# A Practical and Efficient Multistream Framework for Noise Robust Speech Recognition

by

Sri Harish Mallidi

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

Baltimore, Maryland

February, 2018

# Abstract

Performance of an automatic speech recognition system degrade rapidly in presence of a mismatch between training and test acoustic conditions. Usually, the mismatch affects different regions in feature space differently. Significantly lower word error rates can be obtained when regions with high signal-to-noise-ratio are given a high weight, while other regions are deemphasized. This is the fundamental motivation of multi-stream systems. In this thesis, a framework for practical multi-stream architectures is presented.

Past multi-stream systems employ a two stage training procedure. In these systems, stream specific networks are trained in the first stage and a large number of fusion networks are trained in the second stage. Typically, the number of fusion networks required are exponen-

tial to number of streams. For example a 5-stream system requires 31 ($2^5 - 1$) networks.

In this thesis, we first present training technique which eliminates the need for the two stage procedure. The proposed technique also helps in replacing multiple fusion networks with a single network. This not only reduces computational cost during training, but also results in a more robust system. Performance comparison in various noise robust tasks resulted in a word-error-rate reduction of 5 – 10 % relative over baseline models.

During test time, stream combination that produces the lowest error rate needs to be determined. This has to be done both accurately and efficiently. We proposed two performance monitor techniques to accurately determine the streams which are least affected by mismatches. For an efficient testing, a fast algorithm to search for the best stream combination is introduced with a reduction of test time latency by a factor of 20.

**Thesis Committee:**

Prof. Shinji Watanabe, Prof. Najim Dehak (Reader), and

Prof. Hynek Hermansky (Reader and Advisor)

ABSTRACT

# Acknowledgments

Foremost, I am immensely grateful to my advisor Prof. Hynek Hermansky for his guidance and support during my graduate life. During this time, he had provided me freedom to pursue multiple research threads and collaborate with many researchers and allowed me to work on several projects.

My PhD life started by working with lab mates Sriram, Samuel, Sivaram and Vijay. I am really thankful to them for helping me in the initial years of my PhD and continued friendship.

A significant part of this research is done during my stay at Brno University of Technology. I would like to thank my colleagues Karel, Tetsuji, Martin, Santosh, Lukas, Olda, Pavel, Franta at BUT for the fruitful interactions. Also, I would like to thank BBN (John, Tim, Stavros, Spyros, Bing) for providing me an opportunity to work on several interesting

and challenging DARPA, IARPA projects.

I would like to thank CLSP and its members for providing great infrastructure and conducive environment for pursuing research. Specifically, Sanjeev for being cheerful and supportive to all the graduate students, Dan for improving CLSP cluster and Ruth for taking care of all administrative paper work. Also, I would like to thank friends/colleagues at CLSP: Ruizhi, Bernd, Pegah, Keith, Feipeng. In addition, I would like to thank Prof. B. Yegnanarayana from IIIT Hyderabad for introducing me to speech processing.

Special Thanks to my friends Bhaskar, Ashwin and Phani with whom I shared awesome cooking, and movie watching sessions.

The thesis would not have completed without the support of family. I am infinitely indebted to them for being patient with me and putting their needs in the backseat.

Last but not least, I am grateful to the suggestions from my thesis committee members – Najim Dehak and Shinji Watanabe, which resulted in improving the thesis work significantly.

# Dedication

To my parents.

# Contents

# List of Tables

LIST OF TABLES

# List of Figures

# Chapter 1

# Introduction

The problem of automatic speech recognition is to convert a speech signal into sequence of words. Speech recognition has been used in applications like voice based dialing, telephone call routing, call centre analytics, etc. With the exponential growth in computing power, voice based interaction with machines is making in roads in a diverse set of applications. The examples include digital assistants like Amazon Echo or Google Home operating in living rooms, in-car speech recognition by Amazon Alexa, Google Now, Apple Siri, etc. These applications are exposing ASR technology to noises and acoustic mismatches which can have serious impact on accuracy of the system. This thesis focuses on improving robustness of automatic speech recognition systems to noises and acoustic mismatches.

A large number of methods have been proposed to make ASR systems robust to acoustic noises. These methods can be broadly grouped into feature-based and model-based methods. Feature-based methods focus on extracting features from input signal which are invariant to noises, while still retaining speech sound (e.g. phonemes) information. These techniques are usually motivated from human auditory processing or make certain assumptions about the noise type. Model-based approaches compensate for noise by adjusting acoustic model parameters. Usually, a subset of model parameters are re-estimated on noisy speech signal to increase the overall likelihood of the model. Similar to feature-based methods, these techniques also make some assumptions about the noise. While these techniques usually improve accuracy, they incur significant computational cost.

## 1.1 Focus of the thesis

In this thesis, we focus on multi-stream approach for noise robust ASR systems. Multi-stream speech recognition provides an intuitive way to improve robustness of ASR system to noises. In this framework, several streams of information are extracted from the speech signal, and these streams are adaptively fused to improve noise robustness. The fundamental motivation behind multi-stream recognition is, noise or environmental distortion is typically localized in

the signal space. If the streams which are less corrupted can be identified and decisions from these streams can be emphasized, substantial noise robustness can be achieved.

## 1.2 Thesis outline

This thesis is organized as follows:

In *chapter 2*, we provide an overview of automatic speech recognition system. A brief review of techniques which were proposed to improve noise robustness of ASR systems is presented. We then present a detailed discussion about multi-stream approach for noise robust ASR systems.

*Chapter 3* focuses on multi-band systems, which are an archetype of multi-stream approach for ASR. We present the proposed technique which is aimed to improve multi-band systems. An analysis of these techniques and comparison of new system with baseline models is also provided.

In *chapter 4*, we describe how the proposed technique can be used to build a multi-stream system. Application of the techniques is shown to result in multi-stream system which is easier to train and contains significantly lower number of parameters.

In *chapter 5*, two new performance monitor techniques are proposed, which are used to select noise robust regions of multi-stream system. We show

that proposed performance monitor methods improve accuracy. Techniques to reduce run time complexity are also presented in this section.

Finally, *chapter 6* summarizes the major results and contributions of this thesis and highlights some directions for future research.

# Chapter 2

# Overview of noise robust speech recognition

*This chapter introduces automatic speech recognition problem and its machinery. It also discusses previous works on improving noise robustness of speech recognition systems. Finally, the chapter is concluded by discussing the scope of the thesis.*

## 2.1 Automatic Speech Recognition

The task of automatic speech recognition (ASR) system is to convert an input speech waveform into text. In the most general sense, the problem of speech recognition is to estimate function $\mathbf{H}(\cdot)$ which transforms speech signal to text, concisely represented in the following equation

$$word\ sequence = \mathbf{H}(speech\ signal) \tag{2.1}$$

So the problem of building a speech recognizer is to estimate the function $\mathbf{H}(\cdot)$ . Modern ASR systems rely on statistical framework to convert speech to text. In this framework, the transformation function $\mathbf{H}(\cdot)$ has a probabilistic flavor associated with it. A more precise statement of speech recognition problem, in statistical framework, is given by the following equation [4, 5]:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathcal{G}}{\arg\max}\, p(\mathbf{w}|\mathbf{X}) \tag{2.2}$$

where $\mathbf{X} = \mathbf{x}_1, \cdots\cdots \mathbf{x}_T$ corresponds to the sequence of observation feature vectors extracted from the $T$ frames of the input speech signal, $\mathcal{G}$ is a set of all possible word sequences and $\widehat{\mathbf{w}}$ is the hypothesized word sequence.

In order to better understand eqn 2.2, consider the problem of isolated

word recognition with a finite vocabulary ($\mathcal{G}$). Then the task is to assign given speech signal ($\mathbf{X}$) to one of the words in set $\mathcal{G}$. Equation 2.2 suggests to choose the word ($\widehat{\mathrm{w}}$) which has maximum posterior probability. Under the assumption that distribution $p(\cdot)$ is "true" distribution, decision rule of eqn. 2.2 is guaranteed to minimize misclassification rate [6]. Continuous speech recognition, which involves identifying word *sequence*, is in principle an extension to isolated word recognition task with a significantly larger $\mathcal{G}$. Equation 2.2 is also referred to as *maximum a posterior* (MAP) rule for speech recognition, since we choose $\widehat{\mathrm{w}}$ as the one which has maximum posterior probability.

Inference of $p\left(\cdot|\mathbf{X}\right)$ is reversed using Bayes rule as follows:

$$\begin{aligned}\widehat{\mathbf{w}} &= \underset{\mathbf{w}\,\in\,\mathcal{G}}{\arg\max}\, p(\mathbf{w} \mid \mathbf{X}) \\ &= \underset{\mathbf{w}\,\in\,\mathcal{G}}{\arg\max}\, \frac{p(\mathbf{X} \mid \mathbf{w})\; p(\mathbf{w})}{p(\mathbf{X})} \\ &= \underset{\mathbf{w}\,\in\,\mathcal{G}}{\arg\max}\, p(\mathbf{X} \mid \mathbf{w})\; p(\mathbf{w}) \end{aligned} \tag{2.3}$$

The denominator of Bayes rule is ignored as it has no bearing on the $\arg\max$. Inference of distributions $p(\mathbf{X}|\mathrm{w})$ and $p(\mathrm{w})$ is much easier compared to inference of $p(\mathrm{w} \mid \mathbf{X})$. This is due to discrete nature of word sequences w compared to continuous nature of speech features $\mathbf{X}$.

### 2.1.1 Language modeling

In eqn 2.3, $p(\mathbf{w})$ models the word sequence $\mathbf{w} = w_1, w_2, \cdots$. This quantity is referred to as language model (LM). Using chain rule of probability $p(\mathbf{w})$ can be reformulated as

$$\begin{aligned} p(\mathbf{w}) &= p(w_1, w_2, \cdots w_n) \\ &= \prod_{i=1}^{n} p(w_i \mid w_1, \cdots, w_{i-1}) \end{aligned} \tag{2.4}$$

The term $p(w_i \mid w_1, \cdots, w_{i-1})$ represents probability of word $w_i$ given the history $w_1, \cdots, w_{i-1}$. Assuming the availability of enough text corpus, these probabilities can be reliably estimated by counting number of times $w_i$ occurred in the context of $w_1, \cdots, w_{i-1}$. However, having enough data to reliably estimate all possible conditional distributions is impossible. Language models which reduce the word history to n – 1 are referred to as n-gram language models. In n-gram models $p(w_i \mid w_1, \cdots, w_{i-1})$ is approximated as follows

$$p(w_i \mid w_1, \cdots, w_{i-1}) \approx p(w_i \mid w_{i-(n-1)}, \cdots, w_{i-1}) \tag{2.5}$$

Typically, 3-gram language models are used and even this case suffers from data sparsity issues. Widely used approaches to address the data sparsity issue

use models which "back-off" to lower order n-grams for infrequent contexts [7], and those which perform statistical smoothing of the n-gram scores [8, 9].

### 2.1.2 Acoustic modelling

The first term in eqn 2.3, $p(\mathbf{X} \mid \mathrm{w})$, is referred to as acoustic model (AM) component of speech recognition. Direct inference of $p(\mathbf{X} \mid \mathrm{w})$ is difficult as it involves modeling relationship between all possible acoustic sequences and word sequences. In many ASR systems, a word is expressed as a sequence of sub-word units (e.g. phonemes, tri-phones). Word models are then constructed by *glueing* its constituent sub-word unit models. Similarly, model for a word sequence is obtained by glueing word models. In this formulation, it is sufficient to model relationship between acoustics and sub-words units. Hidden Markov model (HMM) is the most widely used model for acoustic modeling in ASR. In HMM framework, the AM term can be written as follows:

$$p(\mathbf{X} \mid \mathbf{w}) = p(\mathbf{X} \mid s_1, s_2, \cdots, s_N) \tag{2.6}$$

$$= \sum_{\boldsymbol{\theta}(s_1:s_N)} \prod_{t=1}^{T} p(\mathbf{x}_t|\theta_t; \Lambda_{s_1:s_N})\; p(\theta_t|\theta_{t-1}; \Lambda_{s_1:s_N}) \tag{2.7}$$

where $s_1, s_2, \cdots, s_N$ is sub-word unit sequence corresponding to words $\mathbf{W}$, $\boldsymbol{\theta}(s_1 : s_N)$ are the possible state sequences of the HMM model defined by $\Lambda_{s_1:s_N}$. Note

that HMM, $\Lambda_{s_1:s_N}$ is obtained by joining HMM models of the sub-word units $s_1, s_2, \cdots, s_N$.

The observation distributions $p(\mathbf{x}_t|\theta_t)$ ($\Lambda$ is removed for convenience) are typically modeled using Gaussian mixture models (GMMs), parameterized by mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ for each $\theta_t$. Jointly, this is referred to as HMM–GMM acoustic model for speech recognition. More recently, deep neural networks (DNNs) are used to estimate the observation probabilities. In the DNN framework, a multi-layer perceptron is trained to classify states ($\theta_t$), given the input speech feature vector $\theta_t$. The trained neural network is used to compute the likelihood ($p(\mathbf{x}_t|\theta_t)$) as follows:

$$p(\mathbf{x}_t|\theta_t) \approx p(\theta_t|\mathbf{x}_t)/p(\theta_t) \tag{2.8}$$

where $p(\theta_t|\mathbf{x}_t)$ is softmax output of the DNN and $p(\theta_t)$ is prior probability of state, estimated from Viterbi alignments.

### 2.1.3 Search

Finding the best word sequence given observation sequence uses the MAP criterion shown in equation 2.3. A brute force application of equation involves first enumerating all possible word sequences. Likelihood of a word sequence generating the given observation sequence is computed by summa-

tion of probabilities of possible state sequences generating the observation sequence. This acoustic model term is then scaled with the prior of word sequence (given by language model term) and the word sequence with the maximum score is chosen as the MAP solution. The best word sequence is usually referred to as 1-best hypothesis. Only in simple ASR tasks, the above mentioned approach is feasible and in large vocabulary continuous speech recognition (LVCSR) more tractable solutions are used in practice. In order to simplify the search problem the following approximation is used:

$$\widehat{\mathbf{w}} = \underset{\mathbf{w} \in \mathcal{G}}{\arg\max}\, p(\mathbf{X} \mid \mathbf{w})\; p(\mathbf{w}) \tag{2.9}$$

$$= \underset{\mathbf{w} \in \mathcal{G}}{\arg\max} \left( \left[ \sum_{\boldsymbol{\theta}(s_1:s_N)} \prod_{t=1}^{T} p(\mathbf{x}_t|\theta_t)\; p(\theta_t|\theta_{t-1}) \right]\; p(\mathbf{w}) \right) \tag{2.10}$$

$$\approx \underset{\mathbf{w} \in \mathcal{G}}{\arg\max} \left( \left[ \max_{\boldsymbol{\theta}(s_1:s_N)} \prod_{t=1}^{T} p(\mathbf{x}_t|\theta_t)\; p(\theta_t|\theta_{t-1}) \right]\; p(\mathbf{w}) \right) \tag{2.11}$$

This is known as Viterbi approximation. The assumption here is that likelihood of the best path through the HMM network dominates summation of the above equation. Therefore it is adequate to approximate the summation term. The search problem is further simplified by employing hypothesis pruning techniques like beam pruning [10, 11].

## 2.2 Noise Robustness of ASR systems

Historically, applications of ASR technology include telephone call routing, voice based dialing, dictation systems in medical and legal domains etc. All these applications only allow users to interact with the machine in restrictive settings. That is, the technology used to work only in a limited-vocabulary scenarios, where the user needs to utter clear, well articulated speech and in noise-free office settings. With the availability of large datasets and computation power, ASR technology has advanced to the stage where more challenging applications are now possible. Digital voice assistants like Amazon Echo, Google Home, etc are examples of device which allow users to interact in noisy, far-field environments. Voice based interaction is also available in mobile phones (e.g. Siri on iPhone etc). In order to tackle the large-scale, real-world applications, ASR technology needs to be robust to varying acoustic environments.

Given the vast number of applications for ASR systems, it is imperative to make ASR technology more robust to acoustic variability. In the following section we provide mathematical formulation of what it means to be noise robust. To make the explanation easier, we refer to training acoustic conditions as "clean" and mismatched speech signal as "noisy". Consider the following example: Let $\mathbf{Y}$ denote noisy speech signal and $\mathbf{X}$ denote corresponding clean

version. The MAP equation to estimate $\widehat{\mathbf{w}}$ is given by

$$\hat{\mathbf{w}}_{\mathbf{Y}} = \underset{\mathbf{w} \in \mathcal{G}}{\arg\max}\, p(\mathbf{Y} \mid \mathbf{w})\; p(\mathbf{w}) \tag{2.12}$$

So for $\hat{\mathbf{w}}_{\mathbf{Y}}$ to be equal $\hat{\mathbf{w}}_{\mathbf{X}}$, we want the same acoustic model likelihoods $p(\mathbf{Y} \mid \mathbf{w}) = p(\mathbf{X} \mid \mathbf{w})$. This equation suggests that in order for an ASR system to produce the same word sequences for clean and noisy conditions, acoustic model needs to be robust to noise and acoustic mis-matches.

The approaches to noise robustness can be grouped into two categories: (i) feature space and (ii) model space.

### 2.2.1 Feature space approaches

Many approaches in this space are motivated by the noise robust ability of human speech recognition (HSR) and try to mimic properties of HSR. Even widely used features such as perceptual linear prediction (PLP) [12] and Mel frequency cepstral coefficient (MFCC) [13] features approximate mechanisms observed in human ear (cochlea). In the case of PLPs, Bark filter-banks, loudness equalization and cubic root compression are examples. In the case of MFCCs, Mel filter-banks approximate non-linear frequency resolution observed in cochlea. Gammatone features use Gammatone filter-banks, which are better at approximating filters observed in ear. Features like RASTA PLP [14], Gabor

features [15] and spectro-temporal features [16] are motivated by higher-order processing in human auditory cortex. Works [17–20] observed that temporal modulations above 25 Hz are not informative for perceiving human speech, which motivates the temporal filter of 1–12 Hz used in RASTA processing. Gabor and spectro-temporal features uses multiple spectral and temporal filters during the feature extraction stage. Several other techniques, which can also be categorized into feature space, attempt to enhance speech by subtracting the noise. Examples of these approaches are minimum mean square error [21], Weiner filtering [22, 23], etc. These techniques assume a noise model (additive or/and convolutive) and attempt to filter out the noise. The key difference between techniques in this group is type of noise model assumption and estimation. Other features such as zero crossing peak amplitude [24], average localized synchrony detection [25], perceptual minimum variance distortionless response [26] etc are motivated by empirical observation.

### 2.2.2 Model space approaches

The central idea in noise robust ASR is to create an acoustic model which is invariant to noise, i.e. $p_\Lambda(\mathbf{Y} \mid \mathbf{w}) = p_\Lambda(\mathbf{X} \mid \mathbf{w})$. Feature space approaches achieve this by extracting features such that $\mathbf{Y} \approx \mathbf{X}$. In contrast, model space approaches modify the parameters of the acoustic model directly.

Consider the case where reference transcription of test utterance is available. Then adapting the model to increase $p_\Lambda(\mathbf{Y} \mid \mathbf{w})$ will increase the accuracy. Since reference transcriptions are not available, most model adaptation methods consider a 2-pass approach: Decode the test utterance to obtain a 1-best hypothesis, and treat the 1-best hypothesis as true transcript and modify the model parameters. Most model space methods differ in re-estimation of model parameters. The popular approach is to use maximum likelihood estimation method [27, 28]. Discriminative methods have also been explored [29–31], but these tend to be sensitive to hypothesis errors [32]. Since the amount of data used for re-estimation step is small, only a subset of the parameters are modified. Widely used example of MLE approach is maximum likelihood linear regression (MLLR) [27] (and its variant feature-space maximum likelihood linear regression (fMLLR) [28]). Several other approaches like parallel model combination [33], model-domain vector taylor series [34], Stereo-based Piecewise LInear Compensation for Environments [35] etc. Most of these methods assume some knowledge about the noise at hand.

Compared to feature based approaches, these approaches typically achieve higher accuracy. However these are computationally expensive, as they involve decoding twice (one for transcription generation and one after parameter re-estimation to obtain final hypothesis), and running an extra expectation-

maximization step. These drawbacks make practical applicability of model based approaches challenging.

## 2.3 Noise robust approaches in DNN acoustic models

In a HMM – GMM acoustic model the observation probabilities ($p(\mathbf{x}_t|\theta_t)$) are obtained by a GMM, and the transitions ($p(\mathbf{x}_t|\theta_t)$) are modeled by an HMM. Unlike a GMM which is a generative model which model likelihoods directly, neural networks are discriminative models which can only model posterior probabilities ($p(\theta_t|\mathbf{x}_t)$). The network is trained to predict the posterior probabilities, which are converted to likelihoods using the following Bayes rule:

$$p(\mathbf{x}_t|\theta_t) \approx p(\theta_t|\mathbf{x}_t)/p(\theta_t) \tag{2.13}$$

where the priors $p(\theta_t)$ are obtained from training data. Plugging in the above terms in acoustic model equation

$$p(\mathbf{X} \mid \mathbf{w}) = \sum_{\boldsymbol{\theta}} \prod_{t=1}^{T} p(\mathbf{x}_t|\theta_t)\; p(\theta_t|\theta_{t-1}) \tag{2.14}$$

$$\approx \sum_{\boldsymbol{\theta}} \prod_{t=1}^{T} \frac{p(\theta_t|\mathbf{x}_t)}{p(\theta_t)}\; p(\theta_t|\theta_{t-1}) \tag{2.15}$$

where $q(\theta_t|\mathbf{x}_t)$ is posterior probability state, obtained from softmax output of the neural network. For a noisy signal $\mathbf{Y}$, the conditional likelihood becomes

$$p(\mathbf{Y} \mid \mathbf{w}) \approx \sum_{\boldsymbol{\theta}} \prod_{t=1}^{T} \frac{p(\theta_t|\mathbf{y}_t)}{p(\theta_t)}\, p(\theta_t|\theta_{t-1}; \Lambda) \tag{2.16}$$

From these equations, it is evident that for improving noise robustness of neural network based acoustic models, posterior probability obtained from noisy speech and clean speech should be as close as possible. That is, $p(\theta_t|\mathbf{y}_t) = p(\theta_t|\mathbf{x}_t)$.

