Brno University of Technology
Faculty of Information Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

HABILITATION THESIS

Temporal processing for feature extraction in speech recognition

Jan Černocký
Department of Computer Graphics and Multimedia

October 2002

Abstract

Speech recognition is a booming research field, having large number of applications in telecommunications (especially mobile), automobile industry, consumer electronics, military and security, etc. Speech recognition systems are classically built from three basic blocks: feature extraction, acoustic matching and language modeling. While the last two are trained on data (annotated databases for acoustics and large speech corpora for the LM), feature extraction block is often neglected and most often, mel-frequency cepstral coefficients (MFCC) are used. This work concentrates on two techniques that should improve the feature extraction.

The first is temporal filtering of feature trajectories using filters designed on data using Linear Discriminant Analysis (LDA). This technique is shown to improve the recognition accuracy of isolated Czech words, confirming previous results on US-English obtained by our colleagues from OGI Portland.

The second part of the work concentrates on more revolutionary approach of feature extraction using TRAPs (temporal patterns) whose fundamentals were also laid at OGI. Several experiments were conducted on three databases during author's stay at OGI. Although we have shown that TRAPs are comparable to MFCC's only on a small vocabulary recognition task, we believe that combination of frequency-band processing and neural nets will become very important in the next decade, and that they will become standard blocks of feature extraction.

A conclusion chapter is included for both methods, giving directions of current and future work both at OGI Portland and VUT Brno.

Abstrakt

Rozpoznávání řeči je rychle se rozvíjejícím oborem s množstvím aplikací v telekomunikacích (zvláště mobilních), automobilovém průmyslu, spotřební elektronice, vojenské a bezpečnostní oblasti, atd. Rozpoznávače řeči se klasicky skládají ze tří základních bloků: výpočtu příznaků (parametrizace), akustického srovnávání a jazykového modelu. Zatímco poslední dva bloky jsou trénovány na datech (akustika na anotovaných řečových databázích, LM na korpusech textových dat), parametrizace je často zanedbávána a na vstupech rozpoznávačů najdeme nejčastěji mel-frekvenční cepstrální koeficienty (MFCC). Tato práce se zaměřuje na dvě techniky, které by měly parametrizaci zkvalitnit.

První z nich je časová filtrace trajektorií parametrů pomocí LDA-filtrů. Tyto jsou získány z řečových dat pomocí Lineární diskriminační analýzy (LDA). V práci ukážeme, že tato technika zlepšuje úspěšnost rozpoznávače při rozpoznávání izolovaných českých slov. Potvrdili jsme tak předchozí výsledky na americké angličtině, získané naší partnerskou skupinou na OGI Portland.

Druhá část práce se zaměřuje na "revolučnější" přístup k parametrizaci pomocí časových trajektorií (TRAPs). Základ této metody byl rovněž položen skupinou na OGI a experimenty popsané v této práci byly provedeny během autorova sedmiměsíčního pobytu v Portlandu. I když jsme prokázali, že TRAP-příznaky jsou srovnatelné s MFCC pouze na rozpoznávání omezeného souboru slov, věříme, že kombinace zpracování v jednotlivých kmitočtových pásmech s neuronovými sítěmi nabude v následující dekádě na důležitosti a žs se tyto techniky stanou standardními bloky v parametrizaci řeči.

K oběma popsaným metodám jsou připojeny kapitoly obsahující závěry a popisující současný stav řešení problematiky a další výzkumné směry.

Contents

List of Figures

Chapter 1

Introduction

This thesis is proposed in order to obtain the “associate professor” degree at Brno University of Technology, Faculty of Information Technology. Its main topic are temporal and data-driven methods for feature extraction in speech recognition. Rather than an individual research work, this is a topic worked on by several of my pre- and post-graduate students as well as students in our partner laboratory at OGI Portland. The following section is devoted to ’credits’ to all the people that have contributed their results and figures to the text.

1.1 Acknowledgments

The first thanks should go to my tutors in speech and signal processing, starting at VUT Brno with Vladimír Šebesta and Milan Sigmund, through Geneviève Baudoin and Gérard Chollet at ESIEE and ENST Paris, till Hynek Hermansky (OGI Portland and VUT Brno). I am specially grateful to Hynek for having invited me to spend 7 months in his US laboratory last year. Most of the material on TRAPs (chapters 5–8) originated there. I am also grateful for challenging discussions with him, insisting always on the question “why we are doing that” rather than “what we are doing”.

Second series of thanks goes to postgraduate students at the Dpt. of Computer Graphics and Multimedia (from seniors to juniors: Lukáš Burget, Petr Motlíček, Franta Grézl, Petr Schwarz, Martin Karafiát, Petr Jenderka and Tomáš Vícha) and at FIT and Inst. of Radioelectronics of FEEC (Pavel Matějka). Special thanks go to Lukáš, for good discussions we’ve had together and also for part of the introductory material on HMM recognition (section 1.3) from his “concepts of dissertation” [4]. Franta Grézl [10] was “responsible” for some of the TRAP figures and results and especially for the new approaches like “LDA-TRAP” and “merger-only” that are only briefly mentioned in this thesis (chapter 9). Thanks also to Peťa Schwarz for contributing the SpeechDat-E recognizer [30]. Martin Karafiát contributed the context-dependent HMMs [22] used in forced alignment of SpeechDat-E.

But as important was to be in contact with Hynek’s students at OGI last year: Pratibha Jain, Sachin Kajarekar, Sunil Sivadas, and Andre Adami. Pratibha was the most helpful, because she has been (and still is) the ’TRAP-girl’ at OGI and she has helped me a lot with TRAP experiments.

Among the pre-graduate students, Jiří Kafka (graduated June 2002) deserves great thanks for his diploma work [20] on LDA-filters tested on the recognition of Czech. His results are the core of the LDA experimental chapter 3.

The greatest thanks go of course to my wife Hanka for a steady support, here in Brno as well as in US last year, and the little Tomášek, and newborn Adámek for some distraction and lots of fun.

1.2 Scope of chapters

This work is divided into 10 chapters. The one you are reading now, contains small review of speech recognition using Hidden Markov models and a section introducing speech feature extraction.

Chapters 2–4 deal with filtering of feature trajectories using filters derived by Linear discriminant analysis: 2 contains some theoretical background, and compares LDA-filters to more commonly known RASTA-filtering, 3 describes experiments conducted on Czech SpeechDat-E database (with filters trained on US-English or Czech) and 4 gives a summary on temporal filtering.

Chapters 5–9 cover using of Temporal trajectory classifiers (TRAPs) as front-end to HMM-based speech recognizer. Chapter 5 will give some basis of TRAPs and describe the system we have used. Chapter 6 describes experiments on the trinity of databases Stories–Numbers–Digits (SND), used in previous work by Sharma and Jain. The following chapter 7 describes experiments performed on the new set of databases with good phonetic coverage. Chapter 8 presents running TRAPS on the SPINE task. Chapter 10 contains summary of the work being currently done in TRAP field by Pratibha Jain and Franta Grézl, and some conclusions.

Chapter 10 concludes this thesis.

1.3 Speech recognition using Hidden Markov Models

Most of current systems [9] [29] for automatic speech recognition consist of three basic function blocks:

- **Feature Extraction -** In this phase speech signal is converted into stream of feature vectors - coefficients which contain only that information about given utterance that is important for its correct recognition. Parameterization is performed for a size reduction of original speech signal data and for preprocessing of that signal into a form fitting requirements of following classification stage. An important property of feature extraction is the suppression of information irrelevant for correct classification, such as information about speaker (e.g. fundamental frequency) and information about transmission channel (e.g. characteristic of a microphone). Currently the most popular features are Mel frequency cepstral coefficients MFCC [5].

- **Classification -** The role of classifier is to find a mapping between sequences of speech feature vectors and recognized fundamental speech elements (words in a vocabulary, phonemes). This mapping can be done for example by simple recognizer based on Dynamic Time Warping (DTW), where the sequences of parameter vectors are stored as references. Word parameters are then compared directly with the references. More advanced classifiers are mostly based on Hidden Markov Models (HMM) [35], where parameters of statistical models are estimated using training utterances and their associated transcriptions. After this process, the well trained models can be used for recognition of unknown utterances. The output of the classifier is a set of possible sequences of speech elements (hypotheses) and their probabilities.

- **Language models -** The role of language models is selection of a hypothesis which is most likely the right sequence of speech elements (sentence) of a given language. The complexity of language model depends on complexity of the problem being solved (continuous speech vs. limited number of commands). Statistical models derived from data are also very often used for this purpose (N-grams). Interested reader can be referred to [19].

In the HMM recognition system [35], it is assumed that the sequence of speech parameter vectors $\mathbf{O}$ corresponding to each unknown utterance is generated by a Markov Model. In the recognition step, the likelihoods of this vector sequence $\mathbf{O}$ generated by individual models are computed, and the word represented by the model with the best likelihood is recognized. Hidden Markov model can be seen as a finite state machine which changes state once every time unit. Each time t that state j is entered, a

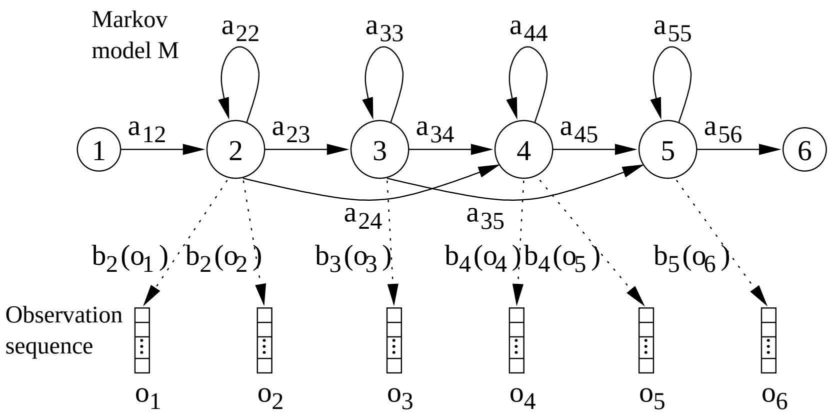

Figure 1.1: The Markov Generation Model

speech parameter vector $\mathbf{o}_t$ is generated from the output probability density $b_j(\mathbf{o}_t)$ and the transition from state i to state j is governed by the probability a_{ij}. The joint probability that vector sequence $\mathbf{O}$ is generated by model M moving through the state sequence X is calculated simply as a product of the transition probabilities and the output probabilities. So for the state sequence X in Figure 1.1,

$$P(\mathbf{O}, X|M) = b_1(\mathbf{o}_1)a_{11}b_2(\mathbf{o}_2)a_{12}b_2(\mathbf{o}_3)\dots \tag{1.1}$$

However, in practice, only the vector sequence $\mathbf{O}$ is known and the state sequence X is hidden. Therefore the required likelihood is computed by summing over all possible state sequences. Most HMM systems represent output distributions by Gaussian mixture densities. The formula for computing $b_j(\mathbf{o}_t)$ is then

$$b_j(\mathbf{o}_t) = \sum_{i=1}^{M} c_{ji} N(\mathbf{o}_t, \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_{ji}), \tag{1.2}$$

where M is the number of mixture components, c_i is the weight of the i-th component and $N(\mathbf{o}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the output value of a multivariate Gaussian with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, that is

$$N(\mathbf{o}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{o}-\boldsymbol{\mu})} \tag{1.3}$$

Instead of full covariance matrix, only its diagonal vector representing variances is often stored. Other matrix elements are assumed to be zero (this supposes that individual elements of the parameter vector is not correlated).

Training of individual models consists of three steps:

- dividing of training sequences of speech parameter vectors into states of a corresponding model,
- estimation of output probability densities. In case of Gaussian Mixture Densities it means estimation of mean vectors and variance vectors.
- estimation of transition probabilities.

At the end of training, we dispose of a trained model of each acoustical unit, which may be a whole word, phoneme, or phoneme in context (eg. triphone).

The simplest case of the **recognition** is an isolated-word recognition. Here, an unknown sequence $\mathbf{O}$ is matched against all the models M_i and label of model giving maximum output probability (only the probability of the best path, so called Viterbi one) is selected as the recognizer output. In continuous speech recognition, the acoustical models form a network and the task of the recognizer (often called “decoder”) is to find the optimal path through this network for given sequence $\mathbf{O}$. This search can be aided by a language model.

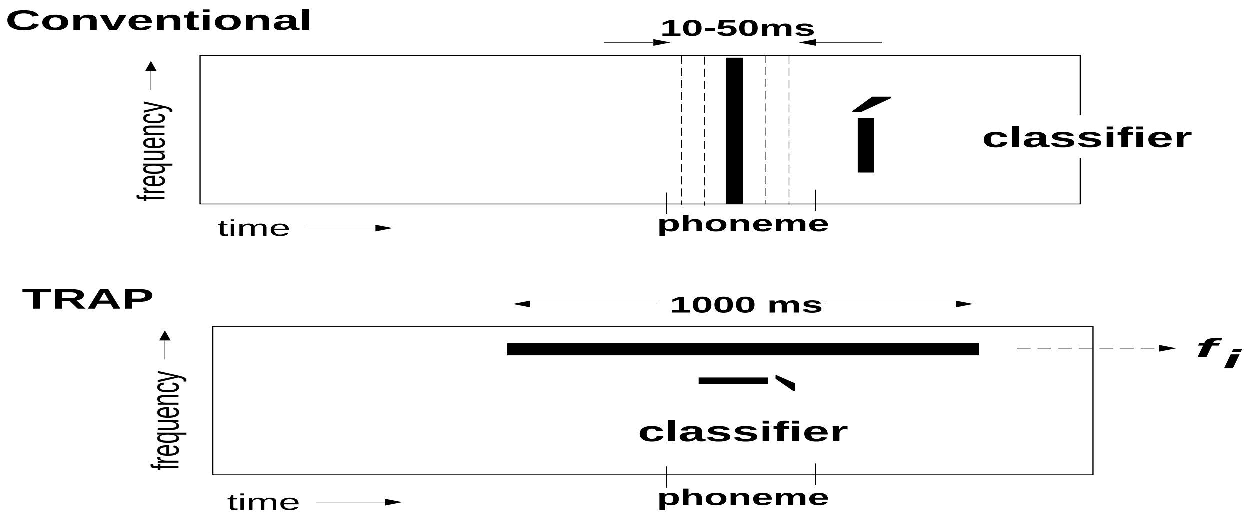

Figure 1.2: "Classical" and LDA-TRAP systems.

1.4 Features for HMM speech recognition

While the acoustical matching (HMM) and decoding (Viterbi algorithm with a pronunciation dictionary and language model) are heavily trained on data, the feature extraction is often considered as "given" and neglected [11]. In the following chapters we will concentrate on temporal processing of features and on some pre-classification that brings the features more closed to the data.

