence-trained neural network based on LF-MMI in different acoustic conditions. We demonstrated that multi-task learning performs better than weight transfer, but weight transfer is preferable since it does not require retraining on the pooled data. We showed that single-stage training is better than two-stage training. Also, training the source model for a long time in weight transfer was not required. A single epoch of training on the source domain was enough and the results on the target domain were not sensitive to the power of the source domain's model. In addition, we showed that the weight transfer performance depends on the speaker adaptation approach. The best performance in i-vector based speaker adaptation was achieved by training the i-vector extractor on a combined source and target data. Also, we showed that both methods are equally applicable to sequence-level objectives like LF-MMI as frame-level CE objectives. Moreover, we examined two approaches in sampling rate mismatch condition and investigated sequence-level T-S learning in this condition.

8.2 Future work

In this thesis, we probed feature learning for supervised training and how the learned features are applied to supervised training problems in ASR. It is possible to learn unsupervised feature embedding, where the objective is to separate similar vs. different acoustic signal pairs. Two acoustic signals are defined to be similar, if they are produced by applying different transformations such as adding noise and reverberation to the same acoustic signal, otherwise they are defined as different. This can result in learning a new feature embedding that is invariant to different noise conditions. Similar approaches have recently shown good results for text-dependent and independent speaker identification tasks [**?**]. However, it would be important to scale these methods to large problems, since they need to compute pairwise statistics between different data points and the time complexity for these algorithms is $O(n^2)$, where n is the size of training data.

Another important property for features is the invariance to speakers. Speaker adaptation often results in large gains in the accuracy of ASR systems. In Subsection 3.1.5, we proposed a new statistic pooling layer, which aggregates the statistics over a moving window of up to 200 frames (2 sec), captures long-term effects in the signal and helps to normalize over some local variances. However, it does not attempt to encourage the invariance over whole utterances of

speakers. Learning new feature embedding with this invariance for ASR can be a new direction for future attempts.

Another important application for joint feature extraction is multi-channel ASR systems. The phase information is necessary for multi-channel processing, where the phase preserves relative delay of the speech at each microphone. The future research is to automatically learn the optimal features in the multi-channel scenario without any signal processing based beamforming approaches.

As shown in Section 3.3, joint feature learning from the spectrogram using multiple scales with a different time resolution showed a nice improvement. To simplify the setup, the features learned at each scale can be investigated in more detail and the improvement can be extended to new modified Mel filter bank features using multiple scales and frame rates.

Bibliography

[1] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks-studies on speech recognition tasks," *arXiv preprint arXiv:1301.3605*, 2013.

[2] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3377–3381.

[3] 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.

[4] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for lvcsr," in *Automatic*

Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013, pp. 315–320.

[5] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*. IEEE, 2013, pp. 8614–8618.

[6] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[7] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust asr," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[8] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[9] R. Schluter, I. Bezrukov, H. Wagner, and H. Ney, "Gammatone features and feature combination for large vocabulary speech recognition," in

Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on, vol. 4. IEEE, 2007, pp. IV–649.

[10] S. Duanmu, *The phonology of standard Chinese*. Oxford University Press, 2007.

[11] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2494–2498.

[12] 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.

[13] N. Jaitly and G. Hinton, "Learning a better representation of speech soundwaves using restricted boltzmann machines," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5884–5887.

[14] D. Palaz, R. Collobert, and M. M. Doss, "Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks," 2013.

[15] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Acoustic modeling with deep neural networks using raw time signal for lvcsr." in *Proc. Interspeech*, 2014.

[16] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4624–4628.

[17] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 215–219.

[18] P. Ghahremani, V. Manohar, D. Povey, and S. Khudanpur, "Acoustic modelling from the signal domain using cnns." in *INTERSPEECH*, 2016, pp. 3434–3438.

[19] E. Variani, T. N. Sainath, I. Shafran, and M. Bacchiani, "Complex linear projection (clp): A discriminative approach to joint feature extraction and acoustic modeling." in *INTERSPEECH*, 2016, pp. 808–812.

[20] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in

Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013, pp. 297–302.

[21] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. Saon, and B. Ramabhadran, "Improvements to filterbank and delta learning within a deep neural network framework," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6839–6843.

[22] S. Mirsamadi, E. Barsoum, and C. Zhang, "Automatic speech emotion recognition using recurrent neural networks with local attention," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2227–2231.