Noise robust techniques in category 1 are directly applicable for neural network based models. In these cases, the assumption is that the extracted features are invariant to acoustic distortions (i.e. $\mathbf{Y} \approx \mathbf{X}$). Due to continuity of parameter space of neural networks, this in turn makes $p(\theta_t|\mathbf{y}_t) \approx p(\theta_t|\mathbf{x}_t)$. The success of techniques in these categories depend on how well we can extract features which are invariant to distortions. Application of techniques in category 2 is more challenging as little success is found in modifying the parameters to an unsupervised cost function. But techniques like feature space MLLR or cMLLR which modifies mean of GMM to maximize the likelihood is applicable. Modifying the mean via linear transform is equivalent to modifying the features. The adapted features can be treated as regular features and pro-

vided as input to neural network. These features are shown to be robust [36,37] to acoustic mismatches.

One of the most used approach is to provide channel or noise information as an additional input to the network. In [38] use iVector [39] along with filter-bank features to train the model. The assumption is an iVector capture channel information implicitly. Explicit specification of noise information as an additional input is also explored in [40]. However, most dominant approach to improve noise robust of DNN acoustic models is train on artificially corrupted speech. Several works have shown that robustness can be increased by just training the network on speech corrupted with artificial noises [37, 41]. The observed improvements can be attributed to ability of neural networks to ingest large amounts of highly variable data.

## 2.4 Scope of the thesis:

In this thesis, we use "*multi-stream*" approach to address noise robustness of ASR systems. Multi-stream speech recognition provides an intuitive way to improve robustness of ASR systems to acoustic mis-matches. In this framework, several streams of information are extracted from the speech signal, and these streams are adaptively fused to improve noise robustness. The fundamental motivation behind multi-stream recognition is that noise or en-

vironmental distortion effects only few streams. The streams which are less corrupted can be identified and decisions from these streams can be emphasized.

In previous sections, we provided a brief description of past noise robust techniques. Most of the techniques perform well only in few noise conditions, and there is no universally accepted noise robust technique which works on all possible acoustic distortions. For example, RASTA filtering is shown to be robust to linear, convolutive distortions observed in telephone channels [14], Weiner filtering is shown to be robust to additive noisy conditions [22] etc. Multi-stream approach can be used to solve this issue. For example, we can construct a 2 stream system with one stream being RASTA features and other stream being Weiner filtered features. If the test signal is corrupted by convolutive distortion we can use RASTA stream, and if it is corrupted by additive distortion we can use Weiner filtering stream. This 2 stream system has the potential of being robust to both convolutive and additive distortions.

Therefore, in order to build a successful multi-stream system one needs to deal with the following three issues [42]:

- **Formation of streams:** streams which are conditionally independent as much as possible, while retaining some cues for recognition of message.
- **Identification of streams:** accurate identification of streams that are

less corrupted for a given test signal.

- **Fusion of streams:** finding combination of streams which gives the best result.

In this thesis we propose techniques to improve stream fusion part of multi-stream system.

# Chapter 3

# Multi-band acoustic model

*In this chapter the proposed multi-band acoustic model is presented. The proposed neural network model combines sub-band specific architecture idea from past multi-band studies with dropout. Resulting model provides significant performance improvements in various noisy speech recognition tasks.*

## 3.1 Overview of past multi-band approaches

The basic idea of multi-band ASR can be summarized as follows:

- **Divide speech signal into several sub-bands:** Each band should focus on narrow frequency region of input speech signal. This ensures localization of noise to a few bands and independence of features from each band.
- **Extract acoustic features in each band:** Acoustic features which capture energy envelope are extracted in each the sub-bands. Most multi-band systems [43–46] implement this as follows: Short-time Fourier analysis of input speech signal is first performed with a frame size of 25 ms and frame shift of 10 ms. In each frame, Mel scale [13] or Bark scale [12, 47] filterbank energies are computed by frequency warping power spectrum of short-term signal. The filterbank energies are then grouped into 5 or 7 groups, where energies in each group correspond to a specific critical band. The energies in each group form acoustic features for the sub-band.
- **Train independent recognizers:** In each sub-band a recognizer is trained on acoustic features from that band. Since each sub-band only sees features from that sub-band, it is unaffected by noises which corrupt other bands. Studies [45, 46] used GMM-HMM based ASR systems in each sub-band. However, most studies focused on using ANN-HMM models [43, 44]. Since ANNs can deal with correlated feature representations, they make ideal candidates for sub-band recognizers. Several ANNs (one for each sub-band) are trained to predict the phoneme class of the current frame.

- **Combine decisions from the sub-band recognizers:** Accuracies obtained from individual sub-band recognizers are far from optimal as each system operates on partial information. Works [48] combine the information from the individual recognizers by averaging the output posterior probabilities of the neural networks. While averaging, the weights can also be non-uniform and test speech signal dependent as shown in [48]. A drawback of posterior fusion approaches is poor performance observed in clean and non-localized noisy conditions. This is due to sub-optimal (i.e. weighted average) combination of information in the sub-bands. A better approach, proposed by [43], is to concatenate posteriors of each band and the concatenated vector is used as input feature to train one more network to classify phoneme. This approach significantly improves the performance and has been employed in several other multi-band/multi-stream works [49].

Figure 3.1 shows an example multi-band architecture, which encapsulates all the points mentioned above. The frequency spectrum of input speech signal is divided into 5 sub-bands and Mel filterbank features are extracted in each band. A neural network is trained separately in each band resulting in five stage1 networks. Posteriors (or bottleneck features, TANDEM features) are extracted from each of the 5 networks (represented by $\mathbf{h}_1, \mathbf{h}_2, \cdots \mathbf{h}_5$ in fig. 3.1).

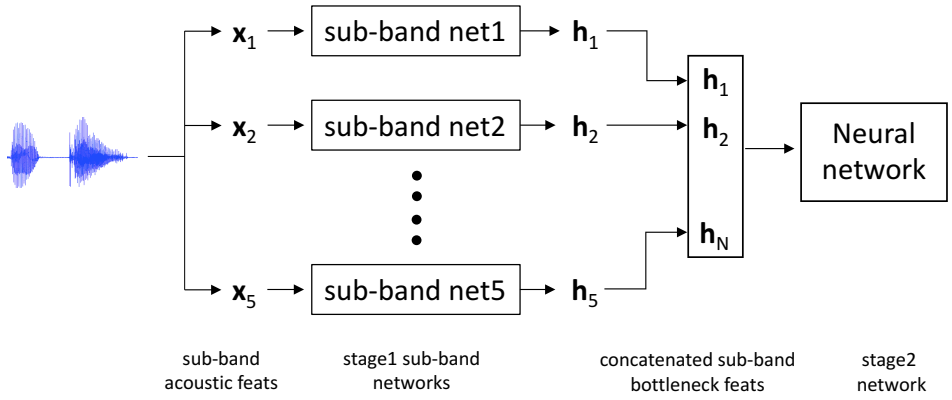


**Figure 3.1:** Block diagram of conventional multi-band acoustic model with 5 sub-bands. The system consists of 2 stages. Stage1 consists of several (5 in this case) independent networks, operating on each sub-band. Network in stage2 merges decisions from the first stage networks.

These features are concatenated to form input representation for stage2 network.

## 3.2 Proposed multi-band system new

This work proposes a new technique to train multi-band model shown in Fig. 3.1. In order to better explain the training technique, we refer $\mathbf{h}_1, \mathbf{h}_2, \cdots; \mathbf{h}_N$ as feature vectors of the $N$ sub-bands. For example, $\mathbf{h}_i$ is bottleneck vector of $i^{th}$ sub-band network and $h_i$'s can be of non-uniform dimensionality. Typically, for a standard multi-band system, the features are concatenated and given as input to the network. That is, the concatenated vector $\mathbf{h} = [\mathbf{h}_1; \mathbf{h}_2; \cdots, \mathbf{h}_N]$ is used as input to the network. However, for the stream-dropout training, we

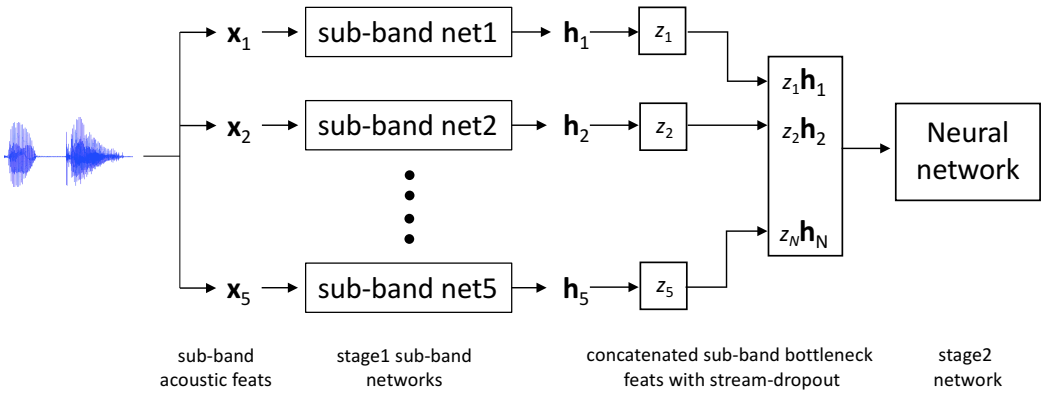


**Figure 3.2:** Schematic of proposed multi-band acoustic model. Stream-dropout is applied on features from each band and concatenated to form input to stage2.

concatenate the feature vectors as follows:

$$\widetilde{\mathbf{h}} = [z_1\mathbf{h}_1; z_2\mathbf{h}_2; \cdots ; z_N\mathbf{h}_N] \tag{3.1}$$

where each $z_i$ is a scalar and binary valued. $z_i$ is referred to as binary mask or simply mask. Fig. 3.2 illustrates the concatenation procedure described above. The concatenated feature vector is then presented as input to the network.

During training of the network, each mask ($z_i$, $i = 1 \cdots N$) is an independent Bernoulli random variable. That is $z_i$ can take value 0 or 1 with some probability $p$. Unless otherwise stated, we use $p = 0.5$ for all the masks. For $N = 2$, the network sees one of the following 3 input patterns in the training

stage:

$$\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{h}_2] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{0}; \mathbf{h}_2] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{0}]$$

whereas for $N = 3$, the network sees one of the following 7 input patterns:

$$\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{h}_2; \mathbf{h}_3] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{0}; \mathbf{h}_2; \mathbf{h}_3] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{0}; \mathbf{h}_3] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{0}; \mathbf{0}; \mathbf{h}_3] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{h}_2; \mathbf{0}] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{0}; \mathbf{h}_2; \mathbf{0}] \text{ or}$$

$$\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{0}; \mathbf{0}]$$

For a system with $N$-streams, number of possible patterns are $2^N - 1$.

During test, no dropout is applied. That is all the dropout masks ($z_i$'s) are set to 1. The network can be treated as a regular, non-dropout network.

## 3.3 Why stream-dropout helps?

Previous section described the proposed training algorithm for a multi-band system. In this section we analyze the effect of noise on this system to understand why stream-dropout improves noise robustness of multi-band system. Finally, we compare the performance of proposed multi-band system with baseline systems.

### 3.3.1 Multi-band setup:

**Acoustic features:** Short-term Fourier analysis of input speech signal is performed with a frame size of 25 ms and frame shift of 10 ms. Per frame, 63 Mel filterbank energies are extracted covering the frequency range of 0 – 8000 Hz. The 63 Mel bands are separated into 5 groups, with each group representing a sub-band. Each sub-band group covers 10, 10, 11, 12, and 21 Mel windows respectively. A 11 frame temporal context is used as feature representation in each band. The context features from Mel windows of the 5 groups form the input acoustic features for the 5 sub-band networks.

**Stage1 networks:** Each sub-band net consists of 2 hidden layers with 1500 ReLU units. This is followed by a bottleneck layer of size 40 and a final softmax layer. The 5 sub-band nets are trained independently of each other to optimize cross-entropy loss criterion. Targets for the cross-entropy loss are

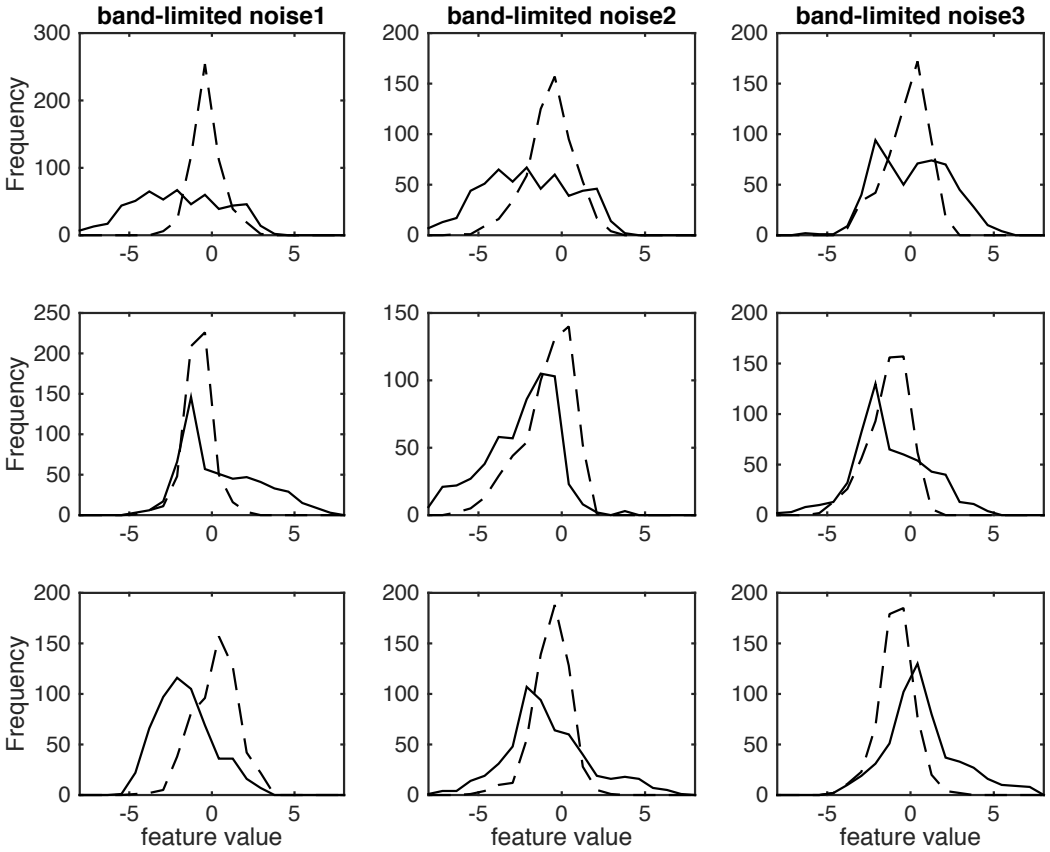


**Figure 3.3:** Comparison of sub-band feature values for a clean signal and its noisy versions. In each plot, solid ('—') line and dashed ('– –') line corresponds to **histograms** clean and noisy feature values, respectively. A column shows specific band-limited noise type and histograms of 3 different feature dimensions are shown in each noise type.

obtained from a GMM-HMM model. The GMM-HMM system is trained on speaker adapted MFCC features [36]. $\sim$2300 tri-phone PDF targets are generated from the GMM–HMM model which are used as targets for all the sub-band nets.

**Stage2 networks:** After training each sub-band network, the final softmax layer is removed. Bottleneck features are extracted from each of the sub-band net, and they are concatenated. The resulting 200 dim ($5 \times 40$) concatenated feature vector is used as input to stage2. The stage2 network consists of 4 hidden layers with 1500 ReLU units in each layer. Previous works [2, 44, 49] in multi-band transformed final softmax outputs and used them as features for stage2. We prefer to use bottleneck features here because, unlike the works [2,44,49], the stage1 networks in this work are trained on senone targets rather than phoneme targets. Senone targets has much higher dimensionality and requires LDA or PCA to reduce the dimensionality prior to concatenation. We observed that LDA or PCA based dimensionality reduction causes significant information loss, and bottleneck features are easier to model for stage2 networks.

### 3.3.2 Effect of noise on sub-band features

In order to analyze the behavior of proposed system, we created synthetic noisy test sets. These test sets are created by adding band-limited noises to clean test signals of Aurora4 test set. We chose band-limited noises, which corrupt low-frequency (500 – 875 Hz), mid-frequency (875 – 1375 Hz) and high-frequency (2000 – 3125 Hz) of input speech signal. Each signal of Aurora4 clean test set is corrupted by the 3 band-limited noises at 0 and 10 dB signal-to-noises. This resulted in 6 variants of original clean test set. We used multi-condition training set of Aurora4 to train the models. The training set consists of 14 hours of multi-condition data, sampled at 16 kHz.

Figure 3.3 shows histograms of several features from sub-band networks, for a clean signal, and its corresponding noisy versions. It can be seen from the figure that histograms of noise signal more centered around zero and have less dynamic range compared to the corresponding clean version. For example, in the first plot of the figure, clean feature value ranges from -8 to 5. Whereas, its noisy version ranges from -2 to 2. This shows that due to presence of noise, feature values towards zero. Since the network in stage2 encounters features with zero values in training, the noisy test utterance is no longer an unseen sample.

Also, this improves prediction accuracy of the network as seen in Fig-

ure 3.4. The figure shows WER results of baseline DNN, baseline multi-band and proposed multi-band models. The models are evaluated in clean and the 3 band-limited noisy conditions at 0 and 10 dB SNRs. It can be observed from the figure that proposed stream-dropout technique results in a model which consistently performs better than baseline multi-band. In some noises and SNRs, the proposed system results in more than 35 % relative reduction in WER. Note that the improvement is only due to stream-dropout training as all other variables (i.e. number of parameters, input features, training targets, training cost function etc) in model training are same.

It can also be observed from the figure that both baseline and proposed multi-band models perform significantly better than a fully connected DNN in all the 3 band-limited noisy conditions. The multi-band systems resulted in more 50 % relative reduction in WERs in the band-limited noises, illustrating that for localized noises multi-band systems can outperform traditional models. This shows when test noises match the multi-band assumption, significant performance improvements can be achieved.

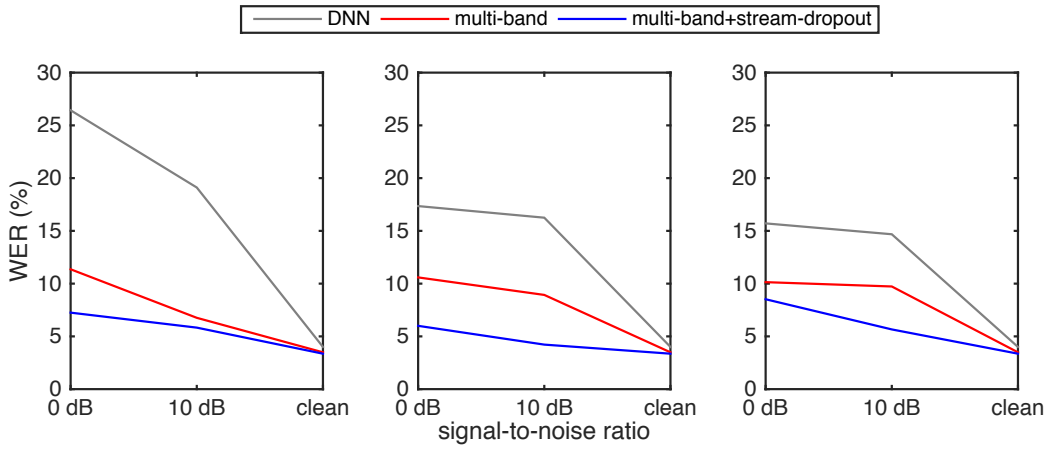


**Figure 3.4:** Comparison of WERs (%) of baseline DNN, baseline and proposed multi-band models. All the models are evaluated in clean test and band-limited noisy conditions.

## 3.4 Speech recognition experiments with real noises

In previous section, we showed that adding stream-dropout and joint-training techniques to multi-band network results in a better performing system in clean and band-limited noisy conditions. In this section, we evaluate the performance of multi-band system in realistic noises. This is to demonstrate that improvements observed in previous section is not limited to synthetic band-limited noises and can be observed even in more real world noises. We used Aurora4 and ASpIRE [50] speech recognition tasks.

**Table 3.1:** WER (%) results of baseline and proposed systems in Aurora4 test conditions.

| | DNN | Multi-band | |
|---|---|---|---|
| | | baseline | proposed |
| Clean Same Mic | | | |
| clean | 4.2 | 3.9 | 3.9 |
| Clean Diff. Mic | | | |
| clean | 11.7 | 9.6 | 6.3 |
| Additive Noise Same Mic | | | |
| airport | 6.7 | 5.7 | 5.0 |
| babble | 6.9 | 5.7 | 5.3 |
| car | 4.6 | 3.8 | 3.4 |
| restaurant | 10.0 | 8.6 | 7.3 |
| street | 10.2 | 7.6 | 6.9 |
| train | 10.1 | 8.3 | 7.5 |
| Avg. | 8.1 | 6.6 | 5.8 |
| Additive Noise Diff Mic | | | |
| airport | 21.3 | 16.2 | 14.0 |
| babble | 22.2 | 16.6 | 15.2 |
| car | 14.6 | 10.3 | 8.5 |
| restaurant | 23.6 | 19.2 | 17.4 |
| street | 24.6 | 19.7 | 18.6 |
| train | 24.5 | 21.1 | 18.0 |
| Avg. | 21.8 | 17.2 | 15.2 |
| Overall Avg. | 13.6 | 10.9 | 9.6 |

### 3.4.1 Aurora4

We use Aurora4 speech database for building the ASR models. The database is based on the DARPA Wall Street Journal (WSJ0) corpus which consist of clean recordings of read speech, with a medium vocabulary size of 5000 words. The training set consists of 14 hours of multi-condition data, sampled at 16 kHz. The 14 hours of data is comprised of 7137 utterance from 83 speakers.