HMMs have one substantial drawback - one HMM state "sees" only the current speech frame and it does not know, what happened before and what will happen later in the feature matrix. Delta and delta-delta features introduced in the 70-ies, have added some notion of trends in feature space. Note, that computation of Δs and $\Delta\Delta$s can be also interpreted as temporal filtering. Hermansky and Morgan, introduced RASTA filtering [13], being inspired by some properties of human auditory periphery, namely by the insensibility to too high and too low frequencies in modulation spectrum. RASTA processing has made especially recognizers based on context-independent phonemes far more robust than their non-filtered counterparts.

In RASTA, the characteristics of filter were "tuned" to match some auditory properties, but it was not shown to be optimal. From the classifier theory, we dispose however of some mathematical tools to design optimal projection of parameter space in order to preserve discriminability. As linear filtering is nothing but projection of the original temporal trajectory on the reversed impulse response of the filter, a filter can be designed using this theory. In chapters 2–4, we will see an application of LDA (linear discriminant analysis) derived filters to the recognition of Czech.

Till now, we are however still limited to feature-preparation using linear processing. Non-linear methods, especially Neural Nets [3] have been widely used in speech processing, mostly to derive class-posteriors for the Viterbi algorithm (they are replacing mixtures of Gaussians). On the other hand, one would like to use NNs directly to derive features and not to "touch" the recognizer itself – the Gaussian mixture modeling (GMM) is nowadays the most wide-spread technique to model the distribution of features in states (this inclination is also given by the popular HTK toolkit, using GMMs). The Tandem-approach or Feature-Net overcomes this barrier by processing the feature stream by NNs, but using the output likelihoods (after some post-processing, as Gaussianization and de-correlation) as input to standard GMM-HMM recognizer [12].

The classification of Temporal Patterns (TRAPs) using NNs, first introduced by Sharma [32], is a natural step in combining the temporal processing and non-linear classification into the feature-extraction block. Unlike conventional recognizers (Fig. 1.2) where the recognition is done of a "instantaneous cut" of the entire speech spectrum, TRAP classifiers detect acoustical units out of long temporal trajectories in each frequency band, and then the results are combined using a merging network. We hope that if the speech is corrupted in one more frequency bands, the merger will still be able to obtain a more reliable output. This idea is similar to multi-band recognizers promoted by Bourlard and Morgan [2], but in our case, we are not forced to "touch" the recognizer's architecture – the robustness should be achieved by the feature extractor itself.

Chapter 2

Data driven features for speech processing

The purpose of feature extraction is a reduction of speech data size and other processing required for an adaptation of these data for classifier (HMM). The standard way of feature extraction consists of the following steps:

- **Segmentation** – Speech signal is divided to segments where the waveform can be regarded as stationary (the typical duration 25 ms). The classifiers generally assume that their input is a sequence of discrete parameter vectors where each parameter vector represents just one such segment - frame.
- **Spectrum** – Current methods of a feature extraction are mostly based on the short term Fourier spectrum and its changes in the time, therefore the power or magnitude Fourier spectrum is computed in the next step for every speech segment.
- **Auditory-like modifications** – Modifications inspired by physiological and psychological findings about human perception of loudness and different sensitivity for different frequencies are performed on spectra of each speech frame.
- **Decorrelation** – Some technique for vector decorrelation is used for a better adaptation of features to requirements of classifier. In the case of HMM, only a variance vector can be used for a description of output probabilities instead of a full covariance matrix (section 1.3).
- **Derivatives** – Feature vectors are usually completed by first and second order derivatives of their time trajectories (delta and acceleration coefficients). These coefficients describe changes and speed of changes of the feature vector in the time.

2.1 Mel frequency cepstral coefficients

Mel frequency cepstral coefficients (MFCC)[5] are a commonly used feature extraction method. Brief description of this method is presented here because MFCC are used as starting point for modifications described in section 2.2. Particular steps of this method are shown on the block scheme in figure 2.1. Output vectors for every step for one segment of a voiced speech (vowel 'iy') are demonstrated in figure 2.2.

First, speech samples are divided into overlapping frames. The usual frame length is 25 ms and the frame rate is 10 ms. Example of one such frame for English vowel 'iy' can be seen in figure 2.2a. First, a pre-emphasis is applied to the framed speech signal: $y(n) = x(n) - 0.97x(n-1)$. This filter amplifies higher frequencies which is an approximation of psychological findings about sensitivity of human

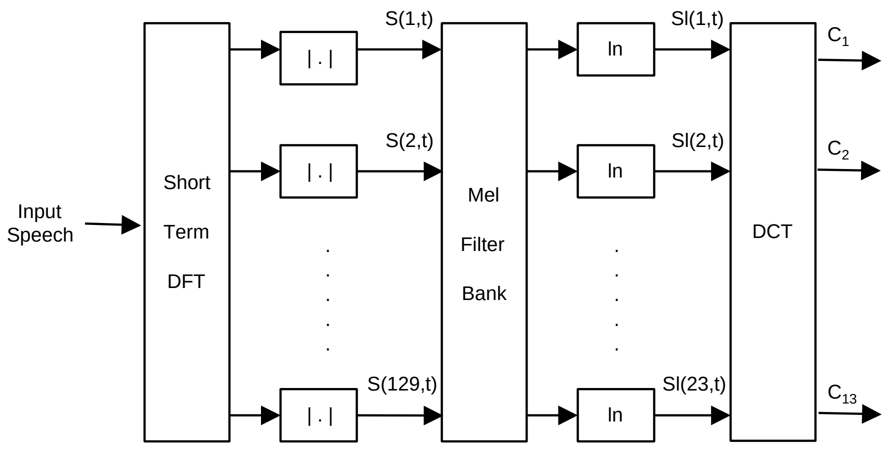

Figure 2.1: Block diagram showing steps of Mel frequency cepstral coefficients computation

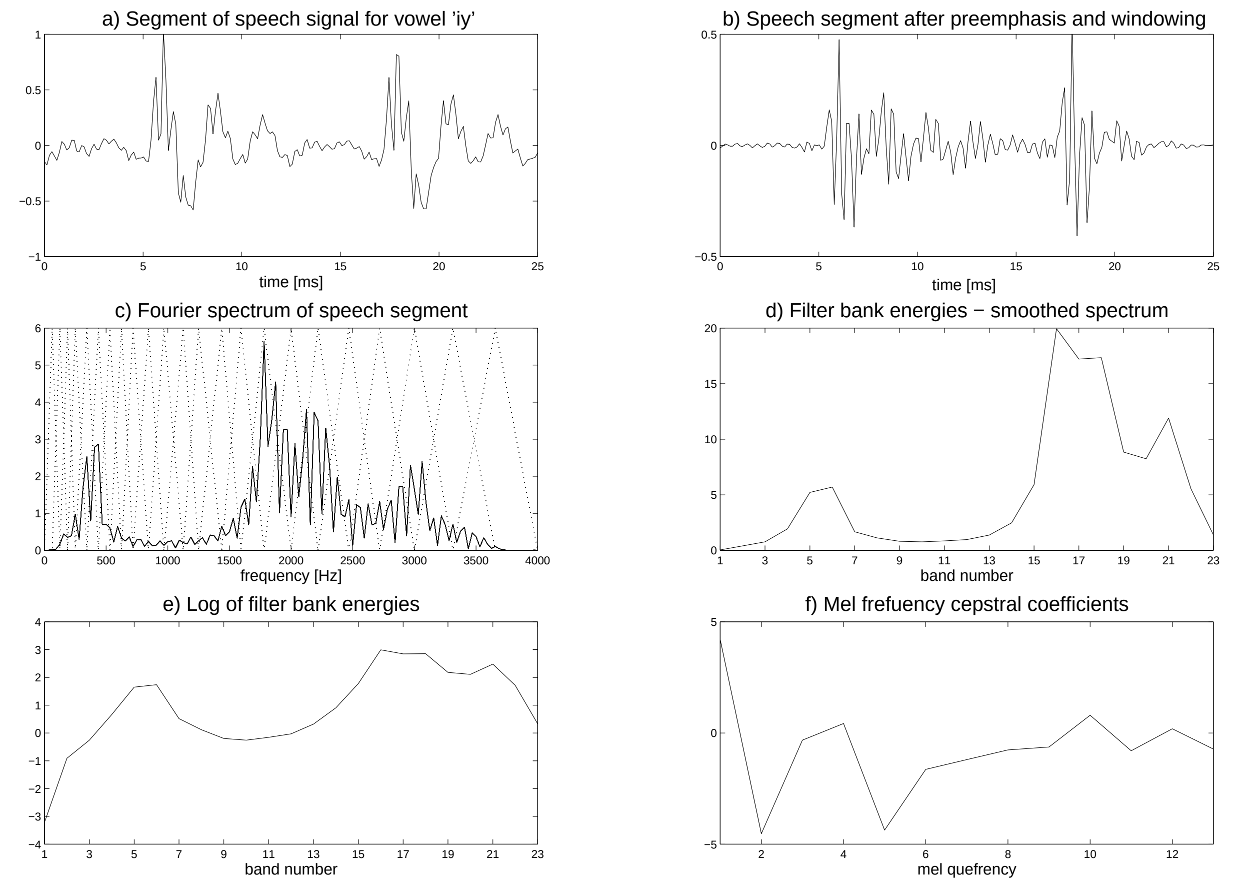

Figure 2.2: Output vectors of particular steps of Mel frequency cepstral coefficients computation

hearing for different frequencies. The Hamming window is applied in the next step (figure 2.2b) and magnitude Fourier spectrum is computed for this windowed frame signal (figure 2.2c). A filter bank is then applied for modification of the magnitude spectrum. Energies in the spectrum are integrated by the set of a band limited triangular weighting functions. Their shape can be seen in figure 2.2c (dotted lines). These weighting functions are equidistantly distributed over Mel scale according to psycho-acoustic findings where better resolution in spectrum is preserved for lower frequencies than for higher frequencies. A vector of the filter bank energies for one frame can be seen as a smoothed and down-sampled version of spectrum (figure 2.2d). The log of integrated spectral energies is taken with agreement to the human perception of sound loudness (figure 2.2e). Temporal trajectories of the log energies of each band can be optionally filtered by RASTA-like bandpass filters [13]. The low pass character of such filter allows to remove fast energy changes which cannot be produced by the human articulatory tract. The high pass character of the filter is responsible for removing a static information about a channel, since it appears as an additive constant to the filter bank band output in the log domain. The feature vector is finally decorrelated and its dimensionality is reduced by its projection to several first cosine basis (Discrete Cosine Transform). A discussion of usage of DCT can be found in section 2.2.1 dealing with Principal Component Analysis.

2.2 Data driven feature extraction methods

While classification and language models are usually based on stochastic approaches where models are trained on data, feature extraction is generally based on knowledge and beliefs. However, since mechanism of the human auditory system is not fully understood, the optimal system for a feature extraction is not known. Moreover, psychoacoustic findings often describe limitations of the human auditory system and we do not know if modeling of those limitations is useful for the speech recognition. Therefore methods of data driven optimizations for some stages of above described standard feature extraction scheme are presented in following sections. The agreement between results of these methods and the psychoacoustic findings is shown. Principal Component Analysis and Linear Discriminant Analysis techniques are used by these optimization methods and they will be described first.

2.2.1 Principal Component Analysis

Principal Component Analysis (PCA) or Karhunen-Loevy transform (KLT) is a technique looking for such linear transform with orthogonal basis where the first base vector shows a direction of the largest variability of training data in N-dimensional space of input vectors. The second base vector than shows a direction perpendicular to direction given by the first vector with the second largest variability and so on. A limitation of this technique is the assumption that input data have Gaussian distribution. This transform has two important properties:

- Elements of outputs vector are decorrelated (their values are not dependent each on other).
- Projection to only several first base vectors can be performed for a dimensional reduction preserving most of a variability of original data.

The figure 2.3 demonstrates the effect of PCA for 2-dimensional data vectors. The gray ellipse represents distribution of data, the axes show the new coordinates (directions) obtained by a rotation of original coordinates using PCA transform. The new distributions of uncorrelated data in both directions are also demonstrated in this figure.

Base vectors of PCA transforms are given by the eigen vectors of a covariance matrix which is computed from training data according to the following equation:

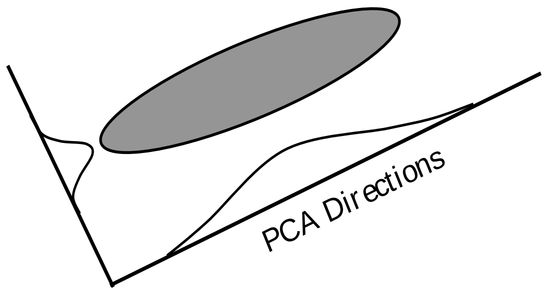

Figure 2.3: Principal Component Analysis for 2-Dimensional Data

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T \tag{2.1}$$

where $\mathbf{C}$ is the covariance matrix derived from N input vectors available for training, $\mathbf{x}_i$ is the i-th training input vectors and $\mathbf{m}$ is the estimated mean vector, that is

$$\mathbf{m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{2.2}$$

The eigen value associated with one eigen vector represents the amount of variability preserved by the projection of input vectors to this particular eigen vector. Therefore only several eigen vectors corresponding to the highest eigen values are used as basis of PCA transform for the purpose of a dimension reduction.

2.2.2 Linear Discriminant Analysis

Analogous to PCA, Linear Discriminant Analysis (LDA) proposed by Hunt [15] is a data driven technique looking for such linear transform allowing a dimension reduction of input data. However, it preserves information important for the linear discrimination among input vectors which belong to different classes. Therefore, unlike the case of PCA, we need also information about the class to which a particular input training vector belongs. The result of LDA are then base vectors of the linear transform sorted by their importance for discriminating among classes. We can therefore pick up only several first basis which preserve almost all the variability in data important for discriminability. Note, that LDA like a PCA ensures decorrelation of transformed data. Moreover, it does not decorrelate only overall training data as it is in the case of PCA, but data belonging to each particular class are also decorrelated. The figure 2.4 demonstrates effect of LDA for 2-dimensional data vectors which belong to two classes. The gray and the empty ellipses represent distributions of data in two different classes with mean vectors $\mathbf{m_1}$ and $\mathbf{m_2}$ and covariance matrices $\mathbf{C_1}$ and $\mathbf{C_2}$. The axes X and Y are coordinates of the original space. Large overlap of the class distributions can be seen in the directions of these original coordinates. The axis Z then shows the direction obtained by LDA in which the classes are well separated. Since this example deals just with two classes and since LDA assumes that distributions of all classes are Gaussian with the same covariance matrix ($\mathbf{C_1} = \mathbf{C_2}$) no other direction can be obtained for a better discrimination.