[23] J. Trmal, M. Wiesner, V. Peddinti, X. Zhang, P. Ghahremani, Y. Wang, V. Manohar, H. Xu, D. Povey, and S. Khudanpur, "The kaldi openkws system: Improving low resource keyword search," *Proc. Interspeech 2017*, pp. 3597–3601, 2017.

[24] V. Manohar, P. Ghahremani, D. Povey, and S. Khudanpur, "A teacher-student learning approach for unsupervised domain adaptation of sequence-trained asr models," in *Spoken Language Technology Workshop (SLT), 2018 IEEE*. IEEE, 2018, pp. 366–369.

[25] M. Sarma, P. Ghahremani, D. Povey, N. K. Goel, K. K. Sarma, and N. De-

hak, "Emotion identification from raw speech signals using dnns," *Proc. Interspeech 2018*, pp. 3097–3101, 2018.

[26] P. Ghahremani, H. Hadian, H. Lv, D. Povey, and S. Khudanpur, "Acoustic modeling from frequency-domain representations of speech." in *INTERSPEECH*, 2018, pp. 3434–3438.

[27] P. Ghahremani, V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Investigation of transfer learning for asr using lf-mmi trained neural networks," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 279–286.

[28] D. Talkin, "A robust algorithm for pitch tracking (rapt)," vol. 495. New York: Elsevier, 1995, p. 518.

[29] B. S. Lee, "Noise robust pitch tracking by subband autocorrelation classification," Ph.D. dissertation, Columbia University, 2012.

[30] D. Povey, A. Ghoshal *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. ASRU*, 2011.

[31] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," vol. 111, 2002, p. 1917.

[32] D. P. Ellis and B. S. Lee, "Noise robust pitch tracking by subband auto-

correlation classification," in *13th Annual Conference of the International Speech Communication Association*, 2012.

[33] M. Wu, D. Wang, and G. Brown, "A multipitch tracking algorithm for noisy speech," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 3, pp. 229–241, 2003.

[34] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[35] K. Kasi and S. A. Zahorian, "Yet another algorithm for pitch tracking," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1. IEEE, 2002, pp. I–361.

[36] F. Plante, G. F. Meyer, and W. A. Ainsworth., "A pitch extraction reference database," in *Eurospeech*, 1995, pp. 837–840.

[37] X. Lei, "Modeling lexical tones for mandarin large vocabulary continuous speech recognition," Ph.D. dissertation, University of Washington, 2006.

[38] G. Chen, S. Khudanpur, D. Povey, J. Trmal, D. Yarowsky, and O. Yilmaz, "Quantifying the value of pronunciation lexicons for keyword search in lowresource languages," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 8560–8564.

BIBLIOGRAPHY

[39] J. G. Fiscus, J. Ajot, J. S. Garofalo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *ICSLP*, 2007.

[40] K. Laskowski, M. Heldner, and J. Edlund, "The fundamental frequency variation spectrum," in *FONETIK*, 2008.

[41] P. Mermelstein, ", psychological and instrumental," vol. 116. New York: Academic, 1976, pp. 374–388.

[42] H. Hermansky, "Perceptual linear predictive (plp) analysis of speech," vol. 87, 1990, pp. 1738–1752.

[43] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, and B. Ramabhadran, "Learning filter banks within a deep neural network framework," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 297–302.

[44] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform cldnns," in *Proc. Interspeech*, 2015.

[45] M. Bhargava and R. Rose, "Architectures for deep neural network based acoustic models defined over windowed speech waveforms," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[46] D. Palaz, R. Collobert *et al.*, "Analysis of cnn-based speech recognition system using raw speech as input," in *Proceedings of Interspeech*, no. EPFL-CONF-210029, 2015.

[47] 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.

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

[49] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," 2016.

[50] J. J. Godfrey *et al.*, "Switchboard: Telephone speech corpus for research and development," in *ICASSP*, 1992.

[51] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013.

[52] M. J. F. Gales and P. C. Woodland, "Mean and Variance Adaptation

Within the MLLR Framework," *Computer Speech and Language*, vol. 10, pp. 249–264, 1996.

[53] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," vol. 19, no. 4. IEEE, 2011, pp. 788–798.

[54] 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.

[55] V. Peddinti, G. Chen, V. Manohar, T. Ko, D. Povey, and S. Khudanpur, "Jhu aspire system: Robust lvcsr with tdnns, i-vector adaptation, and rnn-lms," in *ASRU*, 2015.

[56] 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," 2015.

[57] J. Andén and S. Mallat, "Deep scattering spectrum," vol. 62, no. 16. IEEE, 2014, pp. 4114–4128.