Half of the utterances were recorded by the primary Sennheiser microphone and the other half were recorded using one of a number of different secondary microphones. Both halves include a combination of clean speech and speech corrupted by one of six different noises (street traffic, train station, car, babble, restaurant, airport) at 10-20 dB signal-to-noise ratio. The test set consist of 14 conditions, with 330 utterances for each condition. The conditions include clean set recorded with primary Sennheiser microphone, clean set with secondary microphone, 6 additive noise conditions which include airport, babble, car, restaurant, street and train noise at 5-15 dB signal-to-noise ratio (SNR) and 6 conditions with the combination of additive and channel noise. Note that both training and test sets have same noise conditions, but they differ in SNRs.

Table 3.1 shows performances of DNN, multi-band and multi-band+stream-dropout systems. It can be seen from the table that multi-band system (comparing columns 2 and 4) outperforms DNN system in all the conditions. We observe a 18–21 % relative reduction in WER in noisy conditions by using multi-band system instead of a DNN system. Across all conditions, using multi-band system resulted in WER which is 20 % (relative) lower than baseline DNN. We observed further improvements by using multi-band+stream-dropout system. It can be observed from the table that (comparing rows 3 and 4) proposed system improves performance in all acoustic conditions. Overall stream-dropout

reduces the WER from 10.9 % to 9.6 % (11 % relative reduction).

### 3.4.2 ASpIRE

The training data for ASpIRE task is based on English portion of Fisher corpus [51] (LDC2004S13, LDC2005S13). Multi-condition corpus was created by distorting original Fisher corpus, with real world room impulse responses (RIR) and noise recordings from 3 different databases, the Real World Computing Partnership (RWCP) sound scene database [52], the REVERB challenge database [53] and the Aachen impulse response database [54]. A total of 325 multi-channel recordings of RIRs were selected from the three databases. Three different copies of each recording in Fisher corpus were created by randomly sampling three different RIRs. This procedure resulted in 5500 hours of training data. A 1000 hours subset of this is selected to train the models. We evaluated the performance in *dev* set of 5 hours, which are provided as part of ASpIRE challenge [50]. *dev* set is collected by recording 10 minute telephone conversation between Native American English speakers, on a predefined topic. 15 microphones were installed in the room to collect far-field recording. More details of the *dev* set can be found in [50].

Convolutional neural networks have shown to provide significant performance improvements over fully connected networks in various vision and

speech tasks. The reason for the improvement is due to better feature detectors of convolution operation and translation invariance offered by pooling layers. CNNs used in speech recognition consists of 2-3 convolutional layers followed by fully connected layers. Unlike vision tasks, where a two-dimensional convolutional operation is used, CNNs in speech only employ frequency convolution. It has been hypothesized that frequency convolution, along with pooling layers offer robustness to pitch differences between speakers [55].

In this section, we explore the possibility of using CNN layers to obtain better sub-band features. In conventional CNNs, the number of filterbank features each convolutional layer sees in the range of 40-60. We cannot employ frequency convolution in multi-band architecture as the number of filterbanks are in the same range of filterwidth. Instead of frequency convolution, we propose to use time convolution in each band. The networks in each band consist of two convolutional layers, operating on sub-band filterbank features, with 21 frame context. Delta+acceleration features are also used with original filterbank features. The first CNN layer uses a width of 4 filterbanks and with a step size of 1. This is followed by a max-pooling layer, with pool size 2. The second CNN also employs filter of width 4 and step size of 1. The second CNN is followed by a projection layer, which reduces the dimensionality of features to 40. We apply a stream-dropout to each of these 40 bottleneck features and

**Table 3.2:** Comparison of WERs of various baseline models and proposed of multi-band model. The models are trained on 1000 hr Fisher corpus.

| | | ASpIRE *dev-set* |
|---|---|---|
| 1 | DNN | 43.6 |
| 2 | CNN (freq.) | 39.8 |
| 3 | CNN (time) | 41.8 |
| 4 | Multi-band | 38.9 |

concatenate them and provide as input to a network with 4 fully connected layers.

Table 3.2 shows the WERs obtained from multi-band system and baseline DNN and CNN models. Similar to Aurora4 experiments the DNN model consists of 6 fully connected hidden layers. Each layer consists of 1500 ReLU neurons. The DNN system is trained on 11 frame context features. The targets are obtained from alignments from a GMM – HMM system, which is trained on 5000 hours of speech data. Note that all the neural network models are trained on 1000 hours of speech, using frame-level cross-entropy criterion

CNN (freq.) model is a convolutional network which operates along the frequency axis. The network consists of 2 convolutional layers followed by 3 fully connected layers. First of the convolutional layer operate on 46 Mel filterbanks, with a frequency filter width of 8 points and step size of 1. This is followed by a max-pooling layer of 3. The filter width of second convolutional layer is 4 and step size is 1. Outputs of second hidden layer are provided as inputs a

3 layer fully connected network. In total the network is made up of 13 million parameters. In contrast convolution operation in CNN (time) happens along the time axis. Number of parameters of convolutional layers in CNN (time) are same as that of sub-band networks in multi-band networks. Output of the convolutional layers is provided as inputs 3 fully connected layers. The network consists of 17 M parameters. It can inferred from the table that CNN models perform better than regular DNN models. This could be due to sub-sampling layers in CNN, which provide robustness to acoustic mismatches. Similar to results in Aurora4, we observed improved performance using multi-band system.

## 3.5 Conclusions

In this chapter, we proposed a new multi-band system. The proposed system has the architecture similar to that of past multi-band system but by employing stream-dropout, the parameters of the model can be jointly trained while reducing coadaptation. We first compared proposed system in a controlled test settings, where only few bands are corrupted. Analysis of performance in these test sets showed that proposed system is capable of reducing coadaptation of the model, with features across sub-bands. This resulted in significant improvements in the synthetic noisy test cases. We also evaluated proposed technique in Aurora4 and ASpIRE test sets, where the speech contains real noises.

Around 10 % relative improvement in WER is observed in these test sets. This shows that proposed multi-band performs better than baseline systems, even in test cases which do not exactly match the assumptions of multi-band (band localized noises).

## 3.6 Future work

The important hyper-parameter in a multi-band system is band-definition (i.e. number of bands and band boundaries). The general rule of thumb is to divide the signal into 5 – 10 sub-bands and each sub-band should cover 1 – 3 critical bands. In all the above experiments the sub-bands are defined arbitrarily. A more principled approach to define the bands is required to obtain optimal performance of multi-band system. Another direction to explore is multi-band architecture with recurrent models. Possible approaches are: (i) replace networks in stage1 with LSTMs, (ii) replace stage2 network with LSTMs, (iii) extract bottleneck features from stage2 network and train a final LSTM network. Even though option (iii) is easier implement, it is too cumbersome to perform fast experimentation, especially on large datasets. Options (i) and (ii) are attractive in this sense, but may require careful tuning of other hyper-parameters of the model, like learning rate, number of LSTM layers etc.

# Chapter 4

# New framework for multi-stream speech recognition system

*This chapter introduces multi-stream speech processing. We provide motivation from engineering point of view and describe past architectures. This is followed by new multi-stream framework which uses stream-dropout technique proposed in previous chapter.*

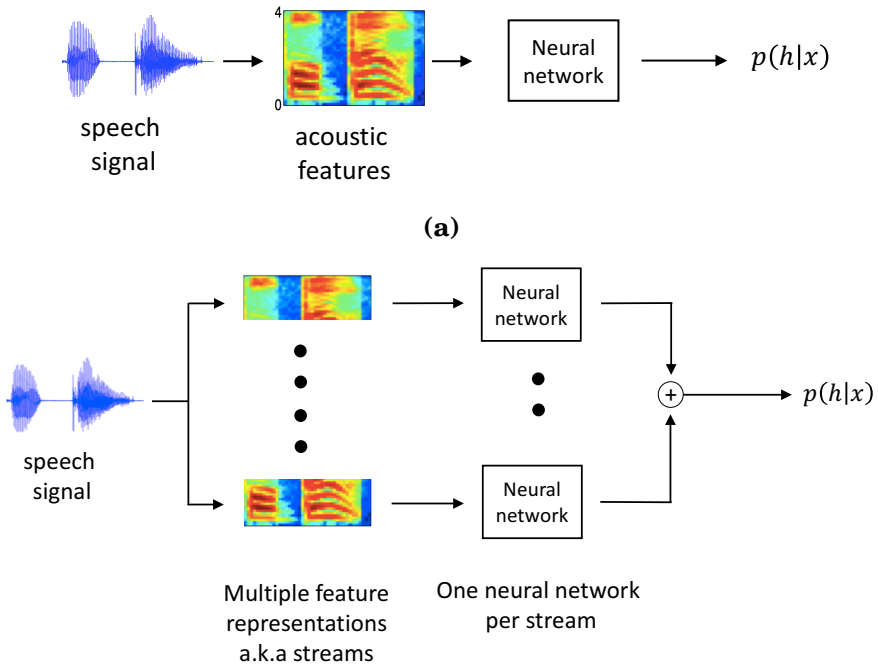


**Figure 4.1:** Illustration of (a) standard neural network based acoustic model architecture and (b) late-fusion multi-stream acoustic model architecture.

## 4.1 Past multi-stream architectures

### 4.1.1 Late-fusion multi-stream architecture

Figure 4.1 shows examples of conventional architecture and late-fusion multi-stream architecture. Conventional architecture involves extracting a single feature representation from the given training data. Examples of widely used feature representations are Mel filter-bank energies [13], Critical Band energies [47] and Mel frequency cepstral coefficients [13]. A neural network

is trained on this feature representation. During test, the same features is extracted from the given test signal, and forward passed through the trained classifier to get phoneme (or tri-phone) posterior probabilities. The phoneme posteriors are divided by the priors of each phoneme to obtain scaled likelihoods, which are then given as inputs to decoder to estimate the word sequence. This architecture is referred to as single-stream architecture due to single feature representation extracted from the signal, and a single classifier.

By contrast, multi-stream architectures involve extracting multiple feature representations from the given training data. Several neural networks are trained, one on each feature representation. During test, the feature representations are forward passed through the respective neural networks. This results in several phoneme posterior vectors, unlike single-stream system which gives only one phoneme posterior vector. Frame-level posteriors from each stream are then merged by weighted linear (or log-linear) combination or non-linear combination using a neural network [56]. The merged phoneme posteriors are used as inputs for the decoder. The main motivation behind using multiple classifiers is, each feature representation is affected by noise differently. This results in some of the feature representations being more robust than others in a given noise. Emphasizing posterior probabilities obtained from these representations, by assigning higher weights, can result in more accurate pos-

teriors. From hereon this architecture is referred to as *late-fusion multi-stream*.

### 4.1.2 Analysis

Many previous implementations of multi-stream system used sub-band streams [56–59], i.e. each stream covers only few frequencies. More general ways forming streams have also been studied. These include streams covering different portions of temporal modulations [60, 61], spectro-temporal (2-D) modulations [62–64], etc.

We first demonstrate robustness of late-fusion multi-stream system using a 2 band system. The two sub-band streams are computed as follows: Log-Mel filterbank features spanning frequency range of 0 – 16000 Hz, are computed from the speech signal. In each Mel band, TRAP features [60] are computed by taking DCT transform over a temporal context of 11 frames. The resulting TRAP features are grouped into Bark critical bands. Mel bands covering first 12 Bark bands are assigned to first stream. The remaining Mel bands features are assigned to second stream.

#### ASR system

A hidden Markov model-deep neural network (HMM-DNN) based ASR system is trained using clean training set of Aurora4 corpus (details in 4.1.2).

A hidden Markov model-Gaussian mixture model (HMM-GMM) system is first trained to generate tri-phone targets, on Mel frequency cepstral coefficients (MFCCs). MFCC features are transformed using linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT). MFCC features are spliced across 7 frames (–left-context=3, –right-context=3) and the dimensionality is reduced to 40 using LDA transform. The 40 dimensional LDA transformed features are diagonalized using MLLT transform estimated over 12 iterations.

The HMM-GMM system trained on MFCC features with LDA+MLLT transform, is used to force align (Viterbi alignments) the training acoustic data to context dependent phoneme (tri-phone) states. We then train a deep neural network (DNN) to estimate tri-phone state posteriors. A fully connected, 6 hidden layer DNN, with 1024 sigmoid units in each hidden layer is used for this purpose. The DNN is first pre-trained using restricted Boltzmann machine (RBM) algorithm [65]. It is then fine-tuned using cross-entropy cost function. The targets used for cross-entropy are the Viterbi alignments generated from the HMM-GMM system.

**Aurora4 data set**

We used Aurora4 training set for the speech recognition experiments. Aurora4 task is a small scale, medium vocabulary speech recognition task, aimed at improving noise and channel robustness [66]. This corpus is derived from the DARPA Wall Street Journal (WSJ0) 5000-word closed vocabulary dictation task.

**Training set:** The clean training set consist of 7,138 utterances recorded using Sennheiser HMD-414 close-talking microphone. These correspond to roughly 15 hours of speech data, from 83 speakers, sampled at 16000 Hz. Along with clean training set, the database also provides multicondition training set. The multicondition training set consist of the same 7,138 utterances filtered using several different microphones and corrupted by adding various noises. Composition of this set is as follows: 3569 utterances (half) recorded with the Sennheiser microphone, and the remaining half recorded with a different microphone (18 different microphone types were used). No noise is added to one-fourth (893 utterances) of each of these subsets. To the remaining three-fourths (2676 utterances) of each of these subsets, 6 different noise types (car, babble, restaurant, street, airport, and train) were added at randomly selected SNRs between 10 and 20 dB. The goal was an equal distribution of noise types and SNRs. Thus, the multicondition dataset consist of one clean set (893 utterances) and 6 noisy

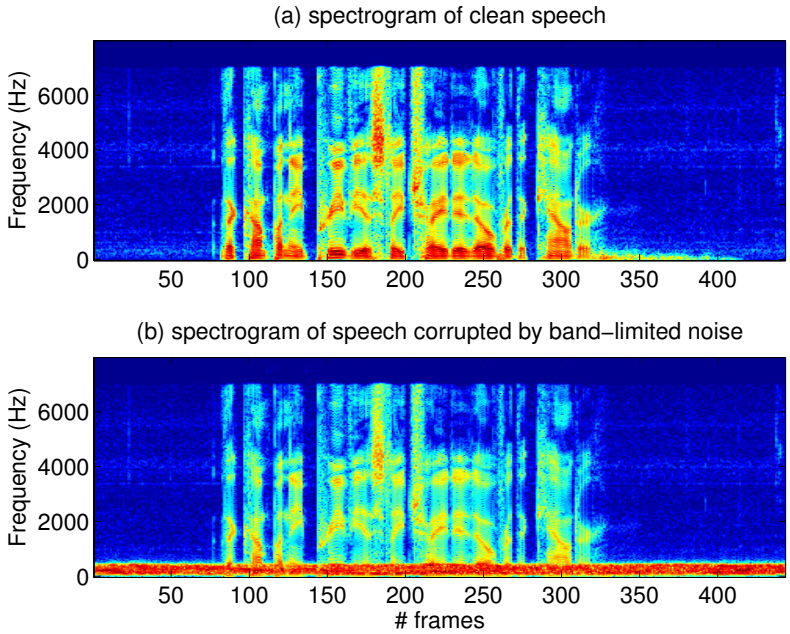


**Figure 4.2:** Comparison of spectrograms of clean speech and speech corrupted by band-limited noises.

subsets (446 utterances each) for both the microphone conditions.

**Band-limited test set:** The clean test set of Aurora4 (test_eval92 set) consist of 330 utterances recorded using Sennheiser microphone. Variant of the clean test set is created by artificially adding band-limited noise at 10 dB signal-to-noise ratio. Figure 4.2 (a) shows spectrogram of a clean speech, where as figure 4.2 (b) shows spectrogram of the same signal corrupted by the artificial band-limited noise. As we can see from the figure, the band-limited noise corrupts only few frequencies, ranging from 500–1000 Hz. This effects stream 1, whereas stream 2 is is mostly unaffected.

**Table 4.1:** Comparison of single-stream and late-fusion multi-stream systems, in band-limited noise and clean conditions.

| | | band-limited | clean |
|---|---|---|---|
| 1 | single-stream | 34.4 | 3.0 |
| 2 | sub-band 1 | 76.2 | 7.6 |
| 3 | sub-band 2 | 7.5 | 4.5 |
| late-fusion multi-stream | | | |
| 4 | **w**=[1/2, 1/2] | 19.0 | 4.0 |
| 5 | **w**=[0.0, 1.0] | 7.5 | 4.5 |

**Results**

Table 4.1 shows comparison results between single-stream system and late-fusion multi-stream system. Row 1 of the table shows WERs obtained from the single-stream system. It is evident from the table that performance of single-stream system degrades significantly even though only a small portion of the spectrum is corrupted by noise. The reason for the degradation might be due to low signal-to-noise ratio (10 dB) of band-limited noise. Rows 2 and 3 of the table show WERs of network trained on sub-band 1 and sub-band 2. It is evident from the table that localized noise does not effect performance of sub-band 2.

Rows 4 and 5 of the table show WERs obtained from late-fusion multi-stream system. The two streams in the system are combined as shown in the figure 4.1b. That is, at each frame sub-band features corresponding to each stream are forward passed through the respective neural networks to obtain

frame level posteriors. Final posterior vector is computed by combining posterior vectors of each stream. We use weighted sum combination rule to combine the posteriors, given by the following equation

$$p_{ct} = \sum_{i=1}^{5} w_i \, p_{it} \tag{4.1}$$

where $p_{ct}$ is combined posterior vector at time $t$, $p_{it}$ is posterior vector of $i^{th}$ stream at time frame $t$, and $w_i$ is weight associated with stream-$i$. The results in row 4 of table 4.1 are obtained by assigning equal weights to all the streams. The systems in rows 1 and 4 are conceptually similar, in the sense that both the systems use all the information present in the input signal. In row 1, the information is fused at the early stages (feature level) and in row 4 the information is fused late, at the posteriors. It can be observed by comparing rows 1 and 4 that late-fusion of streams result in more robust posteriors compared to early fusion.

Performance of late-fusion multi-stream system can be further improved by using the knowledge of which streams are robust in the given test condition. In the present case, we have the knowledge that band-limited noise corrupts only sub-band 1. Row 5 shows the WERs obtained by assigning zero weight to sub-band 1. From the table, we can conclude that further improve-

ment to the performance can be obtained with the knowledge of which streams are corrupted.

Last column of table 4.1 show the WERs obtained in clean condition. It is evident from the table that single-stream system clearly outperforms late-fusion system in clean condition. The reason for this is, single-stream system models features of several streams jointly, thereby accounting for joint information present in the signal. Whereas, in late-fusion multi-stream system, each stream is considered independent of the other. The linear late-fusion of streams, obtained by weighted averaging of streams is not as powerful as non-linear fusion of features using a neural network. There is a clear trade-off in terms of robustness to mismatch data *vs* clean (matched) data, between single-stream and late-fusion multi-stream systems. Single-stream system performs better in clean (matched) data, whereas late-fusion multi-stream system is more robust to mismatches. In next section, we describe another multi-stream architecture which address the drawbacks of late-fusion multi-stream system.

### 4.1.3 Full combination multi-stream architecture

Figure 4.3 depicts the block diagram of a multi-stream system, similar to the one used in [2]. For simplicity, we illustrated a system with 2 streams.

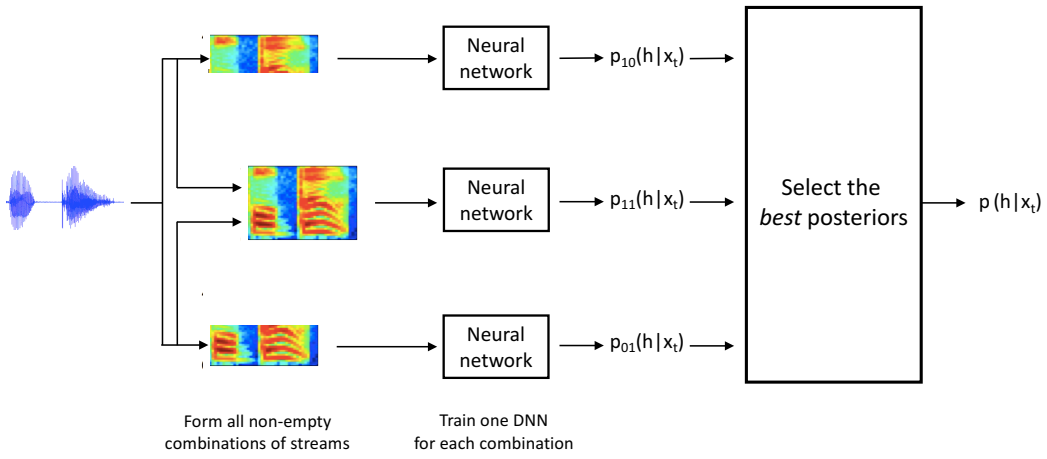


**Figure 4.3:** Block diagram of full combination multi-stream system, based on 2 sub-bands. The architecture is similar to 7 sub-band architecture used in [1, 2]

The principle can be extended to any number of streams. Speech signal is divided into two streams. All non-empty combinations of the streams are formed, resulting the 3 feature representations. Three separate neural networks are trained, one for each feature representation.