Base vectors of LDA transforms are given by the eigen vectors of a matrix $\mathbf{AC} \times \mathbf{WC}^{-1}$. The within-class covariance matrix $\mathbf{WC}$ represents unwanted variability in data and is computed as the

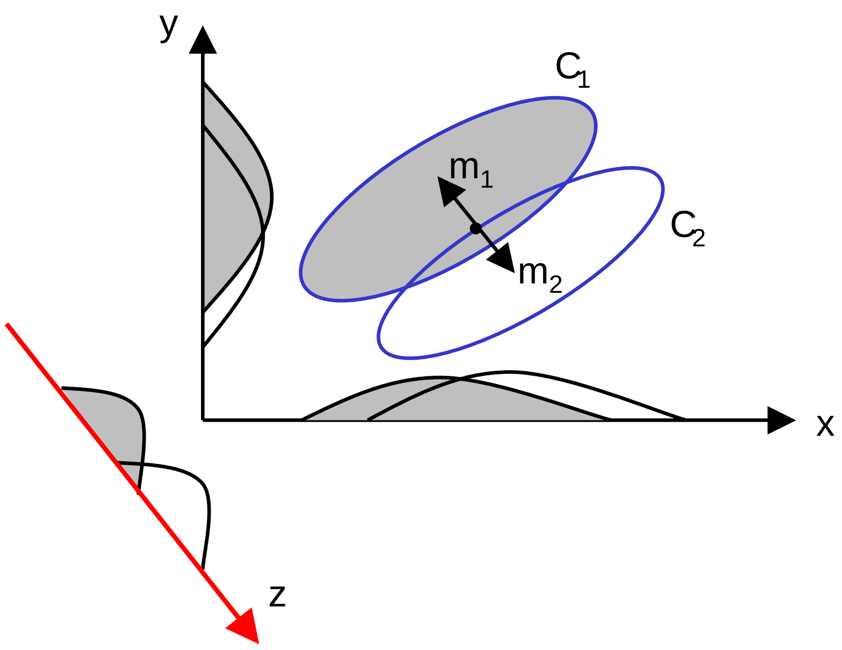

Figure 2.4: Linear Discriminant Analysis for 2-Dimensional Data

average of the covariance matrices of all classes:

$$\mathbf{WC} = \frac{1}{L}\sum_{j=1}^{L}\mathbf{C}_j \tag{2.3}$$

where L is the number of classes and $\mathbf{C}_j$ is the covariance matrix for j-th class:

$$\mathbf{C}_j = \frac{1}{N_j}\sum_{i=1}^{N_j}(\mathbf{x}_i^j - \mathbf{m}_j)(\mathbf{x}_i^j - \mathbf{m}_j)^T \tag{2.4}$$

where N_j is the number of input vectors available for training which belong to class j. $\mathbf{x}_i^j$ is the i-th training input vector belonging to j-th class and $\mathbf{m}_j$ is estimated mean vector for class j:

$$\mathbf{m}_j = \frac{1}{N_j}\sum_{i=1}^{N_j}\mathbf{x}_i^j \tag{2.5}$$

The across-class covariance matrix **AC** represents the wanted variability in data and it is computed as a covariance matrices of weighted mean vectors of all classes:

$$\mathbf{AC} = \frac{1}{N}\sum_{j=1}^{F}N_j(\mathbf{m}_j - \mathbf{m})(\mathbf{m}_j - \mathbf{m})^T \tag{2.6}$$

where N is the total number of all input vectors available for training and $\mathbf{m}$ is the global mean:

$$\mathbf{m} = \frac{1}{N}\sum_{j=1}^{F}N_j\mathbf{m}_j \tag{2.7}$$

An eigen value associated with one eigen vector represents the amount of variability necessary for the discriminability preserved by the projection of input vectors to this particular eigen vector. Only several eigen vectors corresponding to the highest eigen values can be used as LDA transform for the purpose of a dimension reduction.

2.3 Spectral basis derived using Principal Component Analysis

The projection to cosine basis (DCT) is used in the final stage of Mel frequency cepstral coefficients computation. The idea of using DCT allowing dimension a reduction and decorrelation of features is coming from the theory of cepstral analysis (cepstrum = the Fourier spectrum of the log of the magnitude Fourier spectrum) which allows a separation of the input signal and the frequency characteristic of the linear system by which the input is signal processed. In the case of speech, the vocal cords excitation can be seen as the input signal and the vocal tract causing acoustic resonances can be seen as the system. While low cepstral coefficients represent the important information about the system, higher coefficients describe the excitation which is generally considered as unimportant for the recognition (the same speech can be pronounced with different fundamental frequency). Since DCT is applied on the log of a filter bank output in the case of MFCC, where a frequency resolution is already modified for different frequencies, there is no straightforward argument for using DCT in this case. Moreover, there is no intuitive proof of a feature decorrelation using DCT.

An alternative way is the use of Principal Component Analysis for deriving a linear transform replacing DCT. The log of Mel filter bank output vectors of all training data frames is used for computation of a covariance matrix. Example of such covariance matrix derived from speech data from TIMIT database for Mel filter bank with 23 bands is shown in figure 2.5. Values on the matrix diagonal represent variances in particular bands. Values not on the diagonal then represents correlations between bands. Note the large correlation between several neighbor bands that can be seen especially for the lower band numbers.

The figure 2.6 shows eigen values and first 5 eigen vectors of the covariance matrix. The eigen vectors are basis of the transform replacing DCT. For comparison, the cosine basis are plotted (dotted line) together with corresponding base vector derived by PCA (solid line). The eigen values in figure 2.6a indicate that almost all variability in data is preserved by projection to only several first basis. The prominent similarity between PCA and DCT basis can be seen as proof that DCT was not wrong choice for the spectral basis.

2.4 Spectral basis derived using Linear Discriminant Analysis

The projection of the log of filter bank output vectors to the spectral basis derived by PCA (section 2.3) leads to a decorrelation of features. A dimension reduction is also possible since most of the variability is preserved by the projection to only several first spectral basis. However, this still does not ensure that the projection to these several first basis leads to a good separability among classes which should be recognized by classifier. Linear Discriminant Analysis (section 2.2.2) used instead of PCA can be performed to obtain basis which are sorted by their importance for the discriminability among classes.

In our experiments, the log of Mel filter bank output vectors of all training data frames is used for computation of a across-class covariance and within-class covariance matrices. In our speech recognition task, we want to distinguish different phonemes. Therefore the log of filter bank output vectors representing frames labeled by the same phoneme belong to the same class. Examples of such across-class covariance and within-class covariance matrices derived from speech data from TIMIT database for Mel filter bank with 23 bands is shown in figure 2.7.

The figure 2.8 shows first 5 LDA spectral bases given by the eigen vectors of the matrix $\mathbf{AC} \times \mathbf{WC}^{-1}$. For comparison, the cosine basis are plotted (dotted line) together with the corresponding

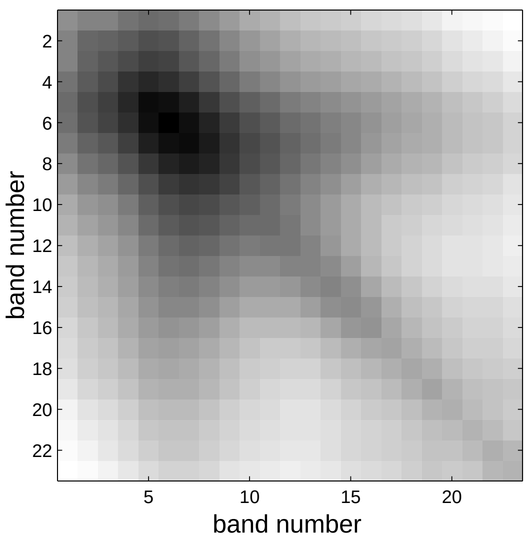

Figure 2.5: Covariance matrix computed from filter bank output vectors

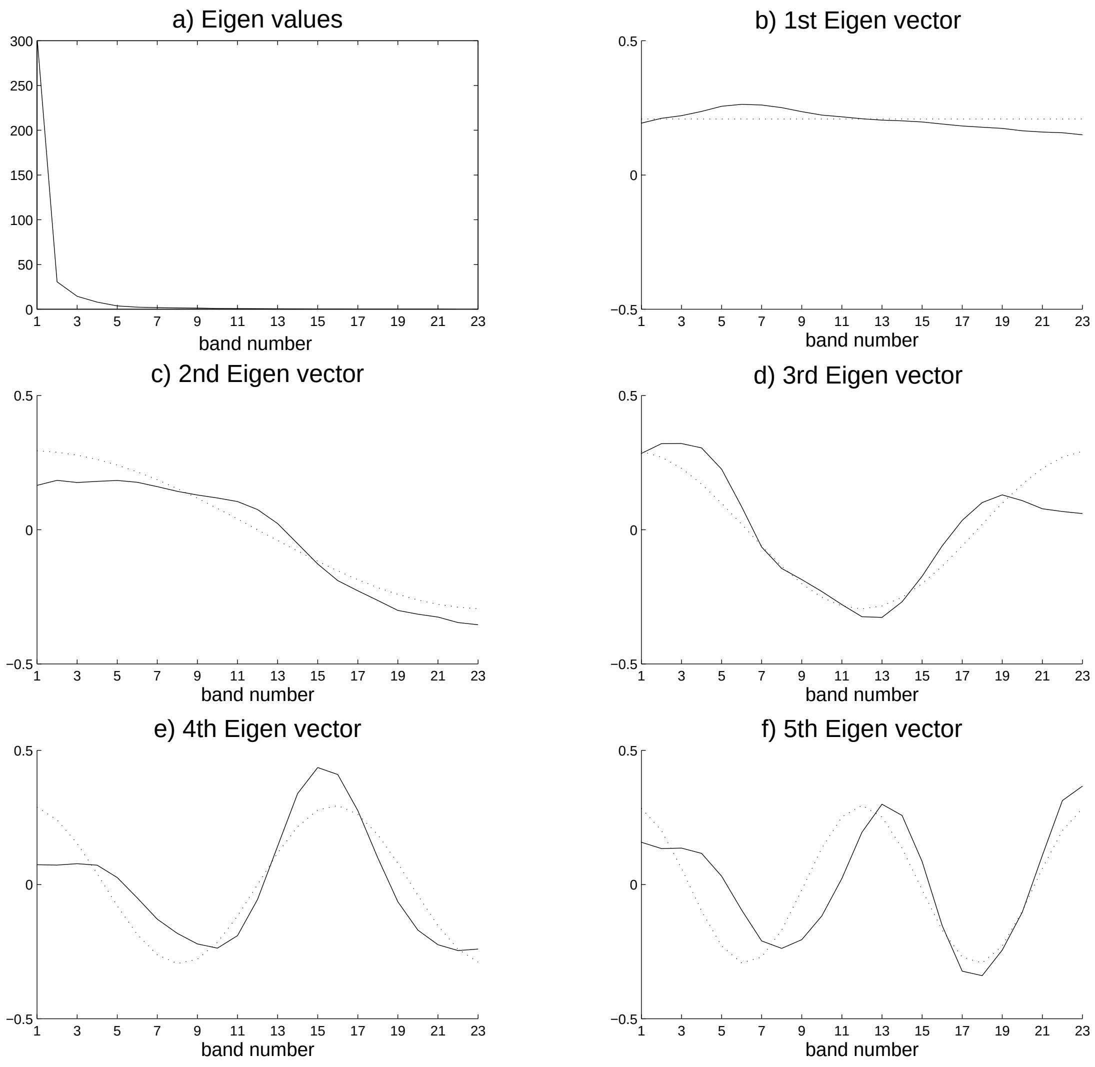

Figure 2.6: Spectral basis derived using PCA

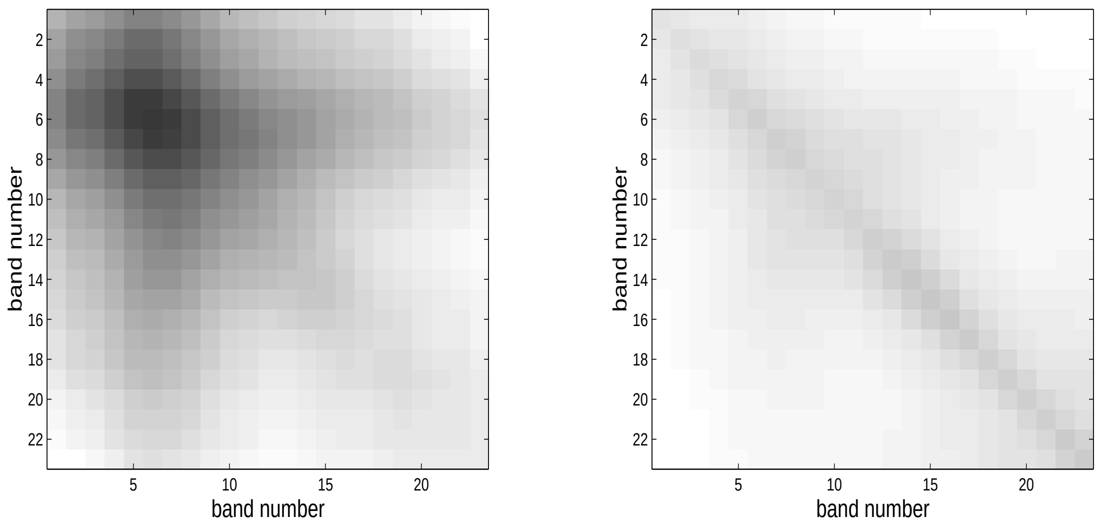

Figure 2.7: Across-class and within-class covariance matrix computed from filter bank output vectors