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

[59] R. G. Vaughan, N. L. Scott, and D. R. White, "The theory of bandpass

sampling," *IEEE Transactions on signal processing*, vol. 39, no. 9, pp. 1973–1984, 1991.

[60] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[61] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving Deep Neural Network Acoustic Models using Generalized Maxout Networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 215–219.

[62] 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.

[63] ——, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of INTERSPEECH*, 2015.

[64] V. Tinto, "Dropout from higher education: A theoretical synthesis of recent research," *Review of educational research*, vol. 45, no. 1, pp. 89–125, 1975.

[65] 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.

[66] D. Povey, A. Ghoshal *et al.*, "The Kaldi Speech Recognition Toolkit," in *Proc. ASRU*, 2011.

[67] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[68] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[69] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," in *Readings in Speech Recognition*. Elsevier, 1990, pp. 393–404.

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

[71] 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, p. 100.

[72] 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.

[73] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5206–5210.

[74] S. Cheung and J. S. Lim, "Combined multiresolution (wide-band/narrow-band) spectrogram," *IEEE Transactions on signal processing*, vol. 40, no. 4, pp. 975–977, 1992.

[75] P. Ghahremani, H. Hadian, H. Lv, D. Povey, and S. Khudanpur, "Acoustic modeling from frequency domain representations of speech," *Proc. Interspeech 2018*, pp. 1596–1600, 2018.

[76] Z. Zhu, J. H. Engel, and A. Hannun, "Learning multiscale features directly from waveforms," *arXiv preprint arXiv:1603.09509*, 2016.

[77] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[78] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout, "Multi-scale deep learning for gesture detection and localization," in *Workshop at the European conference on computer vision*. Springer, 2014, pp. 474–490.

[79] T. Sivanagaraja, M. K. Ho, A. W. Khong, and Y. Wang, "End-to-end speech emotion recognition using multi-scale convolution networks," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017*. IEEE, 2017, pp. 189–192.

[80] C. Chan, P. Ching, and T. Lee, "Noisy speech recognition using de-noised multiresolution analysis acoustic features," *The Journal of the Acoustical Society of America*, vol. 110, no. 5, pp. 2567–2574, 2001.

[81] C.-P. Chan, Y. W. Wong, T. Lee, and P.-C. Ching, "Two-dimensional multiresolution analysis of speech signals and its application to speech recognition," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 405–408.

[82] V. Peddinti, T. Sainath, S. Maymon, B. Ramabhadran, D. Nahamoo, and V. Goel, "Deep scattering spectrum with deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 210–214.

[83] M. Athineos and D. P. Ellis, "Frequency-domain linear prediction for tem-

poral features," in *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*. IEEE, 2003, pp. 261–266.

[84] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 1. IEEE, 1992, pp. 517–520.

[85] J. Niu, Y. Qian, and K. Yu, "Acoustic emotion recognition using deep neural network," in *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*. IEEE, 2014, pp. 128–132.

[86] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Fifteenth annual conference of the international speech communication association*, 2014.

[87] J. Lee and I. Tashev, "High-level feature representation using recurrent neural network for speech emotion recognition," 2015.

[88] M. Neumann and N. T. Vu, "Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech," *arXiv preprint arXiv:1706.00612*, 2017.

[89] D. Palaz, R. Collobert *et al.*, "Analysis of cnn-based speech recognition

system using raw speech as input," in *Proceedings of INTERSPEECH*, no. EPFL-CONF-210029, 2015.

[90] N. Jaitly and G. Hinton, "Learning a better representation of speech soundwaves using restricted boltzmann machines," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5884–5887.

[91] G. Trigeorgis, F. Ringeval, R. Brueckner, E. Marchi, M. A. Nicolaou, B. Schuller, and S. Zafeiriou, "Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5200–5204.

[92] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[93] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for asr," in *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on. IEEE*, 2018.

[94] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural

network embeddings for text-independent speaker verification," in *Proc. Interspeech*, 2017, pp. 999–1003.

[95] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," *Submitted to ICASSP*, 2018.

[96] C. Busso, M. Bulut, C.-C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J. N. Chang, S. Lee, and S. S. Narayanan, "Iemocap: Interactive emotional dyadic motion capture database," *Language resources and evaluation*, vol. 42, no. 4, p. 335, 2008.

[97] M. Tahon and L. Devillers, "Towards a small set of robust acoustic features for emotion recognition: challenges," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 24, no. 1, pp. 16–28, 2016.