### Analysis

Row 6 of table 4.2 shows WERs of full combination multi-stream system in clean and band-limited noisy conditions. It can be observed from the table that full combination system retains the robustness of late-fusion multi-stream system, without any degradation of performance in clean condition. The reason for better performance of full combination system is, we used the knowledge of best streams in order to obtain the results of the table. That is, in clean

test, we used neural which have both the streams (middle neural network in figure 4.3). Whereas in band-limited noisy conditions, we used stream which has only high frequency features (top neural network of figure 4.3).

### 4.1.4 Drawback of full-combination system

In order to build a robust multi-stream system, it is necessary to include as many streams as possible. However increasing the number of streams results in significant increase in the number of networks in the system. For example, a 5-stream system consists of (31 ($2^5 - 1$+5)) networks, and a 9-stream system consists of (511 ($2^9 - 1$+ 9)) networks. This creates a trade-off between robustness and complexity, which can create a road block for practical application of multi-stream system. In the next section, we show that stream-dropout can be used to significantly reduce the complexity of multi-stream system.

**Table 4.2:** Comparison of full combination multi-stream system with single-stream and late-fusion multi-stream systems.

| | | band-limited | clean |
|---|---|---|---|
| 1 | single-stream | 34.4 | 3.0 |
| 2 | sub-band 1 | 76.2 | 7.6 |
| 3 | sub-band 2 | 7.5 | 4.5 |
| late-fusion multi-stream | | | |
| 4 | **w**=[1/2, 1/2] | 19.0 | 4.0 |
| 5 | **w**=[0.0, 1.0] | 7.5 | 4.5 |
| 6 | full combination multi-stream | 7.5 | 3.0 |

## 4.2 How does stream-dropout help multi-stream?

The basic premise of multi-stream approach is acoustic mis-match corrupts only few streams and removing these noisy streams can improve performance of the system. For example, car noise typically corrupts low frequency portion of the signal. So in a multi-stream system, with streams as sub-bands, not using streams corresponding to low frequency portions is better than using all the streams. The information from unreliable streams are removed in full combination system by selecting networks which are not trained with these streams.

In multi-stream with stream-dropout, the noisy streams are removed by *zeroing-out* or *dropout*, i.e. multiply with zero, features from them. Since stream-dropout networks already sees input vector with a portion of it being zeros during training, the network does not breakdown by dropout of noisy streams. We also hypothesize that the network can be tuned to perform comparable to full combination multi-stream system. If stream-dropout multi-stream performs atleast equal to full combination multi-stream, we can achieve a significant reduction in the number of parameters used to build a multi-stream system.

## 4.3 Full combination *vs* stream-dropout

In this section, we compare performance of stream-dropout and full combination multi-stream systems. Aim of the experiments is to study whether we can replace multiple neural networks in full combination systems with a single network trained using stream-dropout. For this purpose, we use sub-band streams. We hypothesize that the conclusions can be valid for other type of streams as no assumption about the type of acoustic features is used.

As described previously, full combination (FC) system consists of 2 stages. The first stage involves processing each sub-band stream independently and deriving sub-band *features*. In most of the previous works [44,67], phoneme posteriors (or pre-softmax outputs known as TANDEM) are used as sub-band features. In the second stage, all non-empty combinations of sub-band features from stage 1 are formed and an MLP is trained for each combination. Along with reducing multiple MLPs in the second stage, we demonstrate that 2 stage architecture can also replaced by a single network.

### 4.3.1 Full combination system

We compare full combination and stream-dropout based multi-stream systems using a multi-band system. Aurora4 [66] database is used for training and evaluation purposes.

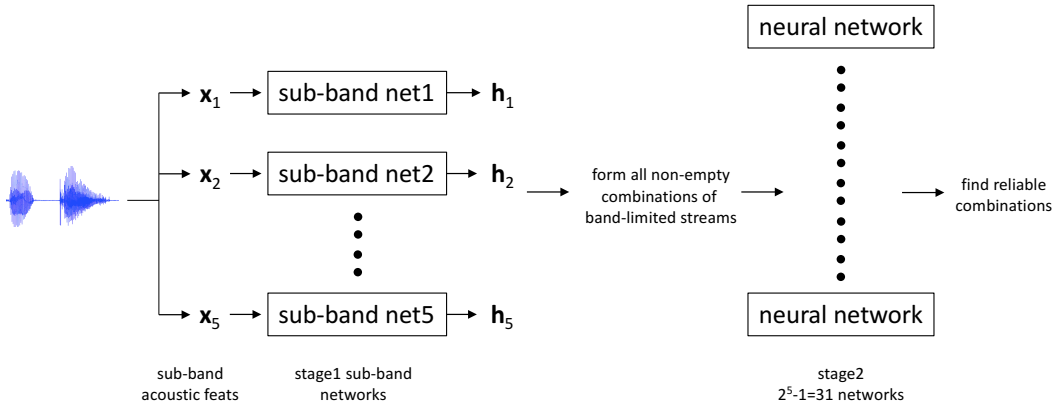


**Figure 4.4:** Architecture of full combination (FC) multi-stream system.

The architecture of FC system is shown in Fig. 4.4. Broadly, the system pipeline consist of 2 stages. The first stage involves extraction of features from sub-band streams and processing them. Formation of sub-band streams is based on Bark critical bands [47]. We used a 5 stream system with each band covering 3 Bark critical bands. The extraction of sub-band streams from the signal is as follows: We extract 63 dimensional Mel filter-bank energies, ranging from 0 to 8000 Hz. The 63 Mel bands are grouped into streams based on which Bark critical band [47] they belong to. The peak of Mel triangular filter is used as criterion for grouping the bands. In each Mel band, a temporal trajectory of 11 frames is transformed using a DCT with 6 basis. This results in 6 dimensional feature vector for each Mel band and a 378 ($63 \times 6$) dimensional feature vector at every time frame.

As shown in Fig. 4.4, one neural network is trained for each stream.

The targets used to train these networks are obtained from a GMM – HMM system. A mono-phone GMM – HMM model is first initialized with means and covariance of all the Gaussians equal to global mean and covariance of the training data. This is referred to as flat-start initialization. Mel frequency cepstral coefficients (MFCCs) with deltas and acceleration features are used for building GMM – HMM models. The parameters of the mono-phone model are reestimated using 3 or 4 iterations of expectation – maximization (EM) algorithm. A tied tri-phone model is initialized by copying the parameters from the corresponding mono-phone model. Parameters of the tri-phone GMM – HMM is reestimated by using EM algorithm. MFCC features are transformed using linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT). MFCC features are spliced across 7 frames (–left-context=3, –right-context=3) and the dimensionality is reduced to 40 using LDA transform. The 40 dimensional LDA transformed features are diagonalized using MLLT transform estimated over 12 iterations. The HMM-GMM system trained on MFCC features, transformed using LDA+MLLT transform, is then used to force align (Viterbi alignments) the training acoustic data to context dependent phoneme (tri-phone) state. The resulting alignments are used as targets to train neural network models.

Architecture of the networks consist of 2 hidden layers with 1500 sig-

moidal units, followed by a linear bottleneck layer with 40 neurons and a final softmax layer with 2037 units. The networks are trained using mini-batch stochastic gradient descent (SGD) algorithm. Once the network is trained, features from the bottleneck layer (40 dimensions per stream) are extracted. The bottleneck features are used as inputs to second stage processing. This approach is slightly different from previous full combination systems [44, 67], which used softmax posteriors of stage 1 networks as inputs to second stage processing. The reason for choosing bottleneck features is, unlike previous works which used mono-phone targets (40–50), we trained the networks on tri-phone state targets (usually 2000). Once the stage 1 networks are trained, bottleneck features are extracted. All possible non-empty combinations ($2^5 - 1 = 31$) of these features are formed and 31 combination networks are trained on these features. The combination networks consist of 4 hidden layers with 1500 sigmoid neurons. We used the same tri-phone targets as that of stage 1 networks, obtained from GMM – HMM system.

### 4.3.2 Stream-dropout multi-stream system

We first show that a single network trained using stream-dropout can be used to replace multiple networks used in full combination system, resulting in new multi-stream architecture shown in Fig. 4.5. Next we show that first and

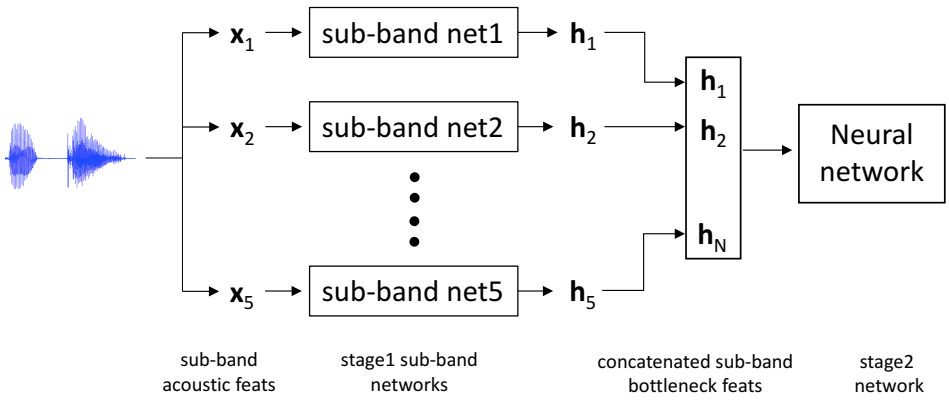


**Figure 4.5:** Architecture of proposed multi-stream system. In this system, we use sub-band as streams.

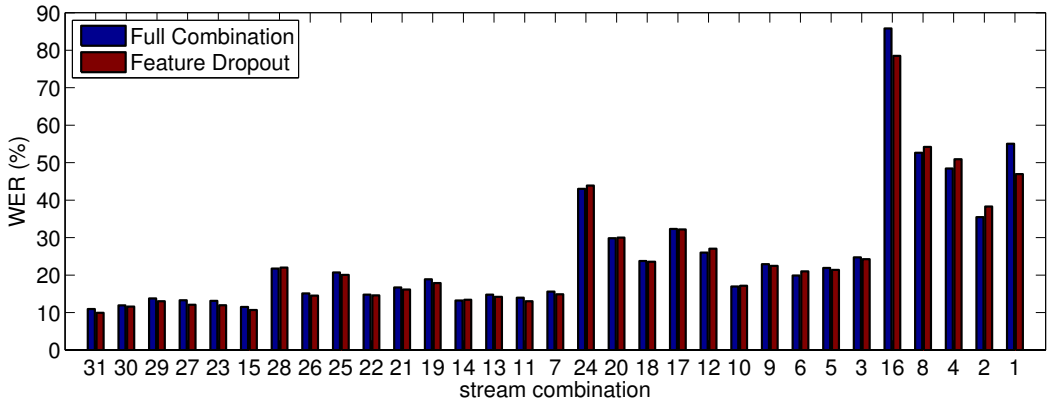


**Figure 4.6:** Comparison of full-combination and stream-dropout system for various stream combinations.

second stage can be jointly trained, there by removing the need to perform the 2 stage procedure used in most of the previous multi-stream implementations [44].

**Multiple networks in stage 2 to single network:**

Bottleneck features from the first stage processing of full combination system are used to construct input feature vector for stream-dropout system. The concatenation procedure is performed as shown in Fig. 4.5 and described in Sec. 3.2. In this scenario, $\mathbf{X}_i$ represents 40 dimensional bottleneck feature vector of stream-$i$. Stream-dropout network is trained by using each $z_i$ as a binary random variable with $p = 0.5$.

Fig. 4.6 shows WERs of full-combination and stream-dropout multi-stream systems for the 31 ($2^5 - 1$) possible stream combinations. Each cluster of bars shows WERs obtained by using the stream combination shown below the cluster. In the case of full-combination system it is the WER obtained by using neural network trained on features defined by the stream combination. Where as in the case of stream-dropout system, it is WER obtained by values of $z_i$ defined by stream combination. For example, lets consider combination number 26 (binary representation of 26 = [1 1 0 1 0]). The red bar of the corresponding cluster, in Fig. 4.6, show WER obtained from network trained on feature vector

($[\mathbf{h}_1; \mathbf{h}_2; \mathbf{h}_4]$), and the blue bar shows WER obtained by SD network by forward passing test data with $\widetilde{\mathbf{h}} = [\mathbf{h}_1; \mathbf{h}_2; \mathbf{0}; \mathbf{h}_4; \mathbf{0}; ]$.

From Fig. 4.6 we can observe that the performance obtained by stream-dropout system is comparable to full-combination system in most of the stream combinations. This demonstrates that multiple networks in full-combination system can be replaced by a single network trained with proposed *stream-dropout* training. Careful observation of the WER values show that in combinations with more streams, stream-dropout system performs better than full-combination system. Whereas in combinations with less number of streams, performance of full-combination system is better than stream-dropout system. This aspect is better illustrated in Table 4.3. Second column in the table show WER of stream combination 31 ($31_2 = 1\ 1\ 1\ 1\ 1$) for SD and FC systems. Third column of the table show averaged WERs of stream combinations with 4 streams present. That is average of streams combinations = $30,\ 29,\ 27,\ 23,\ 15$. Similarly, 3, 4, and 5 clusters show average WERs of stream combinations with 3, 2, and 1 streams, respectively. It is evident from the figure that even though performance of stream-dropout and full-combination systems are comparable, stream-dropout system's performance is better in combinations with more streams and worse in combinations with less number of streams. We hypothesize the reason for lower performance of stream-dropout system in these

**Table 4.3:** Comparison of full-combination (FC) and stream-dropout (SD) multi-stream systems. Columns 2 to 6 show average WERs of stream combinations. Last column shows number of parameters in the system. The systems are evaluated in Aurora4 test set.

| | WER (%) | | | | | # of parameters |
|---|---|---|---|---|---|---|
| | # of streams present | | | | | |
| | 5 | 4 | 3 | 2 | 1 | |
| Full-combination | 10.96 | 12.73 | 16.54 | 26.13 | 55.48 | 432 M |
| Stream-dropout | 9.95 | 11.86 | 16.06 | 26.29 | 53.77 | 72 M |

stream combinations is due to relatively fewer number of examples of a particular input vector type seen during training. Also, it is important to note that very rarely stream combinations with just 1 stream are useful, as these combinations perform significantly worse than combinations with more streams.

The last column of the table shows number of trainable parameters present in full-combination (432 M) and stream-dropout (72 M) systems. It is evident from the table that along with comparable performance, we were able to achieve significant reduction (6 times reduction) in the number of parameters. This allows deployment of multi-stream system in resource constrained applications like on-device speech recognition, and also train deeper models.

**Joint training of stage 1 and 2:**

In previous section, we demonstrated that stream-dropout training can be used to build a multi-stream system which has significantly less com-

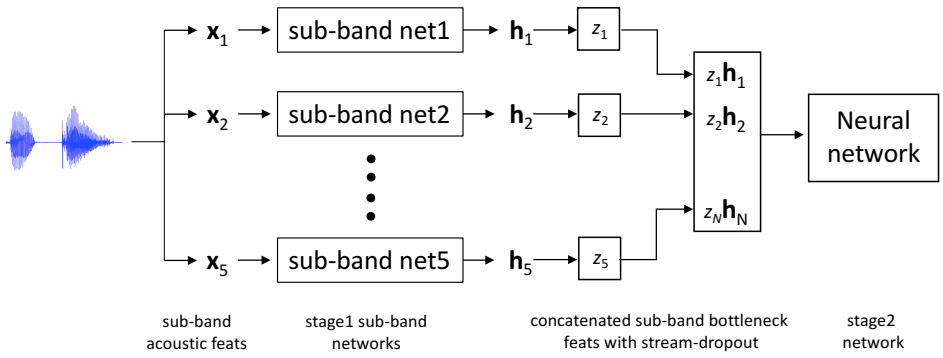


**Figure 4.7:** Architecture of proposed multi-stream system. In this system, we use sub-band as streams.

putational complexity than full combination system. The new multi-stream involves a 2 stage procedure: first, acoustic features in each stream are extracted and processed by using a neural network for each stream. Then processed features (bottleneck features or phoneme posteriors) are extracted from each stream and a stream-dropout network is trained on these features.

The parameters learned in this architecture can be sub-optimal. This is especially true for weights and biases of first stage neural networks, as these are not learned in conjunction with information present in other streams. We hypothesize that new multi-stream architecture can be further improved by jointly training first and stage networks, as illustrated in Fig. 4.7. From figure we can conclude that the resulting neural network is deep ($>$ number of 2 hidden layers) in nature. Previous multi-stream works [2, 44, 67] did not attempt the joint training approach due to difficulty in training these network.

**Table 4.4:** Comparison of full-combination (FC) and stream-dropout (SD) multi-stream systems. Columns 2 to 6 show average WERs of stream combinations. Last column shows number of parameters in the system. All the systems are evaluated in Aurora4 test set.

| System | WER (%) | | | | | | # of parameters |
|---|---|---|---|---|---|---|---|
| | AVG. | # of streams present | | | | | |
| | | 5 | 4 | 3 | 2 | 1 | |
| Full combination | - | 11.4 | 12.9 | 16.6 | 26.0 | 52.2 | 432 M |
| Stream-dropout (A) | 24.9 | 10.1 | 12.1 | 16.4 | 26.7 | 54.2 | 72 M |
| Joint-training systems | | | | | | | |
| Stream-dropout (B) | 24.8 | 10.4 | 12.2 | 16.2 | 26.4 | 54.0 | 72 M |
| Stream-dropout (C) | 24.8 | 10.3 | 12.2 | 16.2 | 26.4 | 54.3 | 72 M |

**A**: 2-step training, with stage1 networks trained first, followed by training of stage2 network.
**B**: 2-step training, with stage1 networks trained first, followed by joint-training of both stage1 and stage2 networks.
**C**: 1-step training, with stage1 and stage2 networks jointly trained in one go.

With the advent of effective initialization techniques [65,68,69] and significant improvement in computational power due to GPUs, we can train these deep architectures.

Table 4.4 shows the performance comparison of various stream-dropout multi-stream systems with full-combination system. First row of the table shows the performance of FC system. The AVG. column shows average of WERs obtained by decoding all possible stream combinations. The other columns show average of WERs of stream combinations with 5, 4, 3, 2 and 1 streams. Row stream-dropout (A), shows WERs obtained by using the 2 stage architecture described in previous section.

Performance of joint training systems are shown in the other rows of the table. Systems in the joint training category differ in terms of initialization and number of hidden neurons of the stage 1 networks. For stream-dropout system (B), we used trained stage 1 networks of stream-dropout (A) and randomly initializing stage 2 network. Then all the parameters of the system (weights and biases of both stage 1 and stage 2) are trained by backpropagating the errors all the way to initial layers of the stage 1 networks. Note that the stream-dropout is employed at the input of stage 2 network. From the table, we can infer that training the model parameters of stage 1 jointly with stage 2 network is resulting in a better performing system. Instead of using trained networks of stage 1, for stream-dropout system (C) we used randomly initialized networks. Table 4.4 shows that performance of system (C) is comparable to system (B). Even though the network in stream-dropout (C) is much deeper than stream-dropout (B), we did not observe any difficulty in training the network. The reason for this is due to ReLU neurons and good initialization of the network [69].

## 4.4 Conclusions

In this chapter we proposed a new multi-stream (MS) architecture. The proposed architecture employs stream-dropout technique, referred to as stream-dropout MS system. The stream-dropout system consists of signifi-

cantly low number of parameters than past full-combination system. Comparison results showed that the parameter reduction is achieved without any noticeable loss in performance.

Currently deep learning ASR research can be classified into two categories: (i) cloud-based ASR, and (ii) on-device ASR. The cloud-based ASR employs massive models to obtain best possible accuracy, whereas on-device models need to work on low power and low memory constraints. Examples include ASR on mobile devices, wake-word keyword detection etc. Also, the latter models are prone to exposure of challenging acoustic conditions as the user can take the device any where she/he wishes to. Stream-dropout technique makes MS systems feasible in these applications, where noise robustness is most needed.

# Chapter 5

# Performance monitoring techniques

*We introduced a new multi-stream architecture in previous chapters, which involves a set of binary masks, one for each stream. In this system, a stream can be included or excluded by setting the corresponding mask value to 1 or 0, respectively. In this chapter, we propose techniques to automatically estimate optimal mask values, further improving the noise robustness of proposed multi-stream system. This is followed by discussion on reducing run-time computational complexity of these techniques.*

The fundamental motivation for using multi-stream approach is noise

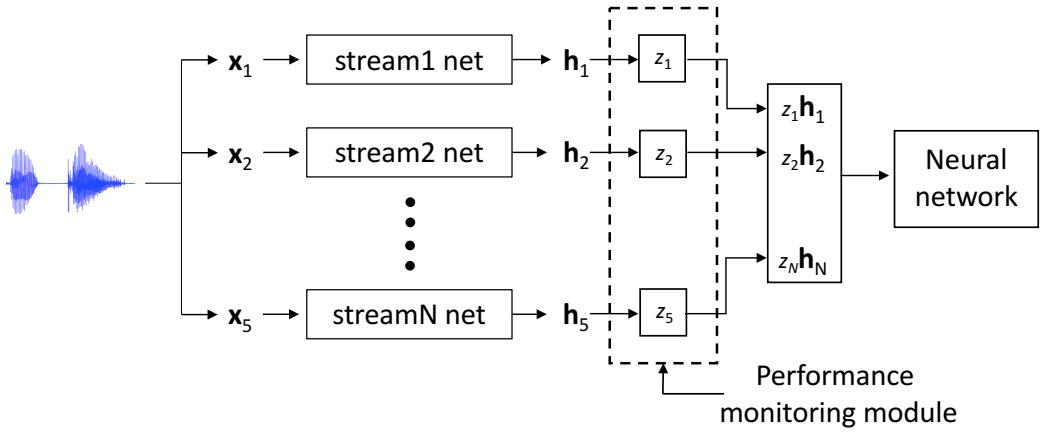


**Figure 5.1:** Stream-dropout multi-stream architecture, with performance monitoring module. During testing of the system the performance module controls the binary masks ($z_i$'s).

or mis-match effects only few portions of signal space. Identifying and emphasizing the reliable regions regions can improve overall recognition accuracy. Historically, techniques which perform this task is referred to as "*performance monitoring*" (PM) techniques. This chapter deals with these techniques. We first describe how performance monitoring (PM) module is used in stream-dropout multi-stream system described in chapter 3. This is followed by description and analysis of proposed techniques, and their comparison with baseline PM methods.