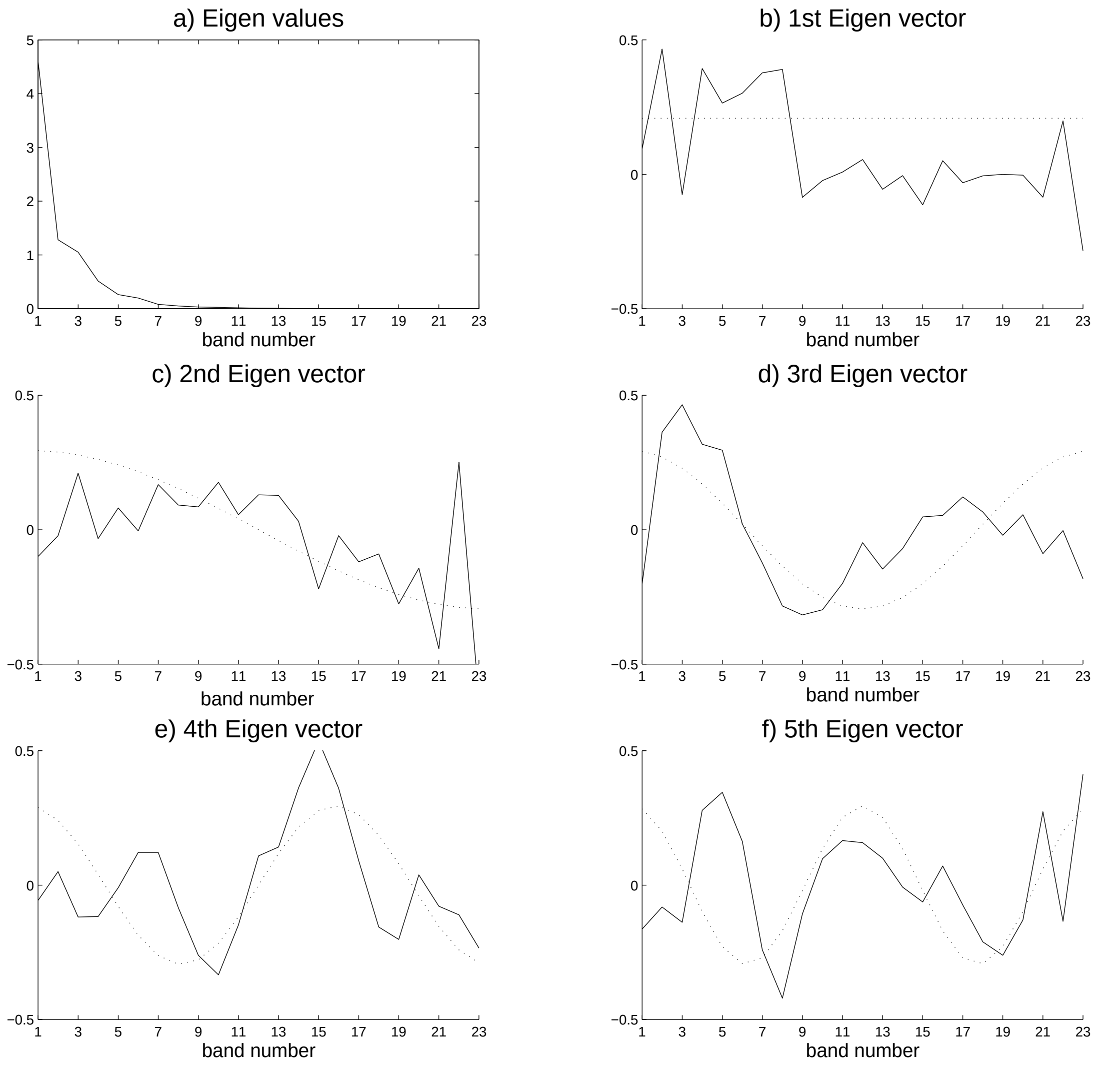

Figure 2.8: Spectral basis derived using LDA

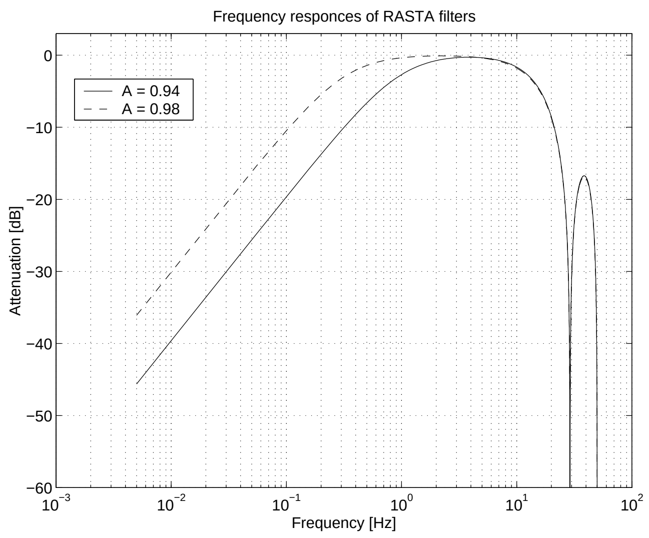

Figure 2.9: Frequency response of RASTA band-pass filter

base vector derived by LDA (solid line). The eigen values in figure 2.8a indicate that almost all variability in data important for a class separability is preserved by the projection to only several first base vectors. Spectral basis derived using LDA are already not so similar to DCT basis (as it was in the case of PCA). This technique is more sensitive to amount of data available for training than PCA and it can be seen that the obtained basis are noisy for this reason. For their smoothing, the projection to several first PCA basis before computing LDA can be performed for removing little bit of variability from data. Note again that the transform derived by LDA (in contrast to PCA) decorrelates also data belonging to individual classes. This is important especially for further processing by HMM classifier where probability distribution functions attached to states of individual models usually model feature vectors belonging to one class (phoneme).

2.5 From RASTA to temporal filters derived using LDA

The original design of those filters [13] for filtering temporal trajectories was inspired by psychological findings about temporal masking and is the core of RASTA algorithm described in [13]. RASTA takes advantage of the fact, that the linguistic message is coded into movements of the vocal tract. It suppresses the spectral components that change more slowly or more quickly than the typical range of change of speech. Temporal trajectories of the log energies of each band are filtered by bandpass filter. The low pass character of such filter allows to remove fast energy changes which cannot be produced by the human articulatory tract. The high pass character of the filter is responsible for removing a static information about a channel, since it appears as an additive constant to the filter bank band output in the log domain. In principle, the RASTA processing can be done on time trajectories of any parameters.

RASTA filter is an IIR filter with the transfer function

$$H(z) = 0.1z^4 \times \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - Az^{-1}} \tag{2.8}$$

where value of A can vary from 0.94 to 0.98. Frequency response for such a filter is in Fig. 2.9.

As mentioned above log energies of each band are filtered by RASTA filter. After filtering, there is inverse logarithmic function for expanding features to original scale. The block scheme of PLP feature extraction with RASTA filtering is shown in Fig. 2.10.

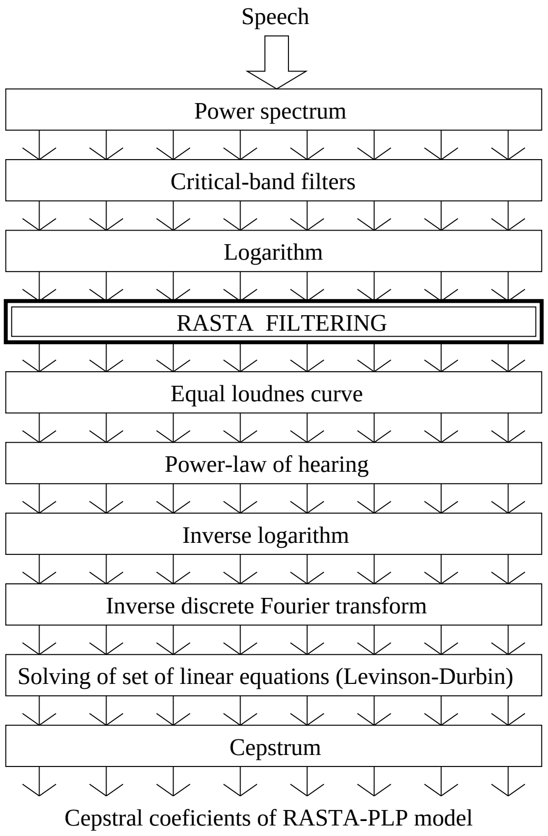

Figure 2.10: Block diagram showing steps of PLP computation with RASTA filtering

Another possibility is to derive impulse responses of such filters using *Linear Discriminant Analysis*. The filters can be derived independently for each band. Vectors which are formed by consecutive values of one band time trajectory are used for the computation of a across-class covariance and within-class covariance matrices. A typical length of the vector corresponds to one second of signal (101 points of a band time trajectory for 10ms frame rate). Vectors with central point representing frames labeled by the same phoneme belong to the same class. Example of across-class covariance and within-class covariance matrices derived this way from OGI-Stories database for one Mel filter bank band output is shown in figure 2.11. The resulting covariance matrices and their eigen vectors are very similar for all bands of the filter bank.

The way how the basis (eigen vectors of $\mathbf{AC} \times \mathbf{WC}^{-1}$ matrix) are applied for data transformation (across time) corresponds to linear filtering using a convolution of a signal with a filter impulse response. In other words, every eigen vector represents one impulse response (inverted in the time) of a filter for filtering time trajectory of one filter bank band. First three eigen vectors of $\mathbf{AC} \times \mathbf{WC}^{-1}$ matrix are shown in figure 2.12 together with their frequency characteristics. The eigen values (not shown in figure) indicate that the projection only to the first eigen vector (the vector with the largest corresponding eigen value - figure 2.12a) preserves most of variability (60%) in data important for a class separability. The frequency response of this filter (figure 2.12b) has a band pass character and its property is similar to the classical RASTA filter. The second and third filters are then similar to the filters used for computation of feature derivatives (delta and acceleration coefficients). Therefore only the first filter can be used and a final feature vector (after decorrelation) can be completed by its derivatives. Another possibility is filtering filter bank output using N first filters (typically 3) and merging the results of this filtering for every frame to a vector with the size equal to N times number of filter bank bands. This vector can be than decorrelated and its size can be reduced using one of proposed methods for deriving of spectral basis.

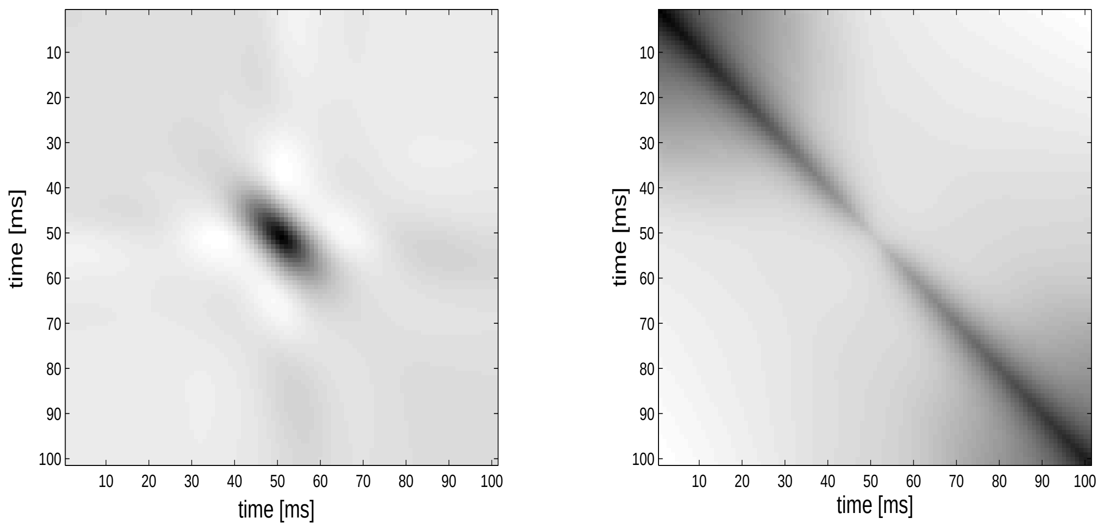

Figure 2.11: Across-class and within-class covariance matrix computed from the time trajectory of one band of a filter bank

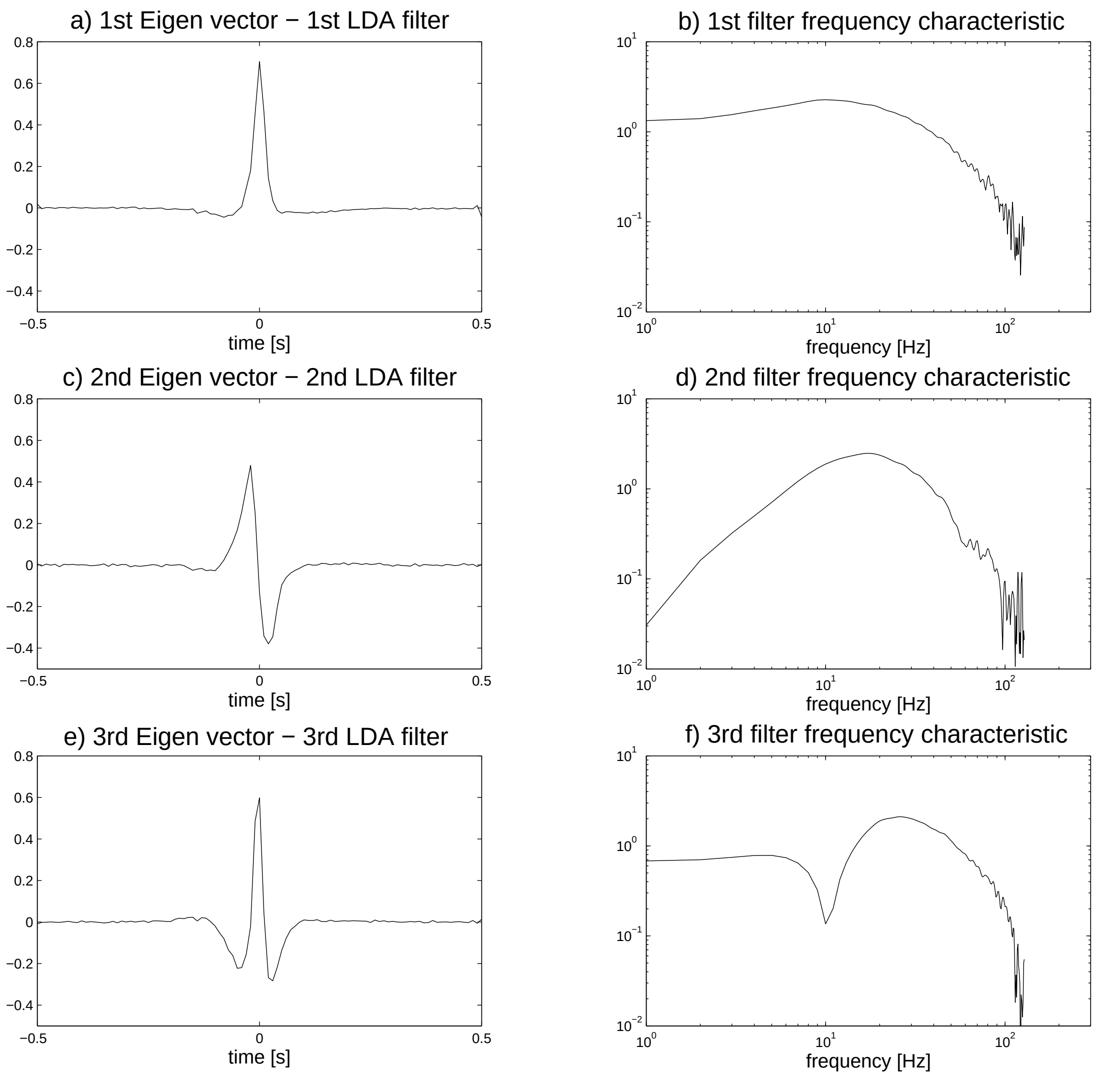

Figure 2.12: Impulse responses and frequency characteristic of temporal filters given by first three LDA eigen vectors

Chapter 3

LDA filters for recognition of Czech

3.1 Databases

3.1.1 The OGI multilanguage telephone speech corpus

The database *OGI Multilanguage Telephone Speech Corpus* serves the needs of automatic language identification and for multi-lingual speech recognition. The corpus consists of maximally nine utterances of each speaker, ranging from simple words until spontaneous recordings of up to on minute. The recordings were made over the fixed network by English, French, German, Japanese, Korean, Chinese, Spanish, Tamil, Persian (Farsi) and Vietnamese speakers.