[98] R. Banse and K. R. Scherer, "Acoustic profiles in vocal emotion expression." *Journal of personality and social psychology*, vol. 70, no. 3, p. 614, 1996.

[99] R. Tato, R. Santos, R. Kompe, and J. M. Pardo, "Emotional space improves emotion recognition," in *Seventh International Conference on Spoken Language Processing*, 2002.

[100] C. M. Lee, S. Narayanan, and R. Pieraccini, "Recognition of negative emotions from the speech signal," in *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*. IEEE, 2001, pp. 240–243.

[101] D. Litman and K. Forbes, "Recognizing emotions from student speech in tutoring dialogues," in *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*. IEEE, 2003, pp. 25–30.

[102] K. R. Scherer, "Vocal communication of emotion: A review of research paradigms," *Speech communication*, vol. 40, no. 1-2, pp. 227–256, 2003.

[103] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition." in *INTERSPEECH*, 2015, pp. 3586–3589.

[104] V. Peddinti, Y. Wang, D. Povey, and S. Khudanpur, "Low latency acoustic modeling using temporal convolution and lstms," *IEEE Signal Processing Letters*, vol. 25, no. 3, pp. 373–377, 2018.

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

[106] T. N. Trong, V. Hautamäki, and K. A. Lee, "Deep language: a comprehensive deep learning approach to end-to-end language recognition," in *Odyssey: the Speaker and Language Recognition Workshop*, 2016.

[107] H. Hermansky, D. P. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional hmm systems," in *icassp*. IEEE, 2000, pp. 1635–1638.

[108] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," vol. 37, no. 3. IEEE, 1989, pp. 328–339.

[109] 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.

[110] K. M. Knill, M. J. Gales, A. Ragni, and S. P. Rath, "Language independent and unsupervised acoustic models for speech recognition and keyword spotting," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

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

[112] S. Strassel and J. Tracey, "Lorelei language packs: Data, tools, and resources for technology development in low resource languages." in *LREC*, 2016.

[113] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measure-

ment of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[114] E. Zwicker, "Subdivision of the audible frequency range into critical bands (frequenzgruppen)," *The Journal of the Acoustical Society of America*, vol. 33, no. 2, pp. 248–248, 1961.

[115] K. Paliwal, B. Shannon, J. Lyons, and K. Wójcicki, "Speech-signal-based frequency warping," *IEEE signal processing letters*, vol. 16, no. 4, pp. 319–322, 2009.

[116] L. Burget and H. Heřmanskỳ, "Data driven design of filter bank for speech recognition," in *International Conference on Text, Speech and Dialogue*. Springer, 2001, pp. 299–304.

[117] A. Biem and S. Katagiri, "Filter bank design based on discriminative feature extraction," in *icassp*. IEEE, 1994, pp. 485–488.

[118] Z. Hu and E. Barnard, "Smoothness analysis for trajectory features," in *icassp*. IEEE, 1997, p. 979.

[119] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," 2018.

[120] M. Harper, "The automatic speech recogition in reverberant environ-

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

[121] S. J. Pan and Q. Yang, "A survey on transfer learning," vol. 22, no. 10. IEEE, 2010, pp. 1345–1359.

[122] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: a survey," vol. 80. Elsevier, 2015, pp. 14–23.

[123] Y. Bengio *et al.*, "Deep learning of representations for unsupervised and transfer learning." vol. 27, 2012, pp. 17–36.

[124] Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. M. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache *et al.*, "Deep learners benefit more from out-of-distribution examples." in *AISTATS*, 2011, pp. 164–172.

[125] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[126] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," vol. 18, no. 7. MIT Press, 2006, pp. 1527–1554.

[127] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Ben-

gio, "Why does unsupervised pre-training help deep learning?" vol. 11, no. Feb, 2010, pp. 625–660.

[128] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[129] P. Swietojanski, A. Ghoshal, and S. Renals, "Unsupervised cross-lingual knowledge transfer in dnn-based lvcsr," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 246–251.

[130] S. Thomas, M. L. Seltzer, K. Church, and H. Hermansky, "Deep neural network features and semi-supervised training for low resource speech recognition," in *Proc. ICASSP*, May 2013, pp. 6704–6708.

[131] K. Veselỳ, M. Karafiát, F. Grézl, M. Janda, and E. Egorova, "The language-independent bottleneck features," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 336–341.

[132] X. Zhang, D. Povey, and S. Khudanpur, "A diversity-penalizing ensemble training method for deep learning." in *INTERSPEECH*, 2015, pp. 3590–3594.