## 5.1 Performance monitoring in stream-dropout multi-stream system

Consider the stream-dropout multi-stream system (referred to as MS system from hereon) illustrated in Fig. 5.1. The first stage involves construction of the streams. This is performed by acoustic feature extraction followed by discriminative transformation using neural networks. In the next stage, fusion of streams from stage1 is performed using stream-dropout component. The stream-dropout component employs binary masks $z_i$ one for each stream. During training each mask is distributed as Bernoulli random variable, and is independent of other masks. That is,

$$z_i \sim \text{Bern}(p = 0.5)$$

$$z_i \perp\!\!\!\perp z_j, \forall i \neq j$$

During test, $z_i$'s are deterministic. In chapter 3, we used $z_i = 1$ for all the sub-band streams. This choice is not optimal, as it is based on the assumption that all streams are equally reliable for all acoustic conditions. Usually noise has a non-uniform effect on the streams. That is, some streams are going to be corrupted more and the other streams are going to be relatively uncorrupted. If the prior knowledge about type of noise is available, we can discard

streams which are highly corrupted by setting the corresponding mask to 0 and by doing that improve performance of the system. Most often this knowledge is not available a priori and therefore there is a need for automatic methods to perform this task. Techniques which try to find the optimal mask value are referred to as performance monitoring methods. Task of performance monitoring (PM) module is to automatically identify optimal mask values, as illustrated in Fig. 5.1.

## 5.2 $\Delta M$ PM measure

In this section, we describe the one of two proposed measures to estimate performance of a neural network classifier. Since the measure is an extension to previously proposed M-measure [3], we first describe M-measure

### 5.2.1 M-measure

Speech message is coded in the form of sequence of sounds (phonemes, syllables etc). Each sound lasts for a certain amount of time and then change into different sound. Therefore it is reasonable to assume that for a high quality speech, frames which are close in time are more similar than speech frames separated by larger time spans. When the speech is degraded by additive noise or other distortions, these affects dominate the signal and sounds become more

similar, resulting frames becoming more similar. The same behavior is observed in posterior probabilities obtained at the output of a neural network [3].

M-measure attempts to quantify this behavior. First it involves computation of $M(\tau)$ which is defined as

$$M(\tau) = \frac{1}{T-\tau} \sum_{t=\tau}^{T} \mathcal{D}(\mathbf{p}_{t-\tau}, \mathbf{p}_t), \tag{5.1}$$

where $\tau$ denotes the time interval between the phoneme posterior probabilities at $t-\tau$ and $t$, $\mathbf{p}_{t-\tau}$ and $\mathbf{p}_t$, and $\mathcal{D}(\mathbf{p}, \mathbf{q})$ denotes the symmetric KL divergence between the posteriors,

$$\mathcal{D}(\mathbf{p}, \mathbf{q}) = \sum_{k=0}^{K} p^{(k)} \log \frac{p^{(k)}}{q^{(k)}} + \sum_{k=0}^{K} q^{(k)} \log \frac{q^{(k)}}{p^{(k)}}, \tag{5.2}$$

where $p^{(k)}$ denotes the $k$-th element of a posterior vector $\mathbf{p} \in \mathbb{R}^K$.

Fig. 5.2 shows the behavior of $M(\tau)$ for clean and noisy speech utterances. It can be observed that $M(\tau)$ is higher for clean speech compared to noisy speech. The reason for this is two frames separated by large distance usually belong to different speech sounds, resulting in higher divergence between posteriors. Whereas in noisy speech, the noise makes the posteriors similar which results in lower divergence values.

The M-measure is defined as average of $M(\tau)$ over several time inter-

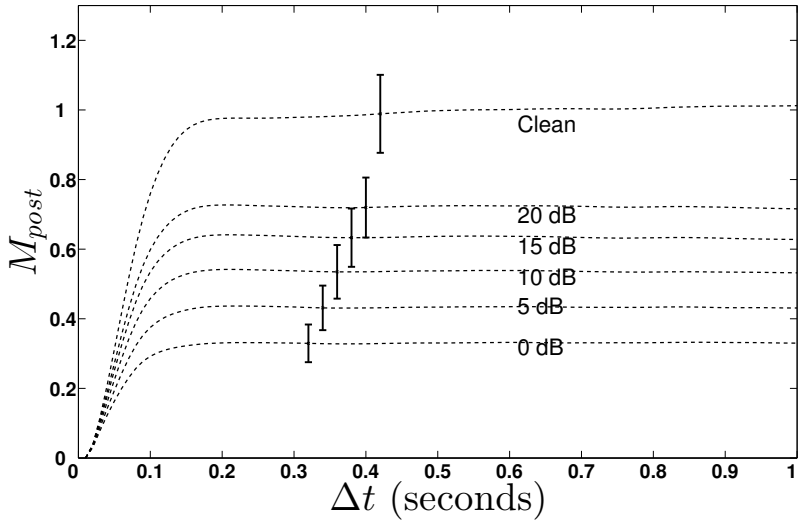


**Figure 5.2:** M-measure curves for in Babble noise at 5 different SNRs, along with variance, using two different MTD formulations [3].

vals $\tau$ and the result is used as measure of performance of neural networks.

$$M = \underset{\{\tau\}}{\text{mean}}[M(\tau)], \tag{5.3}$$

where $\{\tau\}$ consists of 10, 15, 20, $\cdots$, 80 frames (15 intervals).

### 5.2.2 Extending M-measure

M-measure assumes that distance between probability estimates over several time-spans should be large for known data. However, this is not always accurate. If two posteriors are from the same phoneme class, the divergence between them should be small, irrespective of their time seperation. This sit-

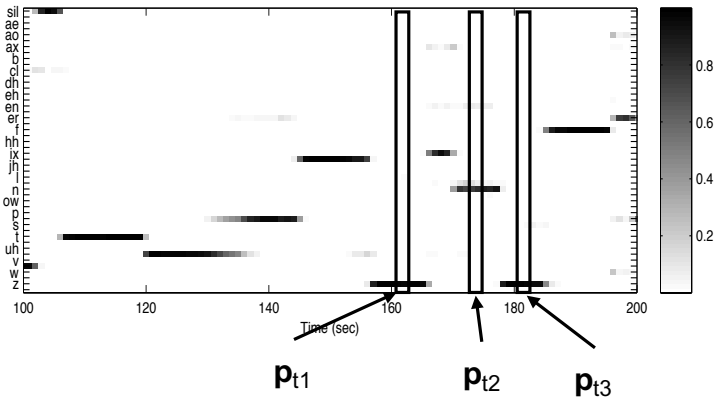


**Figure 5.3:** Posterioragram of a clean speech signal from Aurora4 training set.

uation is illustrated in Fig. 5.3. The figure shows a section of posteriogram of a clean speech signal. Posterior vectors $\mathbf{p}_{t1}$ and $\mathbf{p}_{t3}$ belong to phoneme class $z$, and posterior vector $\mathbf{p}_{t2}$ belongs to phoneme $n$. Therefore, for a good quality posteriogram, $\mathcal{D}(\mathbf{p}_{t1}, \mathbf{p}_{t3})$ should result in a lower value compared to $\mathcal{D}(\mathbf{p}_{t1}, \mathbf{p}_{t2})$.

The M-measure ignores the case that posterior pairs that are separated by large time intervals can belong to the same phoneme class. It accumulates divergence between the frames without considering this kind of phoneme dependency. This implies M-measure selects posteriogram which has $\mathcal{D}(\mathbf{p}_{t1}, \mathbf{p}_{t3}) > \mathcal{D}(\mathbf{p}_{t1}, \mathbf{p}_{t2})$.

This problem is alleviated in the proposed $\Delta M$ measure. We introduce the idea of within-class ($M^{\text{wc}}$) and across-class ($M^{\text{ac}}$) accumulated divergences

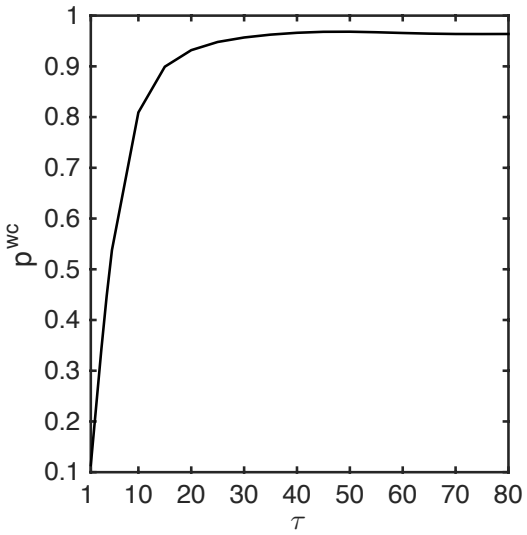


**Figure 5.4:** $p^{\text{ac}}(\tau)$ values for several time gaps ($\tau$).

between posterior vectors, and define $\Delta M$ measure as:

$$\Delta M = M^{\text{ac}} - M^{\text{wc}}. \tag{5.4}$$

In order to estimate $M^{\text{ac}}$ and $M^{\text{wc}}$, we decompose M-measure into

$$M(\tau) = p^{\text{wc}}(\tau) \cdot M^{\text{wc}} + p^{\text{ac}}(\tau) \cdot M^{\text{ac}} + \epsilon_\tau, \tag{5.5}$$

where $M(\tau)$ denotes the original M-measure defined using Eq. (5.1). $p^{\text{wc}}(\tau)$ and $p^{\text{ac}}(\tau)$ denote the probability of a pair of frames separated by $\tau$ being instances from the same and different phonemes, respectively. Note that

$p^{\mathrm{wc}}(\tau) = 1 - p^{\mathrm{ac}}(\tau)$. The error term $\epsilon_\tau$ is included because Eq. (5.5) is an approximate representation of the M-measure. In the equation 5.5, $M(\tau)$, $p^{\mathrm{ac}}(\tau)$ are known quantities. $M(\tau)$ is be computed for each test utterance, and $p^{\mathrm{ac}}(\tau)$ is computed apriori from training data transcriptions. Fig. 5.4 shows $p^{\mathrm{ac}}(\tau)$ values for several time gaps ($\tau$). It can be seen from the figure that even in the case of neighboring frames (i.e. frames separated by $\tau = 1$), 10 % of them correspond to different phones. Also even in the case of frames separated by large time gaps (i.e. $\tau = 80$), 3 % of them correspond to same phone.

Quantities $M^{\mathrm{wc}}$ and $M^{\mathrm{ac}}$ are estimated by solving the following linear equation,

$$\begin{bmatrix} M(\tau_1) \\ \dots \\ M(\tau_N) \end{bmatrix} = \begin{bmatrix} 1 - p^{\mathrm{ac}}(\tau_1) & p^{\mathrm{ac}}(\tau_1) \\ \dots & \dots \\ 1 - p^{\mathrm{ac}}(\tau_N) & p^{\mathrm{ac}}(\tau_N) \end{bmatrix} \begin{bmatrix} M^{\mathrm{wc}} \\ M^{\mathrm{ac}} \end{bmatrix} + \begin{bmatrix} \epsilon_{t_1} \\ \dots \\ \epsilon_{t_N} \end{bmatrix} \tag{5.6}$$

Assuming that $\mathbf{y}$, $\mathbf{A}$, $\mathbf{x}$, and $\epsilon$ are given as

$$\mathbf{y} = \begin{bmatrix} M(\tau_1) & \cdots & M(\tau_N) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^N \tag{5.7}$$

$$\mathbf{A} = \begin{bmatrix} 1 - p^{\mathrm{ac}}(\tau_1) & p^{\mathrm{ac}}(\tau_1) \\ \cdots & \cdots \\ 1 - p^{\mathrm{ac}}(\tau_N) & p^{\mathrm{ac}}(\tau_N) \end{bmatrix} \in \mathbb{R}^{N \times 2} \tag{5.8}$$

$$\mathbf{x} = \begin{bmatrix} M^{\mathrm{wc}} & M^{\mathrm{ac}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^2 \tag{5.9}$$

$$\epsilon = \begin{bmatrix} \epsilon_{t_1} & \cdots & \epsilon_{t_N} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^N \tag{5.10}$$

$M^{\mathrm{wc}}$ and $M^{\mathrm{ac}}$ can be computed by the least square solution:

$$\mathbf{x} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y}. \tag{5.11}$$

The experiments below used the values $(\tau_1, \tau_2, \cdots, \tau_N)$ = (1, 2, 3, 4, 5, 10, 15, 20, $\cdots$, 75, 80) and $N = 20$, which were determined by conducting preliminary experiments. Note that higher $\Delta M$ values indicate more reliable posterior estimates from the DNN classifier.

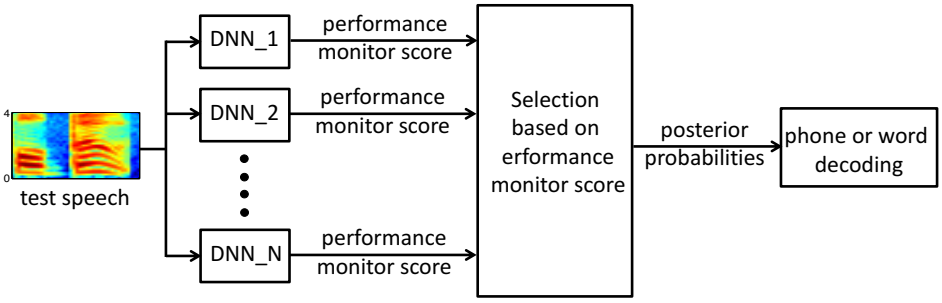


**Figure 5.5:** Stream selection framework used to evaluate various uncertainty estimation measures. Each DNN is trained on a specific noise condition.

### 5.2.3 Stream selection experiments

#### ASR system

In order to compare $\Delta M$ measure with M-measure, we used stream selection framework shown in figure 5.5. The stream selection framework contains several DNN-based classifiers in parallel. Each DNN classifier is referred to as "stream". A sequence of posterior probability vectors is computed for each stream by forward passing a given test utterance through the corresponding DNN. Posteriors of the least uncertain stream are selected, and provided as an input to a phoneme decoder. In each stream, the DNN is trained on a specific noise condition. This results in a multi-stream system where each stream is trained on acoustic environment which is most similar to the environment of test utterance. For a given test utterance, selecting posterior estimates from the stream having the most similar acoustic property as the test utterance re-

sults in the lowest error rate.

**Experiments on TIMIT dataset**

**Experimental setup:**

We used the TIMIT speech dataset [70] for the stream selection experiments. The training set contains 3696 SI and SX utterances from 462 speakers. This totals to 3.12 hours of speech. These are clean, read speech sentences. We used the core development set for the purpose of testing. Five versions of original training set are created by corrupting the clean training speech with four types of additive noise, at various signal-to-noise ratios ranging from 0 dB to 20 dB. We used car, babble, buccaneer1, and buccaneer2 noises from NOISEX database [71]. The original clean training set and four noisy training sets are combined to create a multi-condition training set. The six versions (one clean + four noisy + one multi-condition) of training sets are used to train six different DNNs, where five of them are trained on a specific acoustic condition, and one DNN is trained on multi-condition data. We used a depth of six hidden layers, and each hidden layer consist of 1024 sigmoidal units. Similar to previous studies [72], DNNs are trained on 40 dimensional Mel filter-bank energy features. The DNNs are pre-trained using RBM [65] and fine-tuned using the cross-entropy cost function. The targets used for fine-tuning are context depen-

**Table 5.1:** Comparison of various performance monitoring measures, using phoneme-error-rates (PER) in stream-selection system. The system consists of 5 noise-specific networks and a multi-condition network. The multi-condition network is trained on combined dataset of noise-specific networks.

| | seen noises | | | | | unseen noises | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Train \Test | clean | car | babble | bucc1. | bucc2. | destops. | exhall | f16 | factory | Avg. PER (%) |
| clean | 20.9 | 34.2 | 58.3 | 65.4 | 65.0 | 59.2 | 56.8 | 62.6 | 61.6 | 53.8 |
| car | 23.8 | 22.8 | 58.1 | 65.2 | 64.6 | 56.1 | 54.6 | 62.7 | 60.6 | 52.1 |
| babble | 30.8 | 33.1 | 37.5 | 38.1 | 44.6 | 50.6 | 53.0 | 42.0 | 48.6 | 41.2 |
| bucc1. | 35.4 | 41.3 | 53.7 | 38.1 | 44.9 | 50.6 | 53.0 | 42.0 | 48.6 | 45.3 |
| bucc2. | 37.0 | 45.4 | 58.3 | 45.0 | 37.6 | 50.7 | 56.3 | 46.0 | 51.7 | 47.6 |
| Multi-condition | 22.2 | 24.9 | 39.4 | 42.0 | 43.0 | 39.7 | 38.4 | 39.6 | 40.8 | 36.7 |
| Utterance Oracle | 18.4 | 20.5 | 34.7 | 34.5 | 34.8 | 37.0 | 34.8 | 35.3 | 38.2 | 32.0 |
| Matched condition | 20.9 | 22.8 | 37.5 | 38.1 | 37.6 | 39.7 | 38.4 | 39.6 | 40.8 | 35.0 |
| Entropy | 22.0 | 24.8 | 40.9 | 43.2 | 48.5 | 44.3 | 39.7 | 40.5 | 42.6 | 38.5 |
| M measure | 22.1 | 24.8 | 40.8 | 38.7 | 41.2 | 40.8 | 39.6 | 39.2 | 41.8 | 36.6 |
| $\Delta M$ measure | 22.1 | 24.7 | 40.0 | 38.3 | 41.1 | 41.0 | 39.2 | 39.0 | 41.6 | 36.3 |

dent tri-phone states, generated using a GMM/HMM system trained on clean MFCC features.

We used the development set for testing the models. The development set consists of 34 minutes of speech. Similar to the training set, we corrupted the development set with car, babble, buccaneer1, buccaneer2, destroyerops, exhibition hall, f16 and factory noises, at signal-to-noise ratios of 0, 5, 10, 15 and 20 dB. Four types of noise in this set are seen acoustic variability and the other four noises are unseen acoustic variability in the training set. The whole development set (clean and noisy versions) is referred to as test from here on.

**Experimental results:** Table 5.1 shows the results of test set in various streams. In order to show the upper limit of performance, we defined two oracle stream selection techniques as follows:

- **Utterance oracle:** We select a stream with the lowest error rate for each utterance.
- **Matched condition:** We select a stream trained on the same noise for test utterance with seen noise. For test utterance with unseen noise, we select stream trained with multi-condition data since it results in lower error rate of all the models.

We can infer that error rates of the condition-level oracle streams are

always less than those of individual streams (i.e., clean, car, babble, buccaneer1, and buccaneer2). In addition, the utterance-level oracle streams performs better than the condition-level oracle streams.

We also compared with Entropy measure. It is been observed in several studies that [73, 74] as noise in test data increases, the output posterior probability distribution from a DNN, trained on clean data, converges to non-informative, uniform distribution. This results in an increase in the entropy of the posterior distribution. Based on this observation, entropy was proposed as a performance measure.

It is evident from Table 5.1 that in seen noises, streams trained on a matched noise condition, which corresponds to the condition-level oracle stream, perform better than the streams trained on multi-condition data. Whereas, in the case of unseen noises, choosing the multi-condition stream performs better, as it is more generalizable to unseen noises than condition specific streams.

Table 5.1 shows that entropy of posterior probability, obtained at the output of DNN can be erroneous. M measure is performing better than entropy, which suggests rather than looking at a single frame, measures which look at temporal dynamics of posteriors are better. Further improvement to M measure is obtained by using $\Delta M$ measure.

**Experiments on Aurora4**

We present stream selection experiments performed using Aurora4 database. In this experiment, we attempt to demonstrate that the effectiveness of the proposed measures on TIMIT is generalizable. Similar to the stream selection setups in TIMIT database, we created five versions of training and test sets. The original clean Aurora4 training set is corrupted with car, babble, buccaneer1, and buccaneer2 noises from NOISEX database [71]. A network is trained on each of the noise condition to create condition-specific streams. For testing the stream-selection system, clean subset of Aurora4 test set is is corrupted with car, babble, buccaneer1, buccaneer2, destroyerops, exhall, f16 and factory noises. Table 5.2 shows the stream selection results on the Aurora4 database. From this table, we can conclude that the proposed measures provide significant improvements over the conventional measures. These results also indicate that the proposed measures are generalizable to other databases.