The DB was created at *Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology.* 2485 recordings were made in total. Out of these, 1042 calls are in English, and an average of 144 calls for the other languages. 1345 recordings (29.5 hours of speech) were selected as suitable for processing and evaluations: 246 in English and 122 in average in the other nine languages. Table 3.1 shows the total number of calls, the number of useful calls and the average duration of utterances in seconds per speaker, for all 10 languages. More information on this corpus can be found in [27].

We have used only a fraction of this database, known as *Continuous English Speech.* It contains recordings of approximately one minute of spontaneous English. In this work, we refer to this DB as to OGI-Stories. 208 phonetically aligned recordings were used. For the description of phoneme-set used and phoneme coverage, see section 7.1.1 and Table 7.1 in chapter 7.

language	total calls	useful calls	average length [s] / speaker
English	299	246	89.70
French	149	123	85.38
German	157	118	87.69
Japanese	147	107	79.47
Korean	148	111	71.12
Chinese	174	133	66.44
Spanish	149	117	86.04
Tamil	188	150	67.84
Persian	153	114	79.56
Vietnamese	158	126	69.80

Table 3.1: Numbers of recordings in different languages in OGI multilanguage telephone speech corpus

phoneme	index	#	%	phoneme	index	#	%
t_S	0	26731	0.69	j	22	51148	1.32
e	1	154367	3.97	p	23	50633	1.30
S	2	28114	0.72	e:	24	28296	0.73
c	3	17057	0.44	i:	25	84002	2.16
i	4	119482	3.07	b	26	27088	0.70
n	5	77328	1.99	Z	27	20226	0.52
a	6	126494	3.25	x	28	21125	0.54
k	7	71301	1.83	u:	29	20411	0.52
o	8	120499	3.10	J_	30	10505	0.27
t_s	9	35087	0.90	f	31	26277	0.68
v	10	52078	1.34	J	32	37803	0.97
m	11	64041	1.65	g	33	9984	0.26
z	12	35770	0.92	o_u	34	22284	0.57
t	13	93557	2.41	a_u	35	17366	0.45
r	14	53215	1.37	d_z	36	3336	0.09
u	15	61569	1.58	e_u	37	7007	0.18
h_	16	17251	0.44	o:	38	11834	0.30
a:	17	66801	1.72	N	39	6977	0.18
d	18	40926	1.05	F	40	3474	0.09
l	19	73137	1.88	oth	41	5497	0.14
P_	20	20817	0.54	sp	42	346508	8.91
s	21	105304	2.71	sil	43	1615345	41.55
total					44	3888052	

Table 3.2: List of Czech phonemes and their numbers of occurrences (#) and percentages (%) in SPEECHDAT database.

3.1.2 Czech SpeechDat-E

Czech SpeechDat-E (Eastern European Speech Databases for Creation of Voice Driven Teleservices) contains recordings from 1052 Czech speakers (526 males, 526 females), recorded over the fixed network. It was created by *Brno University of Technology* and *Czech Technical University in Prague* in cooperation with *Lernout & Hauspie France*. The database is delivered on 6 CD's. The data are stored in raw A-law files (F_s=8000Hz), and each waveform file has an associated transcription in ASCII-SAM format [8]. The corpus contains phonetically balanced words and phrases, application words, dates, times, spelled items, strings of digits and some spontaneous answers to questions.

For the training of LDA filters, only correct (no mis-pronunciation marks) phonetically balanced phrases were used — their number is 5147. The recognition experiment was run on isolated phonetically balanced words. More information on this database can be found on the project's Web pages: `http://www.fee.vutbr.cz/SPEECHDAT-E/` and in [34]. In this work, this database will be called SPEECHDAT.

The phonetic alignments for training of LDA filters were produced by Lukáš Burget with context-dependent HMM recognizer (models were generated by Martin Karafiát [22]). The phoneme set was defined during the SpeechDat-E project and is written in SAMPA; more information about Czech SAMPA can be obtained from `http://www.phon.ucl.ac.uk/home/sampa/czech.htm`. Table 3.2 details the phoneme coverage.

3.2 Recognizer and reference results

Isolated word recognizer based on context-independent phoneme models defined in the diploma thesis of Petr Schwarz [30] was used in this work for evaluation of LDA filtering. The recognizer is trained on isolated, phonetically balanced words (2777 items) and tested also on isolated phonetically balanced words (1394 items). The recognition vocabulary contains 2175 words. Two flavors of training and testing were used:

- using full-length items, including sometimes long portions of silence at the end.
- "cut" version of the database where silence portions were stripped using GMM-based voice activity detector, boot-strapped by a simple energy-based segmentation. The diploma thesis of Petr Schwarz [30] presents this speech/silence segmentation in detail.

The recognizer was built on standard HTK [35] tools. Recognition performance was evaluated using the word recognition accuracy. In the tables, the accuracy on non-cut database will be denoted `acc` while the one evaluated with silences stripped-off will be denoted `acc_cut`.

3.2.1 Mel-frequency cepstral coefficients: MFCC_E/0_D_A

To evaluate baseline performance of our recognizer, the very standard feature extraction: MFCC, appended with log-energy and velocity and acceleration features was tested first. Features were computed by the `HCopy` tool with standard frame settings: length length 25 ms, shift 10 ms. We have computed 12 cepstral coefficients + log-energy resulting in a feature vector of 13 coefficients. After appending Δs and $\Delta\Delta$s, we obtained the famous feature vector size of 39.

The second experiment consisted in replacing the log-energy by the 0-th cepstral coefficient, a trick, that according to experience of ASP group at OGI improves the recognition performance in real noise conditions.

The whole run of the recognizer, including computation of features, training, decoding, and scoring, took approximately 5 hours on a single computer with PIII processor. Table 3.3 summarizes the results of baseline experiments. We can draw the following conclusions from this table:

	`acc` [%]	`acc_cut` [%]	_D_A `acc` [%]	_D_A `acc_cut` [%]
MFCC_E			81.60	89.38
MFCC_0	60.47	68.51	**90.24**	88.38

Table 3.3: Baseline results.

- dynamic features dramatically improve the recognition accuracy.
- for log-energy, the recognition accuracy is highly improved by stripping the silences from the database: _D_A `acc` vs. _D_A `acc_cut` difference in the first line.
- using 0-th cepstral coefficient slightly degrades the performance on cut-database, but greatly boosts the performance on the original data (no silence stripped): 90.24% is the maximum accuracy obtained in baseline results. We think this difference should be attributed to the default pseudo-normalization of log-energy in HTK (maximum value is always set to 1) which makes modeling of silence harder and less reliable.

In the following experiments, **MFCC_0_D_A** features will be therefore used as reference.

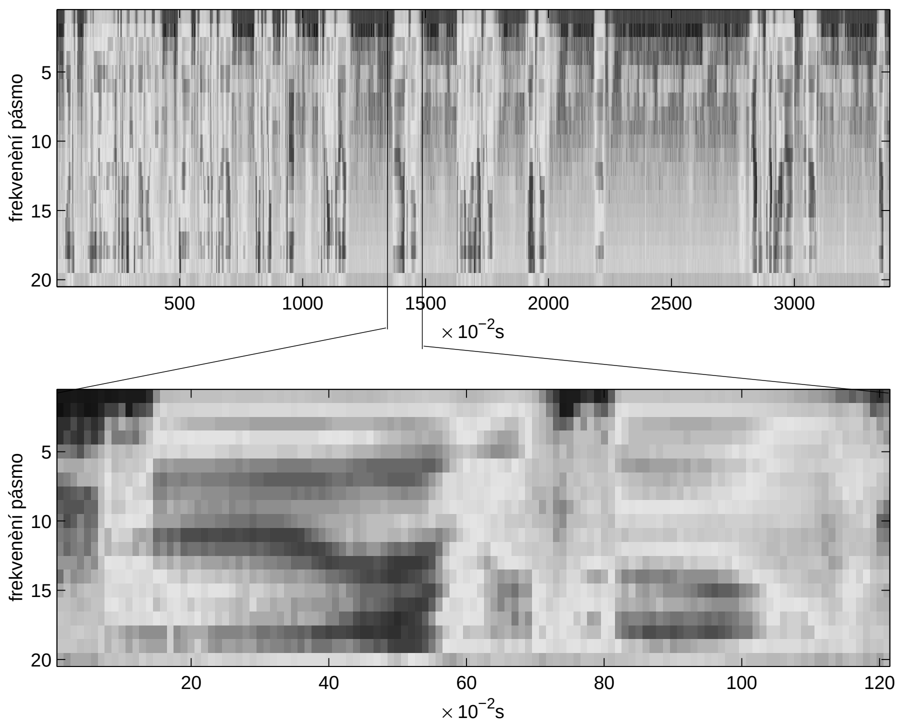

Figure 3.1: Mel filter-bank outputs

3.3 Computing LDA filters

Three questions arise when it comes to practical implementation of LDA filters:

1. which features to choose for LDA filtering.
2. which classes to use while deriving LDA filters.
3. how to handle beginnings and ends of the files in time of the LDA filters computation and implementation.

The following paragraphs will present the possible answers to those questions:

3.3.1 Features for LDA filters

The choice of features for deriving LDA filters was quite obvious: we wanted the LDA features to be as compatible with MFCC's as possible. The choice were therefore log-energies after Mel-scale filterbank, that are used as a mid-product in MFCC computation. They can be output by standard HTK tool `HCopy` using the `FBANK` feature type, they can be then processed by non-HTK software (Matlab in our case), and then converted to MFCC again by `HCopy` with proper specification of the input type. The only problem we have faced was the 0-th cepstral coefficient, that had to be computed 'by hand'. The number of filters in Mel-filterbank was the HTK default: 20. Figures 3.1 and 3.2 show temporal trajectories in all Mel-frequency bands and give a detail of the 10-th band.

3.3.2 Classes for derivation of LDA filters

43 phonetic classes were available for STORIES, however, two of them were excluded:

- the 'other' class `oth` where some rarely occurring phonemes were put and which was not very consistent (see section 7.1.1).

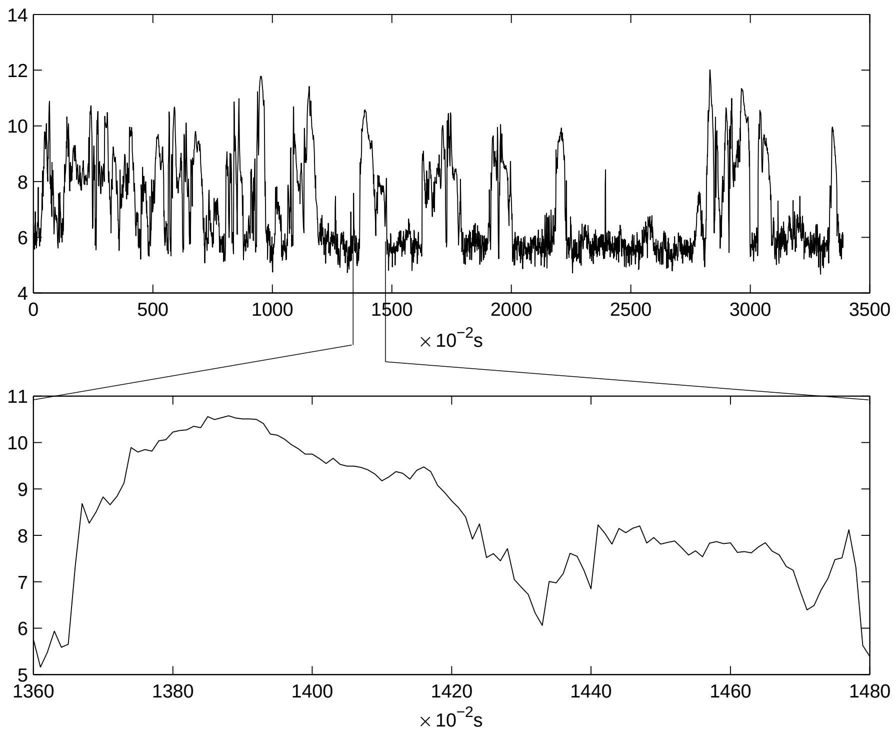

Figure 3.2: Output from 10-th Mel filter

- the silence `pau` with its huge proportion in the database (almost 20% according to Table 7.1. This class with its broad distribution could heavily bias the within-class covariance matrix. Therefore it was discarded too and the derivation of filters was done using 41 classes.

The reader should however note, that "discarding" of classes does mean not using the vectors with *center* in `pau` or `oth`, but still keeping those regions as context while the center lies elsewhere.

3.3.3 Handling edges

As everything, each speech file begins and ends. For derivation and subsequent use of LDA filters, in case we consider 1 second trajectories, we need 50 acoustical frames of left context and 50 frames of right one. There is no exact answer to this problem and the following engineering solutions exist (for illustration, we will consider a file containing 3000 feature vectors numbered 1 ... 3000):

- discard all the vectors on boundaries, begin at 51 and end at 2949. This simple approach can be used for LDA filter derivation where we are permitted to "loose" some vectors (does not harm for long STORIES utterances), it's however hardly usable in recognition, where for example a digit can well lie in the beginning half a second.
- completing the file by zeros:
  0 0 0 . . . 0 1 . . . 3000 0 0 0 . . . 0
  The disadvantage are sharp transitions from zeros to valid vectors.
- flipping of 50 vectors at the beginning and 50 vectors at the end:
  50 49 . . . 1 1 2 3 . . . 3000 3000 2999 . . . 2951
  Now the transitions are smoother, but we risk to "fool" the derivation by reverting the order of time.
- repeat the end of file at the beginning and the beginning at the end:
  2951 2952 . . . 3000 1 2 3 . . . 3000 1 2 . . . 50

Here, if we consider the first half a second to mostly contain speech and the last one to contain mostly silence, we have a danger of accidentally adding some speech at the end of file.

- to cope with this problem, we may repeat the end of file at the end *and* at the beginning:
  2951 2952 . . . 3000 1 2 3 . . . 3000 2951 2952 . . . 3000
- . . . or to flip it at the end:
  2951 2952 . . . 3000 1 2 3 . . . 3000 3000 2999 . . . 2951
- we can also concatenate the the consecutive files in the database. This approach that is the most natural (usable however only in the case we dispose of all the files at the same moment) and was used in most of the recognition experiments.