[133] Z. Tang, D. Wang, and Z. Zhang, "Recurrent neural network training with dark knowledge transfer," in *Acoustics, Speech and Signal Process-*

ing (ICASSP), 2016 IEEE International Conference on. IEEE, 2016, pp. 5900–5904.

[134] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[135] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks." in *ICML*, 2015, pp. 97–105.

[136] D. Wang and T. F. Zheng, "Transfer learning for speech and language processing," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*. IEEE, 2015, pp. 1225–1237.

[137] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid hmm-ann continuous speech recognition system." International Speech Communication Association, 1995.

[138] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, 2012, pp. 366–369.

[139] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. De Mori, "Linear hid-

den transformations for adaptation of hybrid ann/hmm models," vol. 49, no. 10. Elsevier, 2007, pp. 827–835.

[140] Z. Huang, J. Li, S. M. Siniscalchi, I.-F. Chen, J. Wu, and C.-H. Lee, "Rapid adaptation for deep neural networks through multi-task learning," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[141] P. Swietojanski, J. Li, and S. Renals, "Learning hidden unit contributions for unsupervised acoustic model adaptation," vol. 24, no. 8. IEEE, 2016, pp. 1450–1463.

[142] Z. Huang, S. M. Siniscalchi, and C.-H. Lee, "A unified approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition," vol. 218. Elsevier, 2016, pp. 448–459.

[143] G. Heigold, V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean, "Multilingual acoustic models using distributed deep neural networks," in *Proc. ICASSP*. IEEE, 2013, pp. 8619–8623.

[144] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *Proc. ICASSP*. IEEE, 2013, pp. 7304–7308.

[145] F. Grézl, E. Egorova, and M. Karafiát, "Study of large data resources for multilingual training and system porting," vol. 81. Elsevier, 2016, pp. 15–22.

[146] R. Sahraeian and D. Van Compernolle, "Using weighted model averaging in distributed multilingual dnns to improve low resource asr," vol. 81. Elsevier, 2016, pp. 152–158.

[147] M. Karafiat, L. Burget, P. Matejka, O. Glembek, and J. Cernocky, in *Proc. ASRU*. IEEE, Dec., pp. 152–157.

[148] 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, Apr 1986, pp. 49–52.

[149] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. ACM, 2006, pp. 369–376.

[150] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*. IEEE, 2015, pp. 5206–5210.

[151] 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.

[152] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[153] V. Manohar, D. Povey, and S. Khudanpur, "JHU Kaldi System for Arabic MGB-3 ASR Challenge using Diarization, Audio-Transcript alignment and Transfer learning," in *Automatic Speech Recognition and Understanding (ASRU), 2017 IEEE Workshop on*, 2017.

[154] J. Li, M. L. Seltzer, X. Wang, R. Zhao, and Y. Gong, "Large-scale domain adaptation via teacher-student learning," *arXiv preprint arXiv:1708.05466*, 2017.

[155] J. Wong and M. Gales, "Sequence student-teacher training of deep neural networks," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 8, 2016, pp. 2761–2765.

[156] N. Kanda, Y. Fujita, and K. Nagamatsu, "Investigation of lattice-free maximum mutual information-based acoustic models with sequence-

level kullback-leibler divergence," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2017, pp. 69–76.

[157] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Semisupervised training of acoustic models using lattice-free mmi," in *ICASSP*, 2018.

[158] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7893–7897.

[159] Y. Long, Y. Li, H. Ye, and H. Mao, "Domain adaptation of lattice-free mmi based tdnn models for speech recognition," *International Journal of Speech Technology*, vol. 20, no. 1, pp. 171–178, 2017.

[160] V. Manohar, D. Povey, and S. Khudanpur, "JHU Kaldi System for Arabic MGB-3 ASR Challenge using Diarization, Audio-Transcript alignment and Transfer learning," in *Proc. ASRU 2017*, 2017.

[161] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size dnn with output-distribution-based criteria," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[162] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[163] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 165–170.

Vita

Pegah Ghahremani received her Master of Technology by research degree in Electrical and Computer Engineering from Johns Hopkins University in 2014. She completed her PhD. in Electrical and Computer Engineering with Dan Povey and Sanjeev Khudanpur while being affiliated to the Center for Language and Speech Processing (CLSP) at Johns Hopkins University in 2019. She contributes to the acoustic modeling code in Kaldi project. She is currently a Research Scientist at Amazon, Seattle, USA. Her research interests include maachine learning, automatic speech recognition, speaker recognition, and natural language understanding. In the past, she also had intership at Microsoft research (MSR).