### 5.2.4 Stream-dropout multi-stream system

In this section, we illustrate the effectiveness of $\Delta M$ technique as performance monitoring module in stream-dropout multi-stream system. For this purpose, we use a 5 sub-band multi-stream system based on stream-dropout training described at the beginning of the chapter (shown in Fig. 5.1). The sys-

**Table 5.2:** Comparison of various performance monitoring measures, using WERs in stream-selection system. The system consists of 5 noise-specific networks and a multi-condition network. The multi-condition network is trained on combined dataset of noise-specific networks.

| | seen noises | | | | | unseen noises | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Train \Test | clean | car | babble | bucc1. | bucc2. | destops. | exhall | f16 | factory | Avg. WER (%) |
| clean | 6.5 | 18.8 | 76.0 | 76.1 | 79.7 | 59.6 | 68.6 | 62.5 | 73.8 | 58.0 |
| car | 7.7 | 7.0 | 65.4 | 66.9 | 74.2 | 51.3 | 58.8 | 57.3 | 69.2 | 50.9 |
| babble | 14.6 | 24.3 | 22.2 | 49.5 | 60.8 | 38.1 | 24.5 | 32.3 | 35.8 | 33.6 |
| bucc1. | 19.8 | 38.1 | 64.2 | 24.0 | 37.3 | 45.0 | 58.6 | 26.3 | 44.9 | 39.8 |
| bucc2. | 18.5 | 45.5 | 76.5 | 34.6 | 22.4 | 42.1 | 65.5 | 32.6 | 51.6 | 43.3 |
| Multi-condition | 8.7 | 12.5 | 32.7 | 43.3 | 50.0 | 37.7 | 32.1 | 33.3 | 40.7 | 32.4 |
| Utterance Oracle | 4.3 | 5.3 | 20.4 | 21.1 | 20.5 | 27.4 | 21.6 | 19.7 | 30.1 | 18.9 |
| Matched condition | 6.5 | 7.0 | 22.2 | 24.0 | 22.4 | 37.7 | 32.1 | 33.3 | 40.7 | 25.1 |
| Entropy | 8.2 | 12.0 | 36.3 | 41.3 | 48.1 | 39.7 | 34.5 | 33.6 | 43.0 | 33.0 |
| M measure | 6.7 | 8.9 | 38.2 | 30.2 | 34.2 | 41.8 | 35.8 | 30.2 | 46.8 | 30.3 |
| $\Delta M$ measure | 6.7 | 8.3 | 30.5 | 26.4 | 30.4 | 40.5 | 31.2 | 28.4 | 44.8 | 27.5 |

tem consists of 5 binary masks ($z_1, z_2, z_3, z_4$ and $z_5$), where each one can take 0 or 1. For a given speech utterance, this results in 31 (excluding the null combination) possible posteriorgrams. $\Delta M$ is computed on each of the 31 posteriorgrams and the one with largest $\Delta M$ is given as input to lattice generation module.

**Speech material:** In order to illustrate the effectiveness of $\Delta M$ measure we use Aurora4 speech database [66]. The models are trained on clean training set of Aurora4. The training set consists of utterances recorded using Sennheiser HMD-414 close-talking microphone, with 5000 word vocabulary size. These correspond to roughly 15 hours of speech data, from 83 speakers, sampled at 16000 Hz.

The test set can be classified into 14 subsets. A clean subset which consist of 330 utterances is recorded using Sennheiser microphone. The second subset consist of the same 330 utterances recorded over secondary microphones. This is to evaluate the robustness of ASR systems to microphone mismatches. The remaining 12 subsets were defined by adding each of the 6 noise types ( street, babble, train, car, restaurant, airport) at randomly chosen SNR between 5 and 15 dB for each of the microphone types. The goal was to have an equal distribution of each of the 6 noise types and the SNR with an average SNR of 10 dB. These test sets are usually grouped into 4 subsets: clean (1 test case, group

A), additive noise (6 test cases, group B), clean with channel distortion (1 test case, group C) and additive noise with channel distortion (6 test cases, group D). The 330 recordings in each test set correspond to 40 minutes of speech, so the total test set size is close to 9.3 hours.

In the next subsections, we analyze the effect of various parameters involved in computation of $\Delta M$.

**Time span sampling**

It is shown in previous section that computation of $\Delta M$ measure involves computation of $M(\tau)$ at various $\tau$ values. Using all possible $\tau$s is not practical as it involves computation of $\mathcal{O}(T^2)$ KL divergences, where $T$ is length of input utterance. In order to compute $\Delta M$ measure, we need to estimate two quantities $M^{\text{ac}}$ and $M^{\text{wc}}$. Since we are interested in estimation of just two quantities, we hypothesize that using smaller number of $\tau$ s are sufficient for accurate computation. We tested with $\tau$ s $1, 2, 3, 4, 5, \cdots, 75, 80$. Table 5.3 shows the comparison between $\Delta M$ measure computed from all possible intervals and the pre-defined intervals. It can be inferred from the table that using the pre-defined intervals is sufficient there by massively reducing computational complexity.

**Table 5.3:** Comparison of WERs (%) using $\Delta M$ PM, computed using two time-intervals $T1$ and $T2$. $T1$ uses all time steps up to 80 frames, and $T2$ uses time steps $1, 2, 3, 4, 5, \cdots, 65, 70, 75, 80$.

| Condition | Stream-dropout MS | | |
|---|---|---|---|
| | w/o PM | $\Delta M$ (time intervals) | |
| | | $T1$ | $T2$ |
| Clean Same Mic. | | | |
| clean | 2.6 | 2.6 | 2.6 |
| Clean Diff. Mic. | | | |
| clean | 19.8 | 13.6 | 13.3 |
| Additive Noise Same Mic. | | | |
| airport | 32.1 | 31.2 | 31.3 |
| babble | 38.6 | 37.8 | 38.1 |
| car | 18.5 | 16.0 | 14.9 |
| restaurant | 40.8 | 40.4 | 40.5 |
| street | 42.6 | 41.2 | 40.3 |
| train | 44.3 | 42.8 | 42.8 |
| Avg. | 36.1 | 34.9 | 34.6 |
| Additive Noise Diff. Mic. | | | |
| airport | 47.9 | 45.5 | 45.2 |
| babble | 54.2 | 48.9 | 48.9 |
| car | 33.7 | 25.4 | 24.4 |
| restaurant | 56.0 | 53.7 | 52.9 |
| street | 59.1 | 55.0 | 53.7 |
| train | 58.1 | 54.2 | 53.1 |
| Avg. | 51.5 | 47.1 | 46.4 |
| Overall Avg. | 37.6 | 34.8 | 34.3 |

**Effect of class type in DNN estimations**

Typically, in a state-of-the-art ASR systems, the output softmax layer represent categorical posterior distribution over tri-phone state classes. Here, we analyze whether tri-phone posteriors are optimal for performance monitoring using $\Delta M$. We compare tri-phone posteriors with monophone posteriors. Instead of training a new network directly on monophone targets, we reuse the network trained on tri-phone targets to compute mononphone distribution. The monophone distribution is computed by merging probabilities of tri-phone classes. The tri-phone classes to merge to obtain a monophone class is defined many-to-one mapping between tri-phone and monophone classes.

Table 5.4 shows WERs obtained from the stream-dropout multi-stream system, using various performance monitoring measures. Second column of the table (w/o PM) shows performance obtained by using DNN trained using stream-dropout and during testing, we do not use performance monitoring module and and pass data from the all the streams unchanged.

Columns 3 and 4 of the table 5.4 show performance improvements obtained by using $\Delta M$ measure on tri-phone and monophone posteriors, respectively. From the table, we can observe that using $\Delta M$ further improves the performance of the stream-dropout system. Also, it is important to note that $\Delta M$ PM never degrades the performance. The improvement illustrates that

**Table 5.4:** WERs(%) obtained by using $\Delta M$ performance monitoring on triphone and phoneme posteriors in stream-dropout multi-stream system. All the models in the system are trained on clean training set of Aurora4 dataset.

| Condition | Stream-dropout MS | | |
|---|---|---|---|
| | w/o PM | $\Delta M$ | |
| | | PDF post. | phone post. |
| Clean Same Mic | | | |
| clean | 2.6 | 2.6 | 2.6 |
| Clean Diff. Mic | | | |
| clean | 19.8 | 13.3 | 12.8 |
| Additive Noise Same Mic | | | |
| airport | 32.1 | 31.3 | 30.6 |
| babble | 38.6 | 38.1 | 37.2 |
| car | 18.5 | 14.9 | 14.6 |
| restaurant | 40.8 | 40.5 | 40.6 |
| street | 42.6 | 40.3 | 40.3 |
| train | 44.3 | 42.8 | 42.4 |
| Avg. | 36.1 | 34.6 | 34.2 |
| Additive Noise Diff. Mic | | | |
| airport | 47.9 | 45.2 | 45.0 |
| babble | 54.2 | 48.9 | 49.1 |
| car | 33.7 | 24.4 | 24.5 |
| restaurant | 56.0 | 52.9 | 53.0 |
| street | 59.1 | 53.7 | 53.7 |
| train | 58.1 | 53.1 | 52.8 |
| Avg. | 51.5 | 46.4 | 46.3 |
| Overall Avg. | 37.6 | 34.3 | 34.2 |

with explicitly dropping out noisy, non-informative streams can result in a substantial improvement (8.7 % improvement over w/o PM results). From the tables we can infer that monophone posteriors are better suited for performance monitoring task than tri-phone posteriors, as consistent improvements are observed across all the acoustic conditions. We hypothesize that the reason for this improvement is due to smooth nature of phoneme posteriors compared to tri-phone posteriors.

Another advantage of using monophone posteriors is computational complexity. Computation of $\Delta M$ involves computing KL divergence between several pairs of posterior vectors separated by $\tau$ (form equations 5.5). Using tri-phone posteriors can result in a significant computational overload, as we need to compute KL divergence between distributions with 1000s of classes. Using monophone posteriors can alleviate this problem as number of classes are typically in the range of 50-100 (number of phonemes), resulting in a reduction of an order of magnitude in number of operations involved in computing KL divergence.

## 5.3 Autoencoder based PM measure

Application of autoencoders for performance monitoring task is based on the following premise: A DNN classifier is best performing and least uncer-

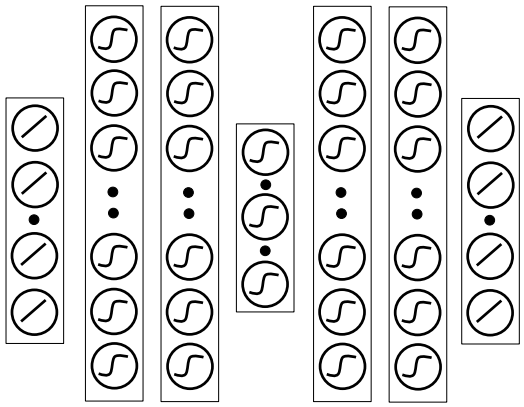

**Figure 5.6:** Illustration of seven layered autoencoder with five non-linear hidden and two linear visible layers.

tain about its posterior probability estimates on its training data. Therefore, uncertainty of a test data can be measured by computing the deviation in probability estimates derived from the test and the training data.

One straightforward way to compute the deviation between training and test data posteriors is to compute KL divergence between all possible train and test data posterior vector pairs. But this approach is computationally intensive, as it can involve massively large number of KL divergence computations for each test frame. One alternative to the above approach is to model the training data posteriors, and evaluate test posteriors on this model. In this work we employ autoencoders to model the training data posteriors.

An autoencoder is a multi-layered feed-forward neural network, used in the context of unsupervised learning. The network is trained to reconstruct

input at its output. This is achieved by training the network to minimize the squared error cost between an output vector from the autoencoder and the corresponding target vector. The targets used to train the network are inputs themselves. The cost function used to optimize the network parameters ($\mathcal{W}$) is described as

$$\min_{\mathcal{W}} \mathbf{E}||\mathbf{x} - \hat{\mathbf{x}}||_2^2, \tag{5.12}$$

where x is an input vector and $\hat{\mathbf{x}}$ is an output vector from the network. Figure 5.6 shows an autoencoder consisting of 5 hidden layers. An autoencoder with more than one non-linear hidden layer is shown to capture complex, non-linear manifolds present in the training data [6, 75, 76]. In order to avoid a trivial identity mapping (the network weights equal to the unit matrix), the number of nodes in the third hidden layer needs to be less than the input (and output) layer.

### 5.3.1 Basic property for performance monitoring

Since the network is trained to minimize the reconstruction error, a vector sampled from the distribution of training data will yield a low reconstruction error compared to vectors drawn from a different distribution. This property is illustrated in Fig. 5.7, which shows distributions of $l_2$ norm of reconstruction error vectors ($||e||^2$), computed from the training data (train), data

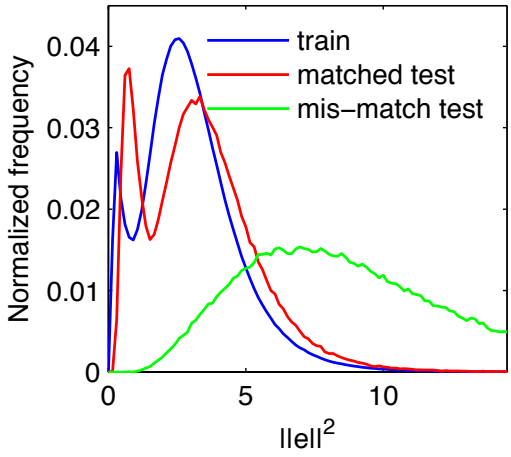


**Figure 5.7:** Illustration of property of autoencoder useful to distinguish matched data and mis-matched data.

similar to the training data (matched test), and data that deviate from the training data (mis-match test). Fig. 5.7 illustrates that the reconstruction error is a good indicator for measuring the mismatch between the training and test data.

### 5.3.2 Stream selection experiments

Tables 5.5 and 5.6 show performance of autoencoder performance monitoring (AE_PM) in stream selection framework described in previous section. It is evident from the tables that AE_PM is performing better than all the other measures. Also, the performance of AE_PM is matching with the condition-level oracle stream. This shows that, in seen noisy cases, AE_PM is successful in selecting condition specific streams and in unseen noisy cases, it is select-

**Table 5.5:** Comparison of various performance monitoring measures, using phoneme-error-rates (PER) in stream-selection system. The system consists of 5 noise-specific networks and a multi-condition network. The multi-condition network is trained on combined dataset of noise-specific networks.

| | seen noises | | | | | unseen noises | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Train \Test | clean | car | babble | bucc1. | bucc2. | destops. | exhall | f16 | factory | Avg. PER (%) |
| clean | 20.9 | 34.2 | 58.3 | 65.4 | 65.0 | 59.2 | 56.8 | 62.6 | 61.6 | 53.8 |
| car | 23.8 | 22.8 | 58.1 | 65.2 | 64.6 | 56.1 | 54.6 | 62.7 | 60.6 | 52.1 |
| babble | 30.8 | 33.1 | 37.5 | 38.1 | 44.6 | 50.6 | 53.0 | 42.0 | 48.6 | 41.2 |
| bucc1. | 35.4 | 41.3 | 53.7 | 38.1 | 44.9 | 50.6 | 53.0 | 42.0 | 48.6 | 45.3 |
| bucc2. | 37.0 | 45.4 | 58.3 | 45.0 | 37.6 | 50.7 | 56.3 | 46.0 | 51.7 | 47.6 |
| Multi-condition | 22.2 | 24.9 | 39.4 | 42.0 | 43.0 | 39.7 | 38.4 | 39.6 | 40.8 | 36.7 |
| Utterance Oracle | 18.4 | 20.5 | 34.7 | 34.5 | 34.8 | 37.0 | 34.8 | 35.3 | 38.2 | 32.0 |
| Matched condition | 20.9 | 22.8 | 37.5 | 38.1 | 37.6 | 39.7 | 38.4 | 39.6 | 40.8 | 35.0 |
| Entropy | 22.0 | 24.8 | 40.9 | 43.2 | 48.5 | 44.3 | 39.7 | 40.5 | 42.6 | 38.5 |
| M measure | 22.1 | 24.8 | 40.8 | 38.7 | 41.2 | 40.8 | 39.6 | 39.2 | 41.8 | 36.6 |
| $\Delta M$ measure | 22.1 | 24.7 | 40.0 | 38.3 | 41.1 | 41.0 | 39.2 | 39.0 | 41.6 | 36.3 |
| AE_PM | 20.9 | 22.9 | 37.0 | 37.2 | 37.1 | 41.0 | 37.8 | 39.0 | 42.0 | 35.0 |
| AE_PM+$\Delta M$ | 20.9 | 22.9 | 36.8 | 36.6 | 36.8 | 39.8 | 37.2 | 39.0 | 41.0 | 34.6 |

**Table 5.6:** Comparison of various performance monitoring measures, using WERs in stream-selection system. The system consists of 5 noise-specific networks and a multi-condition network. The multi-condition network is trained on combined dataset of noise-specific networks.

| | seen noises | | | | | unseen noises | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Train \Test | clean | car | babble | bucc1. | bucc2. | destops. | exhall | f16 | factory | Avg. WER (%) |
| clean | 6.5 | 18.8 | 76.0 | 76.1 | 79.7 | 59.6 | 68.6 | 62.5 | 73.8 | 58.0 |
| car | 7.7 | 7.0 | 65.4 | 66.9 | 74.2 | 51.3 | 58.8 | 57.3 | 69.2 | 50.9 |
| babble | 14.6 | 24.3 | 22.2 | 49.5 | 60.8 | 38.1 | 24.5 | 32.3 | 35.8 | 33.6 |
| bucc1. | 19.8 | 38.1 | 64.2 | 24.0 | 37.3 | 45.0 | 58.6 | 26.3 | 44.9 | 39.8 |
| bucc2. | 18.5 | 45.5 | 76.5 | 34.6 | 22.4 | 42.1 | 65.5 | 32.6 | 51.6 | 43.3 |
| Multi-condition | 8.7 | 12.5 | 32.7 | 43.3 | 50.0 | 37.7 | 32.1 | 33.3 | 40.7 | 32.4 |
| Utterance Oracle | 4.3 | 5.3 | 20.4 | 21.1 | 20.5 | 27.4 | 21.6 | 19.7 | 30.1 | 18.9 |
| Matched condition | 6.5 | 7.0 | 22.2 | 24.0 | 22.4 | 37.7 | 32.1 | 33.3 | 40.7 | 25.1 |
| Entropy | 8.2 | 12.0 | 36.3 | 41.3 | 48.1 | 39.7 | 34.5 | 33.6 | 43.0 | 33.0 |
| M measure | 6.7 | 8.9 | 38.2 | 30.2 | 34.2 | 41.8 | 35.8 | 30.2 | 46.8 | 30.3 |
| $\Delta M$ measure | 6.7 | 8.3 | 30.5 | 26.4 | 30.4 | 40.5 | 31.2 | 28.4 | 44.8 | 27.5 |
| AE_PM | 6.7 | 7.0 | 22.5 | 23.9 | 23.5 | 37.3 | 24.8 | 25.1 | 39.9 | 23.4 |
| AE_LDA+$\Delta M$ | 6.7 | 7.0 | 22.2 | 23.3 | 23.1 | 36.5 | 24.2 | 25.1 | 38.9 | 23.0 |

ing the multi-condition stream. The reason for the better performance could be the ability of autoencoders to model distributions lying on a complex non-linear manifolds [76]. In addition, combination of AE_PM and $\Delta M$ seems to improve over the condition-level oracle stream. This implies that the AE_PM+$\Delta M$ is able to select a stream, not just based on similarity with the stream's training data, but also based on the stream's performance.

### 5.3.3 Stream-dropout multi-stream

In previous section, we demonstrated superiority of autoencoder PM over baseline PM measures in stream selection system with condition specific streams. In this section, we apply autoencoders for performance monitoring in stream-dropout multi-stream system. We first present the experiments performed in order identify the optimal parameter setting for training autoencoders.

#### Optimal features

**Senone posteriors:**

As described previously, application of autoencoders for PM involves modeling training data posteriors using an autoencoder. Performance monitoring score of a test speech utterance is proportional to the reconstruction errors

**Table 5.7:** WERs(%) obtained by using AE_PM trained on raw posteriors and logit-posteriors in stream-dropout multi-stream system. All the models in the system are trained on clean training set of Aurora4 dataset.

| Cond. | Stream-dropout MS | | |
|---|---|---|---|
| | w/o PM | AE_PM | |
| | | raw post. | logit-post. |
| Clean Same Mic | | | |
| clean | 2.6 | 3.1 | 2.6 |
| Clean Diff. Mic | | | |
| clean | 19.8 | 13.2 | 12.8 |
| Additive Noise Same Mic | | | |
| airport | 32.1 | 31.2 | 31.6 |
| babble | 38.6 | 40.9 | 38.6 |
| car | 18.5 | 11.0 | 14.9 |
| restaurant | 40.8 | 43.3 | 40.6 |
| street | 42.6 | 52.2 | 48.5 |
| train | 44.3 | 49.9 | 48.5 |
| Avg. | 36.1 | 38.0 | 37.1 |
| Additive Noise Diff. Mic | | | |
| airport | 47.9 | 47.6 | 46.8 |
| babble | 54.2 | 56.0 | 53.9 |
| car | 33.7 | 25.6 | 28.0 |
| restaurant | 56.0 | 57.6 | 55.8 |
| street | 59.1 | 61.5 | 60.6 |
| train | 58.1 | 61.5 | 59.8 |
| Avg. | 51.5 | 52.1 | 50.8 |
| Overall Avg. | 37.6 | 38.9 | 37.3 |

of the test data posteriors by the autoencoder. Higher reconstruction error indicates greater mis-match between training and test acoustic conditions. We hypothesize that appropriate transformation of raw posteriors to generate features for training autoencoders in AE_PM is crucial. In this section, we present experiments to identify optimal features to train a AE_PM module for stream-dropout MS system.