3.4 Using LDA filters

After computation of LDA filter, the filtering is done by the following equation:

$$y(n) = \sum_{m=1}^{101} x(n+m-50)h(m) \tag{3.1}$$

where $x(n)$ is the filtered trajectory and $h(m)$ is the LDA filter. As this equation does not contain $n-m$ term that we would expect in a convolution, a more precise term would be a *projection of time-trajectory onto LDA eigen vector*. It would be however easy to invert $h(n)$ in time and obtain proper filtering equation.

The basic processing is depicted in Figure 3.3. We are however motivated (section 3.2.1) to use velocity and acceleration parameters. There are two approaches to compute them while using LDA filtering:

- apply LDA filtering for band energies only with the static coefficients, then de-correlate using DCT and compute Δ and $\Delta\Delta$ at the end of processing in standard way, using approximations of first and second derivative.
- use LDA filters related not only to the first, but also second and third biggest eigen-values. As we have seen in section 2.5, they present some similarities to the first and second derivative of the first filter. The outputs of all filters have to be de-correlated by DCT or PCA.

Another issue is the computation of energy-related 0-th cepstral coefficient, which is actually a sum of log-energies in bands. This can be computed from LDA-filtered energy trajectories or from original ones.

3.5 Derivation of LDA filters on STORIES

LDA filters were computed on STORIES using the mathematical apparatus described in section 2.2.2. As mentioned in paragraph 3.3.3, the leading 50 and trailing 50 vectors of each were used just as context. Examples of across-class and within class covariance matrices can be seen in Fig. 3.4. An example of derived filters for 10-th band can be seen in Fig. 3.5.

3.6 Straight application of LDA-filters

In these experiments, LDA filters were applied to filter-bank outputs without any modifications. Four different setups were tested depending on the dealing with edges and processing of c_0:

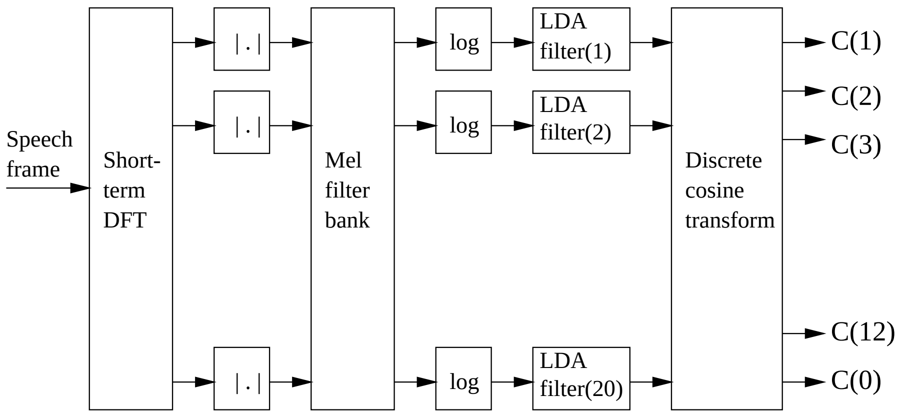

Figure 3.3: Implementation of LDA in MFCC computation.

- In experiments `LDA_zeros`, the SPEECHDAT features were completed by 50 zero vectors on the left and by 50 zero vectors on the right. In `LDA_cat`, the 50 frames of left context were extracted from the previous file while the 50 of right context were extracted from the following one.
- In experiments `xxx_c0_orig`, the value of 0-th cepstral coefficient was computed from original non-filtered trajectories, while for `xxx_c0_fil`, c_0 was extracted from the energies already processed by LDA filtering. Note that technically, the `HCopy` tool can not compute c_0 from filter-bank outputs (`FBANK`), even if mathematically, this is perfectly possible. Instead, the 0-th cepstral feature must be included in the parameter stream (`FBANK_0`). In case we computed c_0 from the filtered trajectories, it was done "by hand" in Matlab.

The results including baseline are summarized in Table 3.4.

	acc [%]	acc_cut [%]	_D_A acc [%]	_D_A acc_cut [%]
MFCC_0	60.47	68.51	90.24	88.38
`LDA_zeros_c0_orig`	58.75	70.23	87.37	85.58
`LDA_zeros_c0_fil`	58.75	69.08	86.79	83.50
`LDA_cat_c0_orig`	72.24	74.53	**90.39**	88.52
`LDA_cat_c0_fil`	73.53	77.76	89.53	88.67

Table 3.4: First results with LDA filtering.

The analysis of those results shows that:

- adding zeros at beginnings and ends of files hurts the recognition performance (except for just the basic parameters on DB with stripped silences). This was expected, as we are introducing quite a serious discontinuity in the data prior to the filtering.
- LDA filtering with adding pieces of previous and following files helps the recognition while using basic parameters only. For the features completed with Δ and $\Delta\Delta$, recognition performances are approximately the same as for the baseline.
- Using original energy trajectories seems to be advantageous for the c_0 coefficient. We may speculate that this coefficient, so needed to distinguish between speech and silence, is "smeared" in case of LDA filtering, thus providing weaker clue for the recognizer.

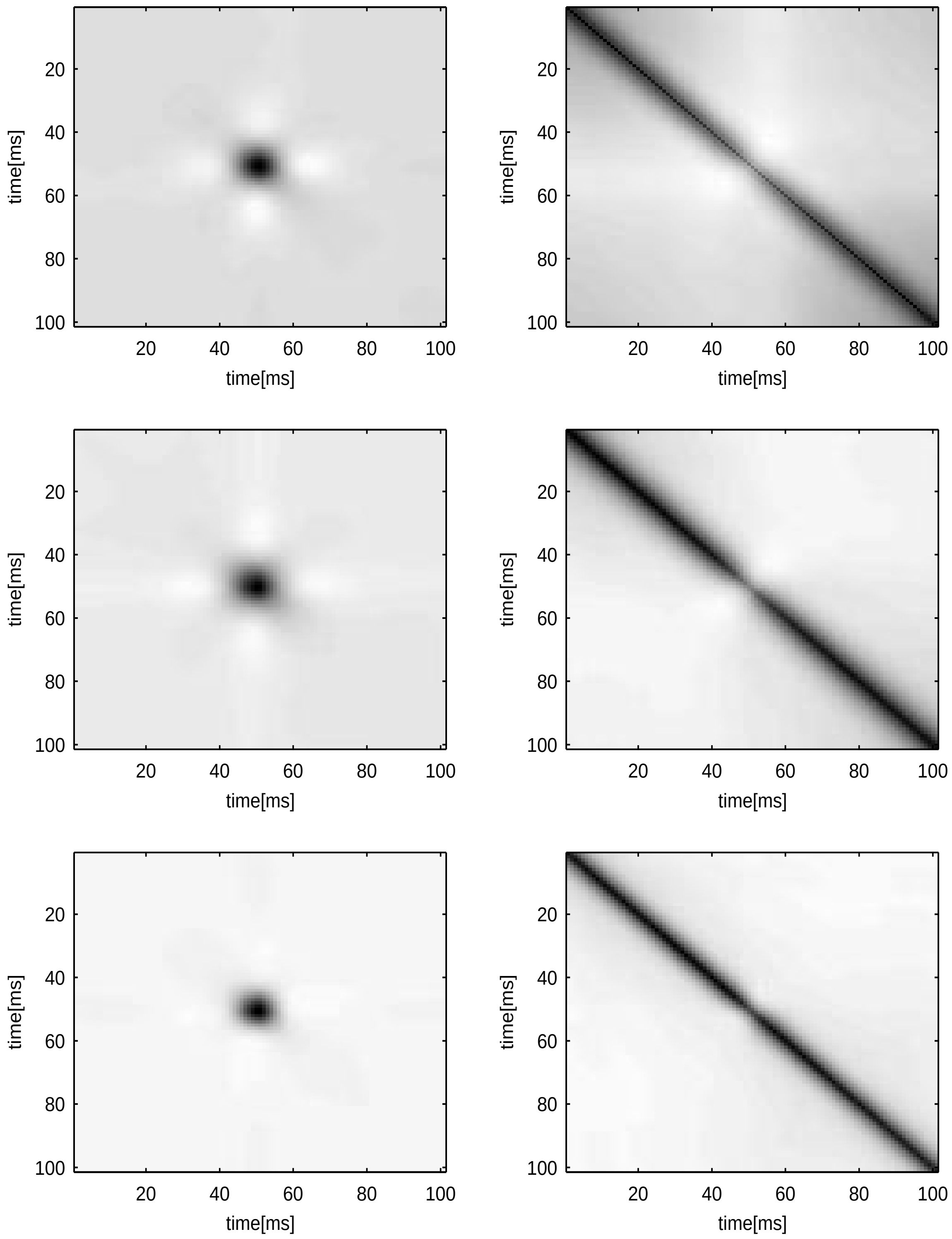

Figure 3.4: Across-class (left column) and within class (right column) covariance matrices for 1-st, 10-th and 20-th frequency bands.

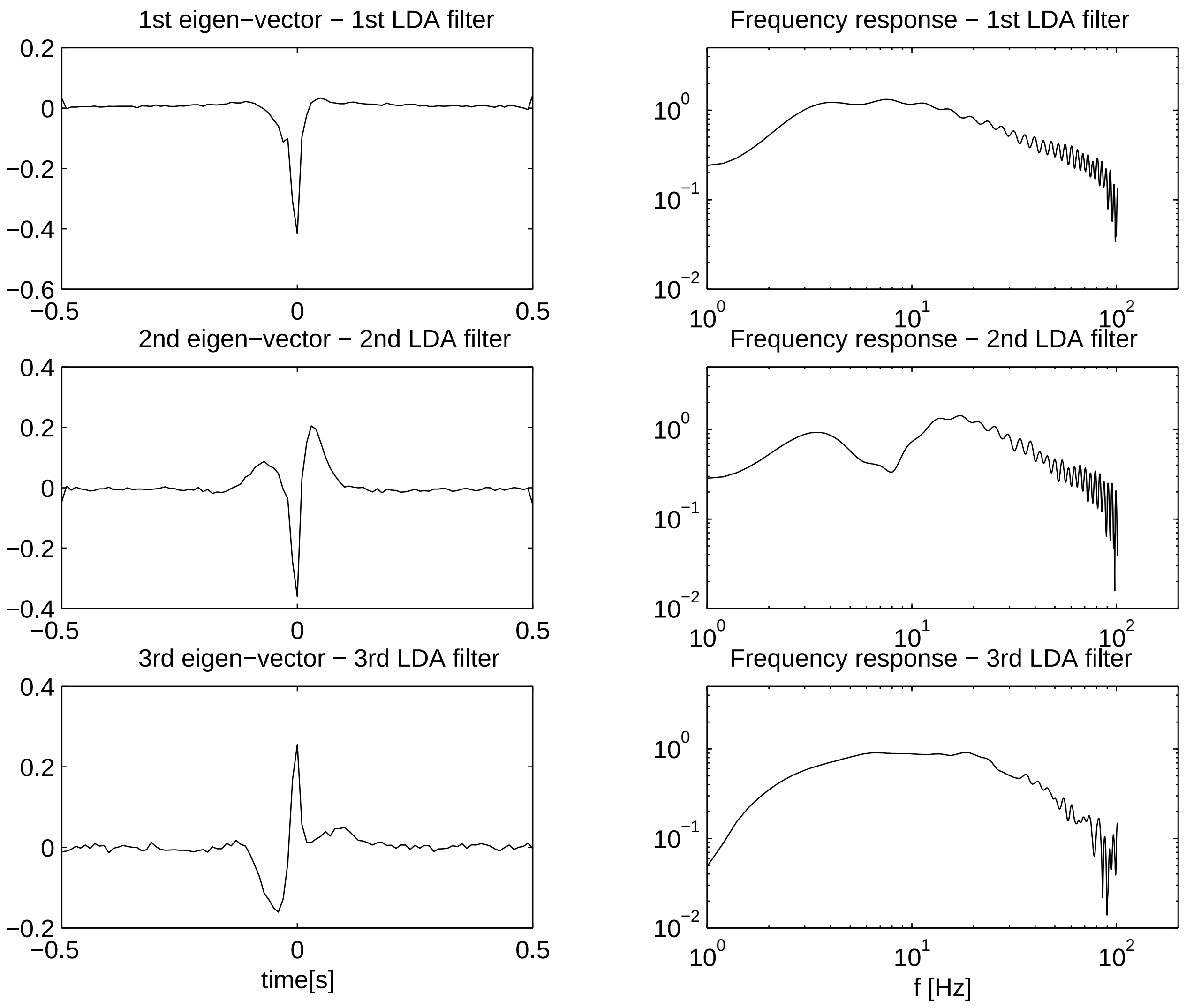

Figure 3.5: First three LDA filters and their frequency responses for the 10-th band.

3.7 Processing edges of filters

Having a closer look at the LDA filters (Fig. 3.5), we find, that while the samples of each filter converge toward zero both for the beginning and end of each filter, the very first (0-th) and very last (100-th) samples have greater value. Since this is obviously an error of estimation, we have proposed three ways to cope with those samples:

1. Weighting filters by a Hann window [18], Fig. 3.6, which not only suppresses the two unwanted samples, but influences the whole length of the filter. This experiment will be denoted `hann`.

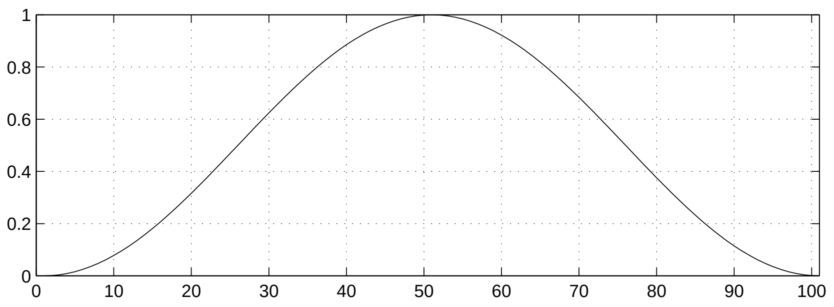

Figure 3.6: Hann window, 101 samples.

2. Influencing first 5 and last 5 samples by a sharply raising and falling function of the form:

$$w(n) = \begin{cases} 0.9(\frac{n}{2})^5 & \text{for} \quad 0 \leq n \leq 4 \\ 0.9(\frac{100-n}{2})^5 & \text{for} \quad N-4 \leq n \leq N \\ 1 & \text{for} \quad 5 \leq n \leq N-5 \end{cases} \tag{3.2}$$

The resulting window is shown in Fig 3.7. Experiments with this processing will be denoted `x5`.

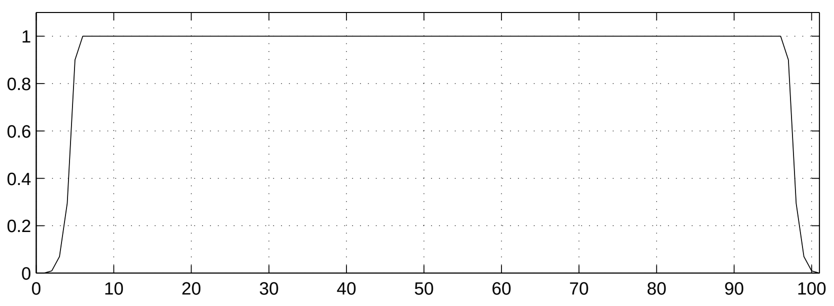

Figure 3.7: Window influencing first 5 and last 5 samples of filter.

3. zeroing of first and last sample, the analysis window is very simple in this case:

$$w(n) = \begin{cases} 1 & \text{for} \quad 1 \leq n \leq N-1 \\ 0 & \text{elsewhere} \end{cases} \tag{3.3}$$

In the table, this experiment will appear with the `firstlast0` label.

The results, including the baseline and previous two, are summarized in Table 3.5. Note that all these experiments use the context creation from previous and following file (`cat`). For comparison, also the baseline and previous LDA results are presented. We see that all modifications do approximately

	acc [%]	acc_cut [%]	_D_A acc [%]	_D_A acc_cut [%]
MFCC_0	60.47	68.51	90.24	88.38
LDA_zeros_c0_orig	58.75	70.23	87.37	85.58
LDA_zeros_c0_fil	58.75	69.08	86.79	83.50
LDA_cat_c0_orig	72.24	74.53	90.39	88.52
LDA_cat_c0_fil	73.53	77.76	89.53	88.67
LDA_cat_hann_c0_orig	69.94	74.46	89.60	88.45
LDA_cat_hann_c0_fil	73.17	77.33	90.75	88.38
LDA_cat_x5_c0_orig	73.53	74.75	90.32	88.64
LDA_cat_x5_c0_fil	74.39	77.04	90.32	88.24
LDA_cat_firstlast0_c0_orig	73.31	74.53	**91.32**	88.31
LDA_cat_firstlast0_c0_fil	74.32	77.47	89.96	88.38

Table 3.5: LDA results with processed filters.

as well as original filters. The "winner" is the simplest processing - just zeroing the first and last sample in each LDA filter. We have also confirmed, that the results on original database (silences included) is consistently better than on the "cut" version and that the c_0 should be computed from the original energy trajectories. The best result so far is recognition accuracy of 91.32% in the `LDA_cat_firstlast0_c0_orig` experiment.

3.8 LDA filters trained on SpeechDat

So far, we have tested the recognition of Czech isolated words with features computed using LDA filters derived on US-English. As a next step, we wanted to train those filters directly on SPEECHDAT to see the influence of:

- training of LDA filters on the target language.
- training of LDA filters on the target database.

Unfortunately, other telephone Czech database was not at our disposal, so that we could not investigate into those two points independently. The following results will therefore present a situation, where the derivation of LDA's is very closed to the target task (language *and* channel characteristics are the same).

The description of material in SPEECHDAT DB is given in paragraph 3.1.2. For the training of LDA filters, the 101-point vectors with centers situated in silence `sil` (41.55% in the database) were not used for similar reasons as for STORIES (paragraph 3.3.2). Inter-word short pauses `sp` and `oth` class were not used as well, so that the so that the number of phonetic classes was 41, accidentally the same as in previous experiments.

The LDA filters were computed in the same way as in previous case. Figure 3.8 shows examples of across-class and within-class covariance matrices, that are, not surprisingly, quite similar to what we have seen for STORIES (Fig. 3.4). Examples of derived first LDA filters are shown in Fig. 3.9. We see again shapes quite similar to the first filter shown for STORIES in Fig 3.5.

3.9 Application of filters derived on SPEECHDAT

The first application of SPEECHDAT-derived LDA filters was on the data with appended context from the previous and following files (`cat`). The results are given in the following Table 3.6. Experiments with SPEECHDAT-derived filters have prefix `LDASD`.

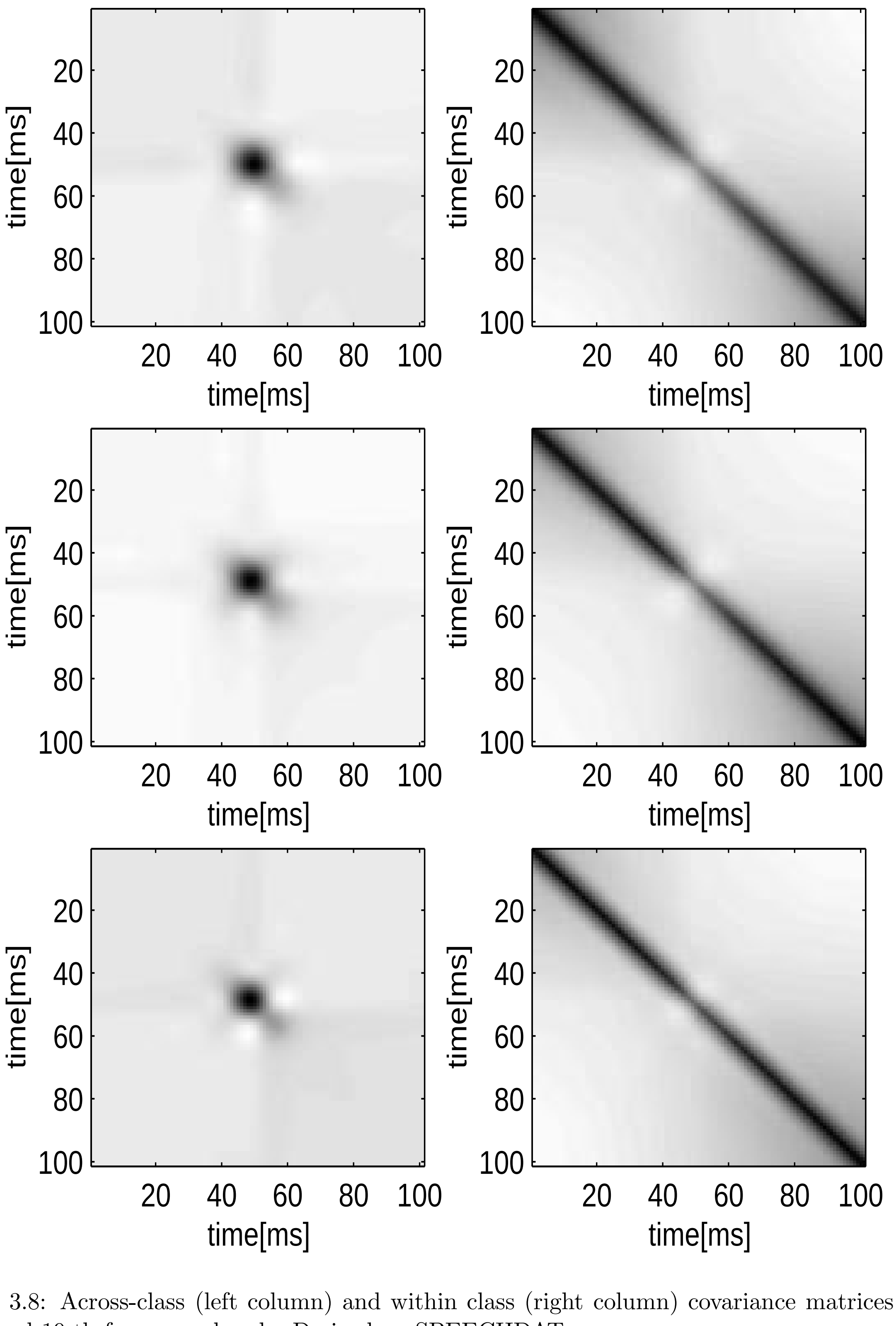

Figure 3.8: Across-class (left column) and within class (right column) covariance matrices for 2-nd, 10-th and 19-th frequency bands. Derived on SPEECHDAT.

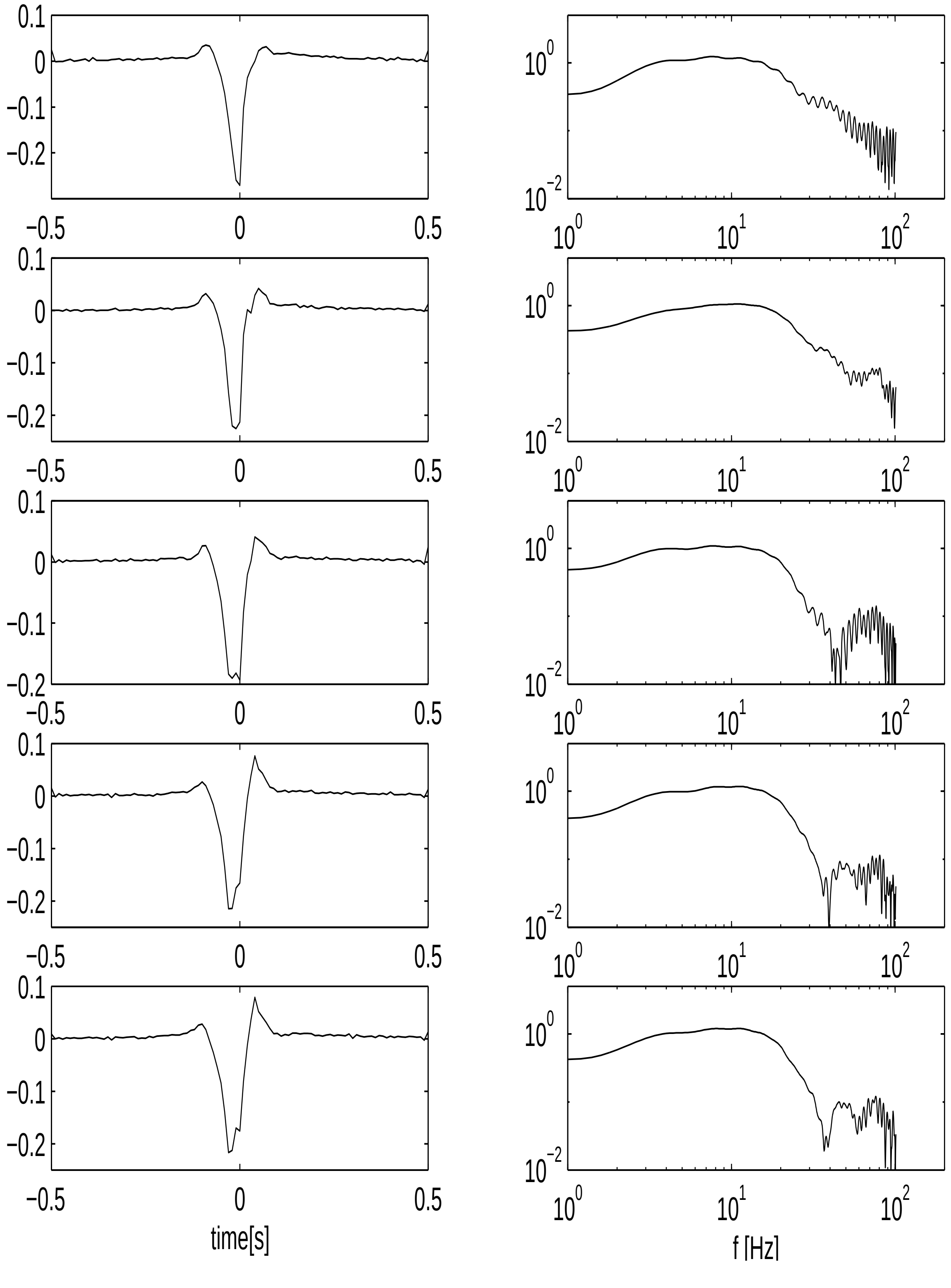

Figure 3.9: LDA filters trained on SPEECHDAT and their frequency responses for 2-nd, 6-th, 10-th, 15-th and 19-th bands.

	acc [%]	acc_cut [%]	_D_A acc [%]	_D_A acc_cut [%]
MFCC_0	60.47	68.51	90.24	88.38
LDA_cat_c0_orig	72.24	74.53	90.39	88.52
LDA_cat_c0_fil	73.53	77.76	89.53	88.67
LDASD_cat_c0_orig	71.88	74.96	91.61	89.31
LDASD_cat_c0_fil	72.31	77.12	**92.11**	88.88

Table 3.6: First results with LDA filters derived on SPEECHDAT.

We see that even non-perfect filters (quite high values of first and last coefficients) perform well if derived on the target data - we see an absolute improvement of almost 2% over the baseline, and SPEECHDAT derived filters outperform their STORIES counterparts. An interesting result is a better result with c_0 derived on *filtered* trajectories (note that for filters derived on STORIES, the "original" c_0 performed always better. . .).

3.10 Processing of filters derived on SPEECHDAT

The last experiments were aimed at processing of SPEECHDAT-derived filters in order to suppress the influence of first and last badly estimated samples. Only the zeroing of those samples (last point in section 3.7) was performed, so that the corresponding results carry the `firstlast0` label. The Table 3.7 shows those final results, again compared to the baseline and the closest STORIES setup:

	acc [%]	acc_cut [%]	_D_A acc [%]	_D_A acc_cut [%]
MFCC_0	60.47	68.51	90.24	88.38
LDA_cat_firstlast0_c0_orig	73.31	74.53	**91.32**	88.31
LDA_cat_firstlast0_c0_fil	74.32	77.47	89.96	88.38
LDASD_cat_firstlast0_c0_orig			91.39	88.88
LDASD_cat_firstlast0_c0_fil			**92.40**	89.10

Table 3.7: Final results with LDA filters derived on SPEECHDAT.

The best result obtained with LDA filtering was obtained in experiment `LDASD_cat_firstlast0_c0_fil`. It outperforms the baseline by more than 2% absolute.

Chapter 4

LDA filtering – conclusions

4.1 Conducted experiments

In the experiments conducted, Mel-filterbank output trajectories were processed by LDA-derived filter filtering. The filters were first trained on US-English OGI-Stories database, then on the recognizer's target data: Czech SpeechDat-E database.

The results have clearly shown the advantage of LDA filtering even for filters derived on a different database (outperforming MFCC_0 baseline by more than 1% absolutely). The filters derived on the target data have shown superior performance, and gave more than 2% absolute improvement over the baseline.

We may conclude, that LDA filtering of temporal trajectories is a cheap and efficient way to improve the performance of simple speech recognizers. This is confirming the results the ASP group at OGI has obtained in numerous experiments during the dissertation of Sachin Kajarekar [21] and during the AURORA project [1].