In order to illustrate the importance of experiments conducted in this section, we used raw tri-phone posteriors to train AE_PM module. The AE_PM is trained as follows: Once the DNN classifier in stream-dropout MS is trained, entire training data is forward passed through the DNN to obtain tri-phone posteriors. During the forward passing, the five binary masks (one for each sub-band stream) are set to 1. The assumption here is for training data, retaining all the streams result in *best* posteriors. Then, an autoencoder is trained to minimize the reconstruction error of these posteriors. The autoencoder consist of an input layer, 2 sigmoidal layers (512 units), a bottleneck layer (25 units), 2 sigmoidal layers (512 units) and an ouput layer, and it is trained in mini-batch SGD algorithm with a batch size of 256 frames. We used newbob learning rate scheduling with initial learning rate of 0.0008. For a given test utterance, 31 posteriograms are extracted, one for each stream combination. PM scores of the 31 posteriograms are computed by AE_PM module and posteriogram with high-

est PM score is provided as input to decoder module. Table 5.7 shows the word error rates obtained by stream-dropout MS system w/o PM and with AE_PM trained on raw tri-phone posteriors. The models are trained on Aurora4 clean training set and tested on standard Aurora4 test set. It can be observed from the table that employing raw posteriors+AE_PM actually degrades the performance of the system significantly compared to w/o PM. This experiment shows that raw tri-phone posteriors are not suitable for AE_PM module, and proper feature preparation is required to AE_PM module.

We hypothesize that the reason for performance degradation observed in using raw posteriors is that they lie in [0, 1] and are hard to model. One way to alleviate this problem is to transform posteriors from [0,1] to [-$\infty$, +$\infty$]. We experimented with logit-transform of posteriors. Column 4 of table 5.7 show results using AE_PM trained logit-posteriors. From the table, we can infer that AE_PM trained on logit-posteriors performs noticeably better than AE_PM trained on raw posteriors. This shows that instead of modeling raw posteriors, modeling transformed (logit-posteriors) results in better autoencoders. Also, using AE_PM trained on logit-posteriors always results in an improvement (albeit marginally) over w/o PM.

We demonstrated that logit transformation on posteriors results in a better PM module. Fig. 5.8 shows eigenvalues of eigenvectors obtained using

**Table 5.8:** WERs(%) of stream-dropout multi-stream system obtained by using several AE_PM modules. Each of the AE_PM module is trained on PCA transformed logit-posteriors, with varying the PCA dimensionality. All the models in the system are trained on clean training set of Aurora4 dataset.

| Condition | Stream-dropout MS + AE PM | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | logit-post. | logit-post._PCA | | | | | | | |
| | | 2004D | 1000D | 500D | 200D | 100D | 50D | 40D | 30D |
| Clean Same Mic | | | | | | | | | |
| clean | 2.6 | 2.7 | 2.6 | 2.6 | 2.7 | 2.7 | 2.9 | 3.1 | 3.2 |
| Clean Diff. Mic | | | | | | | | | |
| clean | 12.8 | 19.3 | 14.4 | 12.7 | 12.9 | 12.8 | 12.7 | 13.3 | 13.8 |
| Additive Noise Same Mic | | | | | | | | | |
| airport | 31.6 | 31.3 | 29.5 | 30.2 | 31.2 | 31.1 | 30.8 | 32.7 | 32.9 |
| babble | 38.6 | 43.3 | 36.5 | 37.3 | 38.1 | 38.5 | 38.2 | 38.0 | 38.2 |
| car | 14.9 | 17.3 | 13.6 | 11.7 | 15.4 | 15.1 | 15.0 | 12.5 | 12.6 |
| restaurant | 40.6 | 46.6 | 42.7 | 40.8 | 40.5 | 40.7 | 40.5 | 41.8 | 42.4 |
| street | 48.5 | 65.5 | 60.3 | 47.7 | 44.1 | 44.4 | 44.3 | 42.6 | 48.6 |
| train | 48.5 | 71.2 | 64.6 | 46.4 | 45.2 | 46.0 | 45.3 | 44.3 | 47.5 |
| Avg. | 37.1 | 45.6 | 41.0 | 35.7 | 35.8 | 35.9 | 35.7 | 35.2 | 37.0 |
| Additive Noise Diff. Mic | | | | | | | | | |
| airport | 46.8 | 48.4 | 45.6 | 46.2 | 46.5 | 45.8 | 46.2 | 47.7 | 47.1 |
| babble | 53.9 | 64.8 | 58.0 | 53.2 | 53.1 | 53.6 | 52.2 | 52.7 | 52.1 |
| car | 28.0 | 36.2 | 28.5 | 25.6 | 27.5 | 27.6 | 27.2 | 27.0 | 27.8 |
| restaurant | 55.8 | 64.6 | 57.9 | 55.0 | 55.0 | 55.0 | 54.5 | 56.5 | 56.9 |
| street | 60.6 | 76.6 | 71.0 | 59.5 | 57.4 | 57.7 | 56.7 | 56.2 | 58.9 |
| train | 59.8 | 77.6 | 73.9 | 60.1 | 57.6 | 58.4 | 57.3 | 57.2 | 58.8 |
| Avg. | 50.8 | 61.2 | 55.7 | 49.8 | 49.5 | 49.6 | 49.0 | 49.5 | 50.3 |
| Overall Avg. | 37.3 | 46.8 | 42.0 | 36.5 | 36.2 | 36.3 | 35.9 | 36.0 | 37.2 |

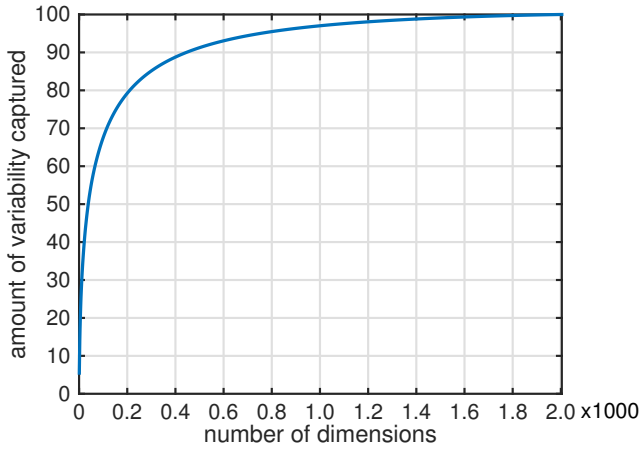


**Figure 5.8:** Principal component analysis on logit-posteriors of DNN trained on clean Aurora4 training set.

principal component analysis (PCA) of logit-posteriors. It is evident from the figure that all the 2004 dimensions of posteriors are not required, as most of the variability (80–90 %) is captured by first few hundred factors. It is worthwhile exploring the option of using dimensionality reduction on logit-posteriors, as only relevant dimensions are retained, resulting in a well trained autoencoders in AE_PM module. Table 5.8 shows the performance of several AE_PM modules, each of them with varying the number of PCA factors retained. Column 2 of the table shows results of AE_PM trained on logit-posteriors, whereas columns 3 – 8 show results of AE_PM trained PCA transformed logit-posteriors. It is evident from the table using all the dimensions of PCA results in a significant reduction in performance. However, instead of using all the dimensions, using dimensions $<$ 500 results in an improvement over logit-posteriors, with 50 di-

mensions giving the best performance. The reason for this behaviour is that the discarded dimensions are not informative as they have very less variance. Removing these non-informative dimensions results in a better autoencoder, thereby resulting in a better PM module. We hypothesize that other types of transformations (e.g. LDA etc) can be used to obtain similar results. However note that transforms like LDA require more data for reliable estimation.

**Phone posteriors:** Table 5.8 shows that AE_PM module trained on dimensionality reduced posteriors performs better than the one which uses all the dimensions. The dimensionality reduction is performed using PCA. Another way of reducing the dimensions of tri-phone posteriors is to convert them to phone posteriors, by merging probabilites of tri-phones which belong to the same phoneme class. Phone posteriors are smoother than tri-phone posteriors, and these are shown to be performing better in the $\Delta M$ PM case. Also, AE_PM trained on phone posteriors of one dataset can be used to evaluate neural networks trained on another dataset (provided both the datasets are from same language).

Comparison of AE_PM module trained on tri-phone logit-posteriors and phone logit-posteriors is shown in Table 5.9. It is evident from the table that using phone posteriors results in a better AE_PM module. AE_PM can be further improved by whitening the phone posteriors. Column 3 of Table 5.9

shows results obtained by AE_PM trained on PCA transformed phone logit-posteriors. The reason for this improvement is due to PCA transform makes the phone logit-posteriors more amenable to modeling by autoencoders. The reason for this improvement is due to whitened logit-posteriors posteriors, using PCA transformation, are more amenable to modeling by autoencoders than raw logit-posteriors.

**Context modeling using autoencoders**

In AE_PM models used in the previous sections, the autoencoders are trained to reconstruct a frame of posteriorgram (or its transformed version). The models assume each posterior vector independent and does not account for temporal dependecy between posterior frames. But the underlying signal is speech, which is produced by the movement of vocal tract. Due to the inertia in these movements, the vocal tract typically stays in a configuration for a certain amount to time to produce a phoneme. And it slowly changes to a different configuration, resulting in different sound unit. This is reflected in the clean posteriogram obtained at the output of neural network as well (Fig. 5.3).

Therefore it makes intuitive sense that incorporating the temporal structure of posteriors into AE_PM models should result in a better performance monitoring module. One, simple, way to integrate the temporal struc-

**Table 5.9:** Comparison between AE_PM modules trained on logit tri-phone posteriors, logit monophone posteriors and PCA transformed logit monophone posteriors. The models are trained on clean Aurora4 dataset.

| Condition | Stream-dropout MS | | |
|---|---|---|---|
| | AE_PM | | |
| | logit-post. | phone-post_logit | phone-post_logit_PCA |
| Clean Same Mic | | | |
| clean | 2.6 | 2.6 | 2.8 |
| Clean Diff. Mic | | | |
| clean | 12.8 | 11.7 | 12.1 |
| Additive Noise Same Mic | | | |
| airport | 31.6 | 33.1 | 32.8 |
| babble | 38.6 | 40.0 | 39.2 |
| car | 14.9 | 12.8 | 13.3 |
| restaurant | 40.6 | 46.4 | 42.7 |
| street | 48.5 | 51.2 | 45.4 |
| train | 48.5 | 50.5 | 47.2 |
| Avg. | 37.1 | 38.8 | 36.8 |
| Additive Noise Diff. Mic | | | |
| airport | 46.8 | 43.2 | 46.3 |
| babble | 53.9 | 50.2 | 49.4 |
| car | 28.0 | 23.4 | 25.0 |
| restaurant | 55.8 | 56.3 | 55.4 |
| street | 60.6 | 57.2 | 55.9 |
| train | 59.8 | 56.4 | 56.6 |
| Avg. | 50.8 | 47.8 | 48.1 |
| Overall Avg. | 37.3 | 37.0 | 35.9 |

ture of posteriorgram is to train the autoencoder on a sequence of posterior frames, rather than a single frame. In this scenario, the model emphasizes stream combination which results in temporal structure similar to that of temporal structure observed in training data. Table 5.10 shows results of several AE_PM models. Each of the models shown in table differ in the amount of context used as input to the autoencoders. It is evident from the table that AE_PM models with context significantly outperforms AE_PM models without context. Also, a consistent improvement is observed till the context of 11 frames (110 milliseconds). Furthermore, adding more context is not resulting in an improvement and the performance is oscillating (11 frames = 34.2 %, 13 frames = 34.4 % and 15 frames = 34.1 %).

Training autoencoders on a context of frames is the first step in incorporating temporal structure in to AE_PM models. The models can be further improved by using recurrent layers (e.g. LSTMs) which are inherently good at modeling the temporal structure. The models are not explored in this work, and we plan to use them in future studies.

## 5.4 Combination of PM measures

In previous sections, we introduced two techniques to perform the task of selecting optimal streams in multi-stream system. The techniques are re-

**Table 5.10:** Comparison of AE_PM modules trained on a different contexts of PCA transformed logit monophone posteriors.

| Condition | Stream-dropout MS | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | w/o PM | AE_PM Context | | | | | | | | | |
| | | 0 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 |
| Clean Same Mic | | | | | | | | | | | |
| clean | 2.6 | 2.8 | 2.6 | 2.6 | 2.6 | 2.7 | 2.7 | 2.7 | 2.7 | 2.8 | 2.8 |
| Clean Diff. Mic | | | | | | | | | | | |
| clean | 19.8 | 12.1 | 11.8 | 11.5 | 11.9 | 11.7 | 11.9 | 12.1 | 11.7 | 12.6 | 12.3 |
| Additive Noise Same Mic | | | | | | | | | | | |
| airport | 32.1 | 32.8 | 31.9 | 32.2 | 32.1 | 31.4 | 31.2 | 31.3 | 31.2 | 31.9 | 32.3 |
| babble | 38.6 | 39.2 | 38.1 | 37.6 | 38.2 | 38.0 | 38.0 | 37.9 | 37.4 | 37.0 | 37.5 |
| car | 18.5 | 13.3 | 12.0 | 12.3 | 12.1 | 11.8 | 11.6 | 11.8 | 11.5 | 11.0 | 11.2 |
| restaurant | 40.8 | 42.7 | 40.9 | 41.0 | 41.1 | 41.1 | 41.3 | 41.3 | 41.0 | 42.7 | 43.7 |
| street | 42.6 | 45.4 | 43.8 | 43.2 | 44.0 | 42.7 | 41.4 | 41.4 | 40.5 | 40.6 | 40.3 |
| train | 44.3 | 47.2 | 45.1 | 44.3 | 45.0 | 44.4 | 43.1 | 43.8 | 42.8 | 41.9 | 42.4 |
| Avg. | 36.1 | 36.8 | 35.3 | 35.1 | 35.4 | 34.8 | 34.3 | 34.5 | 34.0 | 33.9 | 34.4 |
| Additive Noise Diff. Mic | | | | | | | | | | | |
| airport | 47.9 | 46.3 | 45.5 | 45.3 | 44.9 | 44.6 | 44.5 | 44.5 | 44.3 | 45.5 | 45.4 |
| babble | 54.2 | 49.4 | 50.0 | 50.0 | 49.3 | 49.2 | 49.1 | 48.8 | 49.0 | 48.4 | 49.2 |
| car | 33.7 | 25.0 | 24.2 | 23.4 | 23.6 | 23.4 | 22.8 | 23.1 | 22.7 | 23.6 | 23.1 |
| restaurant | 56.0 | 55.4 | 54.1 | 54.4 | 54.5 | 53.8 | 53.6 | 53.7 | 54.1 | 54.2 | 55.1 |
| street | 59.1 | 55.9 | 54.9 | 54.8 | 55.4 | 54.9 | 54.0 | 54.9 | 53.9 | 54.8 | 54.0 |
| train | 58.1 | 56.6 | 54.8 | 54.5 | 54.7 | 54.6 | 54.2 | 54.1 | 54.0 | 54.3 | 54.0 |
| Avg. | 51.5 | 48.1 | 47.3 | 47.1 | 47.1 | 46.8 | 46.4 | 46.5 | 46.3 | 46.8 | 46.8 |
| Overall Avg. | 37.6 | 35.9 | 34.9 | 34.8 | 34.9 | 34.6 | 34.2 | 34.4 | 34.1 | 34.4 | 34.5 |

**Table 5.11:** Performance comparison (WERs (%)) of rank based combination of $\Delta M$_PM and AE_PM measures.

| Condition | Stream Dropout DNN | | | |
|---|---|---|---|---|
| | w/o PM | $\Delta M$_PM | AE_PM | $\Delta M$_PM+AE_PM |
| Clean Same Mic | | | | |
| clean | 2.6 | 2.6 | 2.7 | 2.6 |
| Clean Diff. Mic | | | | |
| clean | 19.8 | 12.8 | 11.7 | 11.5 |
| Additive Noise Same Mic | | | | |
| airport | 32.1 | 30.6 | 31.2 | 30.5 |
| babble | 38.6 | 37.2 | 37.4 | 37.1 |
| car | 18.5 | 14.6 | 11.5 | 13.2 |
| restaurant | 40.8 | 40.6 | 41.0 | 40.7 |
| street | 42.6 | 40.3 | 40.5 | 39.6 |
| train | 44.3 | 42.4 | 42.8 | 42.6 |
| Avg. | 36.1 | 34.2 | 34.0 | 33.8 |
| Additive Noise Diff. Mic | | | | |
| airport | 47.9 | 45.0 | 44.3 | 44.4 |
| babble | 54.2 | 49.1 | 49.0 | 49.2 |
| car | 33.7 | 24.5 | 22.7 | 23.6 |
| restaurant | 56.0 | 53.0 | 54.1 | 53.0 |
| street | 59.1 | 53.7 | 53.9 | 53.0 |
| train | 58.1 | 52.8 | 54.0 | 53.1 |
| Avg. | 51.5 | 46.3 | 46.3 | 46.1 |
| Overall Avg. | 37.6 | 34.2 | 34.1 | 33.8 |

ferred to as $\Delta M$_PM and AE_PM. The $\Delta M$_PM emphasizes stream combination which results in a posterior frames which have greater within-class and across-class separability. Whereas, AE_PM measure emphasizes stream combination which results in posteriors similar to that of training data. The fundamental principles of these two measures are different, so it is natural to assume that the fusion of these techniques should result in a better performance monitoring module. In this section, we present experiments aimed at combining the two techniques.

In order to make the problem more abstract, we introduce the following notation. Let us assume the multi-stream consists of $K$ streams, resulting in $N = 2^K - 1$ stream combinations. Given a test utterance, the stream-dropout multi-stream system generates $N$ posteriograms $P_1, P_2, \cdots P_n \cdots P_N$. We use performance monitoring module to select the *best* posteriogram and give it as input to decoder (to generate lattice). More formally, the best posteriogram is defined as

$$P_{final} = \underset{P \in \{P_1, P_2, \cdots, P_N\}}{\arg\max} s(P) \tag{5.13}$$

where $s()$ is a scoring function, implemented using PM module, described in the above section.

The above equation can be modified to account for multiple perfor-

mance techniques as:

$$P_{final} = \underset{P \in \{P_1, P_2, \cdots, P_N\}}{\arg\max} g(s_1(P), s_2(P), \cdots s_M(P)) \tag{5.14}$$

where $M$ is number of PM techniques and $g()$ is a scalar valued function which combines the individual PM scores ($s_i(P)$).

The simplest way of combining the two proposed PM techniques is to add their scores $s_1(P)$ and $s_2(P)$. The main issue with this combination is that the combined score can be biased towards one of the measures, because of the difference in dynamic ranges of the scores.

We propose to use rank based combination to combine the scores from the PM modules. In this method, the combined scores are computed as follows:

$$RP(g) = (\Pi_{m=1}^{M} r_{g,m})^{1/M} \tag{5.15}$$

where $r_{g,m}$ rank of $m^{th}$ PM measure and $g^{th}$ stream combination. The choice of using rank based combination is guided by two factors: (i) Different scales: (ii) The issue with different scales of the scores can be normalized using Z-norm ( [77, 78]). But the scores are most often non-Gaussian in nature, and multiple PM methods can produce which are distributed differently. The advantage of using rank based combination is it is non-parametric in nature, as it depends

only on the ranks of individual scores. Table 5.11 shows results of combination of the measures.

It is evident from the table that combining both the measures further improve overall average performance (34.2 % to 33.8 %). Also, it can be observed from the table that the combination improves only in cases when both $\Delta M$_PM and AE_PM are performing comparably. In cases where one of the measure is considerably worse than the other, combination WER is close to mid-point. This shows there is further scope of improvement in combination. One possible extension is to assign an exponent parameter to each individual rank, and tune this parameter.

## 5.5 Improving test time complexity

For a given test utterance, we need to identify stream combination which results in lowest error-rate. In previous works [2, 44, 74], posteriors of all the stream combinations are evaluated using performance monitoring technique. In this approach, number of forward passes required to identify best stream combination increases exponentially with number of streams. For example, a 7-stream system requires 127 forward passes [2]. The large number of forward passes at test time can deter practical applicability of the system.

In this section, we propose a technique to reduce the test time com-

**Table 5.12:** WERs (%) obtained by proposed combination pruning algorithm and evaluating all possible stream combinations.

| Condition | all_combination | topN=1 |
|---|---|---|
| Clean Same Mic | | |
| clean | 2.6 | 2.6 |
| Clean Diff. Mic | | |
| clean | 10.1 | 10.2 |
| Additive Noise Same Mic. | | |
| airport | 30.0 | 30.0 |
| babble | 36.2 | 36.5 |
| car | 13.7 | 13.8 |
| restaurant | 39.9 | 39.8 |
| street | 37.6 | 37.5 |
| train | 39.0 | 39.4 |
| Avg. | 32.7 | 32.8 |
| Additive Noise Diff. Mic | | |
| airport | 43.9 | 43.9 |
| babble | 48.1 | 48.4 |
| car | 22.2 | 22.6 |
| restaurant | 51.9 | 51.9 |
| street | 52.9 | 53.3 |
| train | 51.4 | 51.5 |
| Avg. | 45.2 | 45.4 |
| Overall Avg. | 33.1 | 33.2 |

plexity of multi-stream system. The technique is based on the idea that stream combinations can be organized into a tree, where each node represents a stream combination. Figure 5.9 shows an example organization of stream combinations in a tree. Using this organization, we introduce parent, child relation between stream combinations. We traverse the tree from root node to leaf nodes, and at each step we prune the possible stream combinations, based on performance monitor scores. Pseudo-code of the search algorithm is described in algorithm 1. Using the algorithm, we reduce the worst-case complexity from $O(2^N)$ to $O(N^2)$ number of evaluations, where N . In practice, number of evaluations required are much fewer than $O(N^2)$. For the 9-stream system used in table 5.12, number of evaluations required to find best stream combination are $\approx 30$.

Table 5.12 shows WER results obtained by proposed combination pruning algorithm, with $topN = 1$. That is, at each stage of the tree we only consider best combination and proceed down. It can be observed from the table that the performance degradation is very small compared to evaluating all possible combinations. This shows that proposed technique is effective in selecting good stream combinations with significantly less number of evaluations.

```
1  parent_node = root_node ;
2  parent_score = root_score ;
3  FindBestChildNode(parent_node, parent_score)
4  |  compute scores of all child nodes ;
5  |  if parent_score ≥ max(child_scores) then
6  |  |  return (parent_node, parent_score) ;
7  |  else
8  |  |  best_child_node = (child_node with score ==
   |  |   max(child_scores)) ;
9  |  |  FindBestChildNode(best_child_node,
   |  |   best_child_node_score) ;
10 |  end
11 end
```

**Algorithm 1:** *Search algorithm used to find best stream combination. In our implementation* root_node *refers to combination where all the streams are present.*

## 5.6 Conclusions and Future work

In this chapter, we proposed two new performance monitor techniques, $\Delta M$_PM and AE_PM. Motivation for $\Delta M$_PM is that for a clean signal, divergence between within-class posterior frames should be lower than across-class posteriors. The measure is defined as the difference between average across-class and within-class divergences. The AE_PM is based on the principle that a classifier performs best on data similar to its training data. An autoencoder is used to model training data posteriors of the classifier, and reconstruction error of the test data is used as measure to select the streams. We also proposed a technique to combine the proposed measures. The combination method is based

on rank of the individual measures, which makes it usable for combining any number of heterogeneously distributed performance monitor methods. Finally a method is proposed to rapidly identify the optimal stream combination, which obviates the need to search all possible combinations.

Experimental comparisons carried out in a multi-stream phoneme recognition paradigm demonstrated the effectiveness of the proposed measures. The proposed measures yielded improvements over the existing measures, and achieved almost the same performance as the oracle performance. In addition, the stream selection framework with proposed uncertainty estimation performed more robust against noise than the conventional multi-condition training approach. The measures were shown to generalize well to multi-stream LVCSR system developed on AURORA4 database.

Although AE_PM is shown to be best of the techniques, we observed the technique to be highly sensitive to its hyper-parameters. This needs to be addressed as it involves tuning the model when the acoustic model and/or dataset is changed. Exploring methods to make AE_PM stable is required. This can be done by employing other kinds of architectures, different cost functions etc. Also approaches to further improve the run-time speed is required.

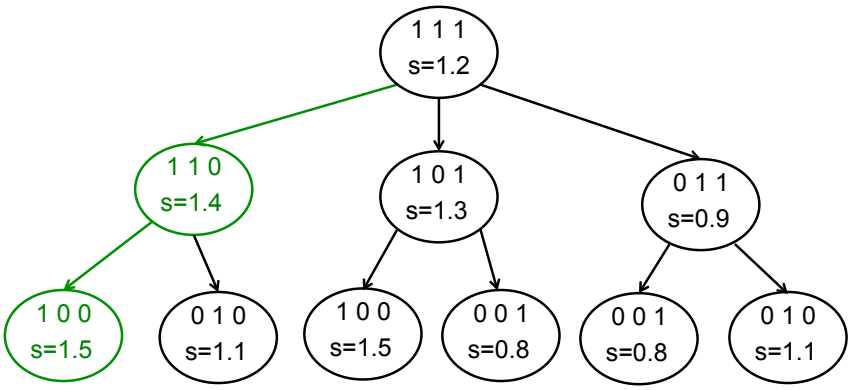


**Figure 5.9:** Illustration of tree organization of all possible stream combinations.

# Chapter 6

# Summary and Future Extensions

*In this chapter, we summarize the important contributions of the thesis. We discuss limitations of the proposed techniques. Also, we present a discussion on future directions to address the limitations and further improve the multi-band ASR for noise robust speech recognition.*

## 6.1 Contributions of the thesis

In order to highlight the main contributions of the thesis, we provide a summary table (Tab. 6.1) which shows the improvements obtained by using

**Table 6.1:** Comparison of WERs obtained in Aurora4 test set, with the models trained on Aurora4 multi-condition data. The table highlights the improvements obtained by proposed techniques (columns 4 and 5). Also, the scope of potential improvement is illustrated by using oracle performance monitor (column 6).

| | DNN | Multi-band | Multi-band+stream-dropout | | |
|---|---|---|---|---|---|
| | | | w/oPM | AE+$\Delta M$_PM | oraclePM |
| Clean | 4.2 | 3.9 | 3.9 | 4.0 | 1.7 |
| Clean+Diff Mic | 11.7 | 9.6 | 6.4 | 5.9 | 3.1 |
| Noise | 8.1 | 6.6 | 5.8 | 5.8 | 3.8 |
| Noise+Diff Mic | 21.8 | 17.2 | 15.2 | 13.3 | 8.6 |
| Overall Avg. | 13.6 | 10.9 | 9.6 | 8.8 | 5.7 |

each of the proposed techniques.

WERs in column 2 (labelled *DNN*) show the results obtained by using a standard deep neural network (DNN) model. We used a fully connected model with 6 hidden layers, with each hidden layer consisting of 1500 ReLU neurons. This system forms the conventional architecture for acoustic model in most of the ASR systems. Column 3 shows the results of architecture, which is similar to the one used in past multi-band studies. The main difference between DNN and multi-band systems is, the initial layers of multi-band system are localized to one sub-band. This makes the features obtained from these layers to be unaffected by distortions in other sub-bands, thereby preserving the robustness. In Aurora4 test set, multi-band architecture results in a significant improvement $\approx 20$ % relative reduction in WER. Models in column 2 and 3 form the baselines for proposed techniques in the thesis.

Column 4 shows the WER results obtained by application of stream-dropout technique to multi-band system in column 3. This is the first of the two proposed techniques of the thesis. It can be observed from the table that by application of stream-dropout we can further reduce the WER from 10.9 to 9.6 % ($\approx$ 12 % relative reduction in WER). Note that since all other parameters (number of weights, training data, cost function etc) are same as that of multi-band system, it can be concluded that the improvement is mainly due to stream-dropout training of the model.

Results in columns 5 and 6 shows that performance of the system can be further improved by using performance monitor techniques to select optimal sub-bands. Performance of proposed PM methods is shown in column 5. It can seen from the table that application of PM methods can improve the performance by $\approx$ 8.3 % relative (9.6 to 8.8 %). Note that, typically PM method can select different sub-bands for every utterance. This makes intuitive sense as each test utterance has different kind of noises. The final column in the table (column 6) shows the WERs obtained by using oracle performance monitor. Results in this column are obtained by decoding a given utterance for each sub-band combination and selecting the combination which results in lowest WER. Using oracle PM, we obtain a significant 35 % relative reduction in WER. This is to demonstrate the upper-limit of the multi-band system with *perfect* perfor-

mance monitor.

All the contributions of the thesis can be grouped into a new multi-band acoustic model, and improvements to multi-stream acoustic model training and testing.

- **New multi-band acoustic model (Chap. 3, Sec. 3.2):** The main difference between proposed multi-band and the ones used in [43, 44, 49] is all the networks in the multi-band system are trained jointly, while still retaining robustness to coadaptation across sub-bands. This is performed by an end-to-end training of the network parameters (joint training) and employing a stream-dropout component while concatenating features from sub-band specific networks (reduces coadaptation).

- **New multi-stream architecture (Chap. 4, Sec. 4.3):** Past multi-stream architecture employs one classifier for each stream combination. While this architecture is manageable to build acoustic models using small number of streams (3 – 5), the complexity explodes with increasing the number of streams. We show that the stream-dropout technique, proposed for multi-band AMs, can be used to significantly reduce the parameters involved in conventional multi-stream system without any degradation. Even though the comparison is performed on sub-band streams, this approach is applicable to any kind of streams.

- **Two performance monitor techniques (Chap. 5):** Proposed $\Delta M_$PM improves over M-measure by incorporating probability of frames separated by $\Delta_t$ belonging to the same phone class (Chap. 5, Sec. 5.2.2). The second method is an extension to GMM based performance monitor [67], where GMMs are replaced with autoencoders to model the training data posteriors (Chap. 5, Sec. 5.3 ). Extensive experiments were performed to find the optimal hyper-parameters settings of autoencoders (Chap. 5, Sec. 5.3.3). The techniques are then applied to proposed stream-dropout multi-band system to further improve its robustness (Chap. 5, Sec. 5.3.3).

- **Rank-based combination of PM measures (Chap. 5, Sec. 5.4):** A rank-based technique is proposed to combine various performance monitor measures. Since the combination is based on ranks, it is robust to dynamic range of individual measures. A mathematical formulation is proposed to combine heterogenously distributed PM measures.

- **Improving test time speed (Chap. 5, Sec. 5.5):** In past multi-stream works [2, 49], optimal stream combination is identified by computing PM score for all possible stream combinations. This brute force approach cannot be applied in the case where more than 9 streams are used. In this thesis, a search algorithm is proposed which massively decreases the number of computations involved. The stream combinations are organized in

a tree structure, and at each stage improbable (defined by low PM score) combinations are pruned.

## 6.2 Future research directions

**Band-definition:** The width of sub-bands in the experiments are 1 – 3 Bark critical band long. However we observed that the sub-band definition is critical to multi-band system's performance. The hypothesize is, this might be due to type of colored nature of the test noisy conditions. Further analysis of this aspect is required to obtain a more robust system.

**Recurrent models:** All the experiments performed in this thesis used feed forward neural networks (i.e. DNNs or CNNs). Most state-of-the-art speech recognition systems use some for recurrent architectures (e.g. RNNs, LSTMs, LSTMPs etc). Application of recurrent models in multi-band architecture needs to be explored. One approach is to replace the feed-forward models in sub-band networks with recurrent models, and the sub-band features are then provided as input to fusion network. A fully recurrent model can also be explored by replacing all the networks with recurrent networks.

**Different types of streams:** The proposed techniques, both the training-time and test-time ones, are evaluated using sub-band streams and noise-specific streams. Analysis of proposed multi-stream system, with other kinds of streams (e.g. temporal modulation streams, spectro-temporal modulation streams) is required.

**Hyper-parameter settings for autoencoders:** Comparison experiments of various PM techniques showed that autoencoders based performance monitor is the best performing measure. However the measure is observed to sensitive to hyper-parameter settings used to train autoencoders. Stabilizing the autoencoders is necessary to quickly adapt to different databases.

**Exploring different types of autoencoders:** Modeling long-term posterior trajectories is performed by simply concatenating the posterior frames. Recurrent architectures might be better suited for this task as they are more flexible and can ingest large amounts of data more effectively. Also, in this thesis we used mean-squared-error as the cost function to train the autoencoders. Other ways of modeling the posteriors might further improve the performance of autoencoders, such as variational autoencoders, generative adversarial networks etc.

# Bibliography

[1] H. Hermansky, S. Tibrewala, and M. Pavel, "Towards ASR on partially corrupted speech," in *Proceedings of ICSLP*. ESCA, 1998.

[2] E. Variani and H. Hermansky, "Estimating classifier performance in unknown noise," in *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, 2012, pp. 1800–1803. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2012/i12_1800.html

[3] H. Hermansky, E. Variani, and V. Peddinti, "Mean temporal distance: Predicting asr error from temporal properties of speech signal," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 7423–7426.

[4] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach

to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 2, pp. 179–190, March 1983.

[5] F. Jelinek, L. Bahl, and R. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Inf. Theor.*, vol. 21, no. 3, pp. 250–256, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1975.1055384

[6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[7] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987. [Online]. Available: http://dx.doi.org/10.1109/TASSP.1987.1165125

[8] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, May 1995, pp. 181–184 vol.1.

[9] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modelling," *Computer Speech & Language*, vol. 8, no. 1, pp. 1–38, 1994.

[10] B. Lowerre, "The harpy speech understanding system," in *Readings in speech recognition*. Morgan Kaufmann Publishers Inc., 1990, pp. 576–586.

[11] H. Ney and S. Ortmanns, "Progress in dynamic programming search for lvcsr," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1224–1240, 2000.

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

[13] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[14] H. Hermansky and N. Morgan, "Rasta processing of speech," *IEEE transactions on speech and audio processing*, vol. 2, no. 4, pp. 578–589, 1994.

[15] N. Moritz, M. R. Schädler, K. Adiloglu, B. T. Meyer, T. Jürgens, T. Gerkmann, B. Kollmeier, S. Doclo, and S. Goetze, "Noise robust distant automatic speech recognition utilizing nmf based source separation and auditory feature extraction," *Proc. of CHiME*, pp. 1–6, 2013.

[16] N. Mesgarani, M. Slaney, and S. A. Shamma, "Discrimination of speech

from nonspeech based on multiscale spectro-temporal modulations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 920–930, 2006.

[17] R. Drullman, J. M. Festen, and R. Plomp, "Effect of temporal envelope smearing on speech reception," *The Journal of the Acoustical Society of America*, vol. 95, no. 2, pp. 1053–1064, 1994.

[18] ——, "Effect of reducing slow temporal modulations on speech reception," *The Journal of the Acoustical Society of America*, vol. 95, no. 5, pp. 2670–2680, 1994.

[19] T. Arai, M. Pavel, H. Hermansky, and C. Avendano, "Intelligibility of speech with filtered time trajectories of spectral envelopes," in *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, vol. 4. IEEE, 1996, pp. 2490–2493.

[20] R. V. Shannon, F.-G. Zeng, V. Kamath, J. Wygonski, and M. Ekelid, "Speech recognition with primarily temporal cues," *Science*, vol. 270, no. 5234, p. 303, 1995.

[21] Y. Ephraim, "On minimum mean square error speech enhancement," in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. IEEE, 1991, pp. 997–1000.

BIBLIOGRAPHY

[22] A. G. Adami, L. Burget, S. Dupont, H. Garudadri, F. Grezl, H. Hermansky, P. Jain, S. S. Kajarekar, N. Morgan, and S. Sivadas, "Qualcomm-icsi-ogi features for asr." in *INTERSPEECH*, 2002.

[23] S. Ganapathy, S. Thomas, and H. Hermansky, "Temporal envelope compensation for robust phoneme recognition using modulation spectrum," *The Journal of the Acoustical Society of America*, vol. 128, no. 6, pp. 3769–3780, 2010.

[24] D.-S. Kim, S.-Y. Lee, and R. M. Kil, "Auditory processing of speech signals for robust speech recognition in real-world noisy environments," *IEEE Transactions on speech and audio processing*, vol. 7, no. 1, pp. 55–69, 1999.

[25] A. M. A. Ali, J. Van der Spiegel, and P. Mueller, "Robust auditory-based speech processing using the average localized synchrony detection," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 279–292, 2002.

[26] U. H. Yapanel and J. H. Hansen, "A new perceptually motivated mvdr-based acoustic front-end (pmvdr) for robust automatic speech recognition," *Speech Communication*, vol. 50, no. 2, pp. 142–152, 2008.

[27] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regres-

sion for speaker adaptation of continuous density hidden markov models," *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[28] M. J. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[29] J. Wu and Q. Huo, "Supervised adaptation of mce-trained cdhmms using minimum classification error linear regression," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1. IEEE, 2002, pp. I–605.

[30] X. He and W. Chou, "Minimum classification error linear regression for acoustic model adaptation of continuous density hmms," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 1. IEEE, 2003, pp. I–I.

[31] K. Yu, M. Gales, and P. C. Woodland, "Unsupervised adaptation with discriminative mapping transforms," *IEEE transactions on audio, speech, and language processing*, vol. 17, no. 4, pp. 714–723, 2009.

[32] L. Wang and P. Woodland, "Mpe-based discriminative linear transform for speaker adaptation," in *Acoustics, Speech, and Signal Processing,*

*2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 1. IEEE, 2004, pp. I–321.

[33] M. J. F. Gales, "Model-based techniques for noise robust speech recognition," Ph.D. dissertation, University of Cambridge Cambridge, 1995.

[34] P. J. Moreno, "Speech recognition in noisy environments," 1996.

[35] L. Deng, A. Acero, M. Plumpe, and X. Huang, "Large-vocabulary speech recognition under adverse acoustic environments." in *INTERSPEECH*, 2000, pp. 806–809.

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

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

[38] 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.

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

[40] S. Kim, B. Raj, and I. Lane, "Environmental noise embeddings for robust speech recognition," *arXiv preprint arXiv:1601.02553*, 2016.

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

[42] H. Hermansky, "Multistream recognition of speech: Dealing with unknown unknowns," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1076–1088, 2013.

[43] H. Hermansky, S. Tibrewala, and M. Pavel, "Towards asr on partially corrupted speech," in *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, vol. 1, Oct 1996, pp. 462–465 vol.1.

[44] S. Tibrewala and H. Hermansky, "Sub-band based recognition of noisy speech," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, Apr 1997, pp. 1255–1258 vol.2.

[45] H. Bourlard, S. Dupont, H. Hermansky, and N. Morgan, "Towards

subband-based speech recognition," in *1996 8th European Signal Processing Conference (EUSIPCO 1996)*, Sept 1996, pp. 1–4.

[46] H. Bourlard and S. Dupont, "Subband-based speech recognition," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, Apr 1997, pp. 1251–1254 vol.2.

[47] T. Sporer and K. Brandenburg, "Constraints of filter banks used for perceptual measurement," *Journal of the Audio Engineering Society*, vol. 43, no. 3, pp. 107–116, 1995.

[48] A. Morris, A. Hagen, H. Glotin, and H. Bourlard, "Multi-stream adaptive evidence combination for noise robust asr," *Speech Communication*, vol. 34, no. 1, pp. 25–40, 2001.

[49] F. Li, S. H. R. Mallidi, and H. Hermansky, "Phone recognition in critical bands using sub-band temporal modulations," in *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, 2012, pp. 1816–1819. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2012/i12_1816.html

[50] M. Harper, "The automatic speech recognition in reverberant environ-

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

[51] C. Cieri, D. Miller, and K. Walker, "The fisher corpus: a resource for the next generations of speech-to-text."

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

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

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

[55] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A.-r. Mohamed, G. Dahl, and B. Ramabhadran, "Deep convolutional neural networks for large-scale speech tasks," *Neural Networks*, vol. 64, pp. 39–48, 2015.

[56] S. Tibrewala and H. Hermansky, "Multi-stream approach in acoustic modeling," 1997.

[57] H. Bourlard, S. Dupont, H. Hermansky, and N. Morgan, "Towards subband-based speech recognition," in *Proceedings of EUSIPCO*. EURASIP, 1996.

[58] S. Tibrewala and H. Hermansky, "Sub-band based recognition of noisy speech," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '97, Munich, Germany, April 21-24,*, 1997, pp. 1255–1258.

[59] A. C. Morris, A. Hagen, H. Glotin, and H. Bourlard, "Multi-stream adaptive evidence combination for noise robust ASR," *Speech Communication*, vol. 34, no. 1-2, pp. 25–40, 2001.

[60] H. Hermansky and S. Sharma, "Traps – classifiers of temporal patterns," in *IN PROCEEDINGS OF 5TH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING, ICSLP 98*, 1998, pp. 1003–1006.

[61] H. Hermansky and P. Fousek, "Multiresolution RASTA filtering for TANDEM-based asr," in *Proc. of Interspeech 2005*, 2005, pp. 361–364.

[62] B. T. Meyer and B. Kollmeier, "Optimization and evaluation of gabor feature sets for asr," in *in Proc. Interspeech*, 2008.

[63] M. Kleinschmidt, D. Gelbart, M. Kleinschmidt, D. Gelbart, D. Technische, D. T. Modalitsspezifische, M. K. B, and D. Gelbarta, "Improving word accuracy with gabor feature extraction," in *in ICSLP*, 2002.

[64] N. Mesgarani, S. Thomas, and H. Hermansky, "Adaptive stream fusion in multistream recognition of speech," in *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, 2011, pp. 2329–2332. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2011/i11_2329.html

[65] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[66] N. Parihar and J. Picone, "Aurora working group: DSR front end LVCSR evaluation," Tech. Rep., 2002.

[67] T. Ogawa, F. Li, and H. Hermansky, "Stream selection and integration in multistream asr using gmm-based performance monitoring." in *INTERSPEECH*, 2013, pp. 3332–3336.

[68] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, 2010, pp. 249–256. [Online]. Available: http://www.jmlr.org/proceedings/papers/v9/glorot10a.html

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[70] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[71] "Assessment for automatic speech recognition: Ii. noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Communication*, vol. 12, no. 3, pp. 247 – 251, 1993.

[72] A.-r. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *Acoustics, Speech and Sig-*

*nal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4273–4276.

[73] S. Okawa, E. Bocchieri, and A. Potamianos, "Multi-band speech recognition in noisy environments," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, vol. 2, May 1998, pp. 641–644 vol.2.

[74] H. Misra, H. Bourlard, and V. Tyagi, "New entropy based combination rules in hmm/ann multi-stream asr," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 2, April 2003, pp. II–741–4 vol.2.

[75] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4, pp. 291–294, 1988.

[76] B. Yegnanarayana and S. Kishore, "Aann: an alternative to {GMM} for pattern recognition," *Neural Networks*, vol. 15, no. 3, pp. 459 – 469, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608002000199

[77] K.-P. Li and J. E. Porter, "Normalizations and selection of speech segments for speaker recognition scoring," in *Acoustics, Speech, and Signal Process-*

*ing, 1988. ICASSP-88., 1988 International Conference on*. IEEE, 1988, pp. 595–598.

[78] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, no. 1, pp. 42–54, 2000.

# Vita

Sri Harish Mallidi received his Bachelor of Technology and Master of Science by Research degrees in Electronics and Communications Engineering from International Institute of Information Technology, Hyderabad, India in 2010. He completed his PhD. Electrical and Computer Science while being affiliated to the Center for Language and Speech Processing (CLSP) at The Johns Hopkins University in 2018. He is currently a Research Scientist at Amazon, Seattle, USA. His research interests include machine learning, automatic speech recognition, speech signal processing, language identification, speaker recognition. In the past he has been part of several summer workshops at the CLSP and has also worked at Brno University of Technology, Brno, CZ and Raytheon BBN Technologies, Cambridge, MA.

VITA