4.2 Open problems and questions

Even if the results were encouraging, we should not forget that there are many unresolved problems in this work necessitating further research. From the more technical to more general they include:

- in this work, we have not investigated the possibility to use *LDA-filters corresponding to the 2-nd and 3-rd eigenvalues* for computation of Δ and $\Delta\Delta$ approximations. When trying to use those, we should cope with the correlation of those features:
  - in MFCC, the discrete cosine transform is used to convert the estimate of log-spectrum back to temporal domain, but its main purpose is to decorrelate the features. Will the use of DCT be justified also for features filtered by the 2-nd and 3-rd filters ?
  - there is of course a possibility to train a PCA (section 2.2.1) to decorrelate the features. Should this transform be trained per-stream (ie. separately for 1-st, 2-nd and 3-rd filter) or as a whole ?
- the current work does not make any use of feature normalization while we know, that off-line or on-line normalization dramatically improves the recognition accuracies for noise conditions [16]). On the other hand, this normalization can behave in quite unpredictable way for clean conditions, and for scenarios with some non-standard features (see for example the work of Petr Motlíček on all-pole modeling of log-spectra [26].
- there are many *engineering choices* to be made while computing/implementing the LDA filters:

  - selection of classes was done by a brief investigation of how do the phoneme proportions look like. We have not at all investigated using of broad phonetic classes, that are known to perform well for TRAP-based systems [17]. Also, the ASP group at OGI reports to obtain slightly smoother filters and better results while dividing each phoneme into 3 sub-classes. The original work of Jiří Kafka [20] however reports quite discouraging results of such experiments. Kafka has argued that there is too little data to estimate the covariance matrices reliably for this method.
  - post-processing of filter coefficients was found to be quite important for good performance. However, only a few "ad-hoc" experiments were performed, based on "looking at the filter".

- The final point questions the very base of this work: the linear discriminant analysis LDA. The basic problem of this method is that it expects the same statistical distribution of data in all classes. This is obviously never satisfied, so that the derived filters are globally optimal, but they can perform very badly for the discrimination of some classes.

  The problem is not only to find mathematical methods that relax the assumption of equal distribution of data within classes, but also to implement them and test their performances on speech recognition tasks.

4.3 More on LDA...

The previous chapter has described only one of possible applications of LDA in speech processing – filtering of trajectories in order to obtain MFCCs more robust in noise. LDA is however a more general technique allowing for dimensionality reduction while preserving the discriminability and for the analysis of variation in speech.

4.3.1 2-dimensional LDA

Sachin Kajarekar studied in his thesis [21] among others the the long-term phone variability in temporal domain when used in feature extraction. The goal was to improve the robustness of the features by incorporating information about longer time-span. Linear discriminant analysis was used to extract the useful features from 101x15 dimensional block of spectrogram. Two types of analysis were used. First analysis, referred as joint analysis, used all the correlations in this block. Second analysis, referred as combined analysis, assumed that time and frequency domains were independent. The discriminant features from both analysis were used in speech recognition experiments.

The results showed that incorporating longer time-span in the design of feature extraction improved performance of the system. Specifically, features from joint analysis performed worse than combined analysis because joint analysis over-fitted the training data and had poor generalization. This showed that results of analysis of variability can also be used to improve the performance of speech recognition system.

4.3.2 LDA in TRAPs

František Grézl uses LDA to project 101-point temporal trajectories on several LDA basis before training the TRAP-based system [10]. More on TRAPs can be found in the following chapters 5-9. Grézl has shown that processing of TRAPs using LDA improves the performance (due probably to the fact, that the input layer to the MLP is smaller, though the data preserve the discriminability information).

Anoher Grézl's work was testing of a "merger-only" TRAP system. More details on those experiments including block diagrams in paragraph 9.2.2.

Chapter 5

TempoRAl Patterns – TRAPs

"Classical" features for speech processing (as MFCC's) provide information about the entire spectrum of speech signal for a very limited time (the spectrum is usually computed in frames of 20-25 ms with a window-shift of 10 ms) [35]. If noise is present in the speech signal, it affects the entire feature vector, and impairs the accuracy of the recognizer. Multi-stream approach [32, 3] overcomes this problem by running several speech recognizers independently in different frequency bands, and recombining their results. The recognition in bands and recombination of results is mostly done using an HMM-ANN hybrid speech recognizer.

In recent years, people around Hynek Hermansky have shown [14, 31, 12, 6] that non-linear mapping using ANN can be used in conventional HMM-GMM recognizers. The net simply produces a stream of features, which is, after post-processing, used as input to HMMs. This opens the possibility to use such non-traditional features with "standard architecture" speech recognizer, as HTK (in the Aurora task) or Sphinx (in the SPINE project).

5.1 TRAP architecture

Briefly described, a TRAP system (Fig. 5.1) classifies long (for example 1 second) temporal trajectories of spectral features into classes neural networks (NN). Then, band-outputs are merged by another NN to form the final probability vector. If used with an HMM recognition system, those probabilities are post-processed to better fit HMM requirements (feature independence and Gaussian distribution).

In more detail, the TRAP system consists of the following:

- input features - log of energies in critical bands. We used 15 Bark-scale critical bands from 0 to 4000 Hz, the log energies were computed by the 'rasta' executable from ICSI. The issues concerning the edge-effects are discussed in section 5.1.1. We need to have phonetic labels for the input data.
- generation of TRAPS. Section 5.1.2 discusses the normalization, Hamming widowing, and multi-band traps.
- TRAP- or band-classifiers perform classification of TRAPs into phonetic classes or broad phonetic categories. See 5.1.3 for training of those nets.
- post-processing of band-classifier outputs. This involves conversion of linear probabilities to logs and multiplication by priors (which, as we will see, is completely useless) - see section 5.1.4.
- merging net putting all the band-classifier outputs together. Section 5.1.5 gives more detail.
- post-processing of the merger output (again phoneme probabilities) to form features suitable for an HMM recognizer. This very important step is presented in section 5.1.6.

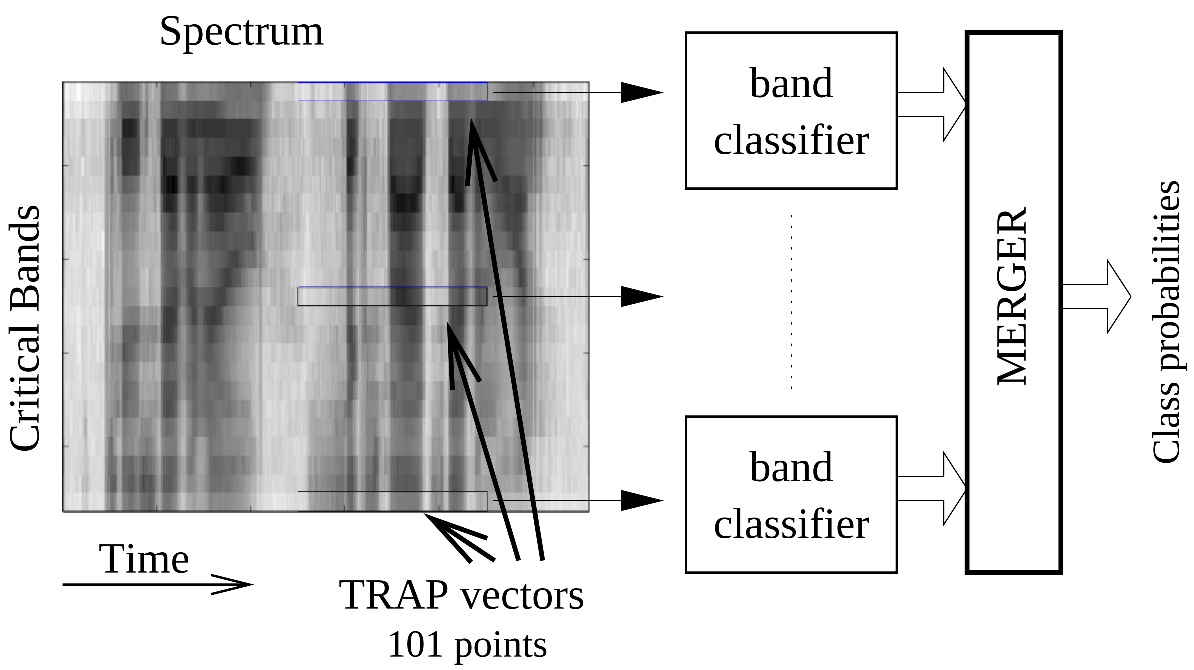

Figure 5.1: Overview of the TRAP system.

- HMM recognizer. We worked with two HMM recognizers (Digits built using HTK and Sphinx on SPINE). Those are described in respective chapters: 6 and 8. HMM recognizer was not tested with the "reference set", though some nets trained on the TIMIT portion of this set were used later in SPINE.

5.1.1 The input: Log energies

are in all experiments computed on Bark-scale using the `rasta` executable from ICSI with the following command-line parameters: `-M -T -A -w 25 -s 10 -L -R`.

- the signal is divided into frames of 25 ms with widow shift of 10 ms. For used 8000 Hz sampling frequency, this means 200-sample frames with shift of 80 samples.
- power FFT spectrum is taken.
- filter energies are computed using a 15-band Bark-scaled filterbank.
- log is taken.

We should note, that the TRAPs used need fairly long context (50 past and 50 future frames for 101-point TRAPs), so that a special care must be given to the beginning and end of each file. In the baseline experiments, the first 50 and last 50 frames of each utterance were flipped to create the context for first and last frame. In recent experiments on SPINE ("improved data" - `idata`), the left context was taken from previous file(s) in the data-set, while the context for the last frame was taken from the following file(s)[1]. The source file for TRAPs were created with those extra left- and extra right-frames.

We used pre-generated phonetic labels:

- converted from Stories and Numbers label-files by Sharma for the Stories-Numbers-Digits (SND) experiments.

[1]a context-list was created, and if a left-context file was shorter than 50 frames, the algorithm went deeper to the list to gather sufficient number of frames.

- converted and re-mapped from TIMIT and Stories for the reference setup by Lukáš Burget.
- generated by Sunil Sivadas for SPINE.

Phoneme sets used in different experiments differ, and have been further re-mapped, see descriptions in respective experimental chapters.

Log energy features together with labels are stored in p-files (format defined for ICSI NN tools).

5.1.2 Generation of TRAPS

A TRAP is nothing but a piece of temporal trajectory of a given band energy of certain length. Most of experiments were conducted with 101-point (1 second) TRAPs. The label of the TRAP is the original label of its central frame. In addition to a mechanical re-arranging of 101-point trajectories into an output matrix, the following options were tested:

- **mean and variance normalization:** mostly done independently for each TRAP. Sentence-based mean and variance normalization were tested too (and proved to give similar results as the TRAP-based on SND). Advantage of the sentence-based normalization is that we can tell the NN training software (Quicknet) to select TRAPs on-line (just by specifying left and right context) rather than to create huge input p-files. For multi-band TRAPs, the normalization is always independent band-by-band.
- **Hamming windowing** of TRAPs was done rather for historical reasons (when Sharma did experiments with distance-based classification of TRAPs, the Hamming windowing helped to pre-accentuate the center of TRAP in the distance computation). As the NN training software does a global mean and variance normalization of each feature prior to NN training, the effect of Hamming windowing is canceled.
- **Discarding some labels**. It was found advantageous to discard TRAPs carrying some data from the training. In the SND experiments, all the phonemes that did not appear in Numbers had to be discarded. In reference experiments, frames carrying the 'other' label were discarded.
- **Balancing the data**. The amounts of frames per class in the training set are mostly heavily unbalanced (most of silence frames, followed by long vowels, little data for phonemes like 'th', etc.). The data can be balanced prior to NN training by specifying down-sampling factors per class.

5.1.3 Band classifiers

Band classifiers (also called TRAP classifiers, small nets, first step, band-posterior estimators) classify the TRAPs into phonetic classes, or, in some experiments, into broad phonetic categories. Each band classifier is a standard multi-layer perceptron (MLP) with 3 layers:

- the input layer's size is determined by the length of TRAP (mostly 101 points).
- the hidden layer, with sigmoid non-linearities, having 300 neurons in most experiments.
- the output layer whose size is given by the number of classes. The softmax [3] non-linearity was used in final layer in band-classifiers.

The training data is split into a training and cross-validation (CV) sets. The learning rate of the net is determined upon the accuracy on the CV set after each epoch by the "new-Bob" (see the documentation to QuickNet training executable `qnstrn`) algorithm (the constants below apply to our training scripts):

1. the training starts with a learning rate of 0.008.
2. if the improvement of error on CV set is >0.5%, the training continues with the initial rate.

3. if it is less, the learning rate will be halved in each of consecutive epochs. This is called “ramping”.
4. if the improvement of the CV accuracy is <0.5%, the training would stop. In case the improvement was negative (deterioration), the weights are restored from a backup of the previous (more successful) epoch.

5.1.4 Post-processing of band-classifiers output

Before being introduced to the merger, the following processing is done on class posteriors:

- log is taken to gaussianize the posteriors. An experiment was conducted also with letting the softmax output intact, but it gave slightly worse performance.
- multiplication by priors (some experiments) done physically as the addition of priors in the log-domain. This is again a historical step, which does not have sense while training the merger: before training, the data are globally mean and variance normalized so that any prior effect is canceled.

5.1.5 Merger

The merger is generally trained on different data from those used for training band-classifiers. It implies that for this training data, TRAPs must be generated and forward-passed through the band classifiers. Resulting posteriors (after post-processing described above) are then used to train the merger.

The classifier is an MLP with 3 layers:

- the input layer’s size is determined by the product of number of bands times the number of classes per band. For example, for 15 bands and 42 phonemes, the input layer size is 630.
- the hidden layer, with sigmoid non-linearities. We used mostly 300 neurons in the hidden layer, which seems quite few compared to the input layer size. Unfortunately, more neurons in the hidden layer result in very long training files.
- the output layer whose size is given by the number of classes. Softmax was used in final layer for training. In the forward-pass, the softmax was kept and followed by an additional off-net non-linearity (log or atanh), or it was removed.

The training of the merger was also driven by the “new-Bob” algorithm for determination of the learning rate.

5.1.6 Merger output post-processing

We want to convert the output of the merger to feature files suitable for HMM recognizer. Two steps are necessary:

1. **Gaussianization**: the outputs of softmax are not Gaussian at all, they have bi-modal distribution with sharp peaks closed to 0 and 1 for most represented classes (as silence) and peaky uni-modal distribution (peak closed to 0) for the other classes. The Gaussianization can be done in one of the following ways:
    (a) taking log of the softmax output. This is going to expand the probabilities closed to 0 to an approximately Gaussian shape, but the problem of probabilities closed to 1 persists: they are going to create a sharp edge in the resulting distribution, or even a peak.
    (b) hyperbolic arcus-tangens of softmax output: $\operatorname{atanh}(2x+1)$, where x is the softmax output, which produces more Gaussian-like distribution.

   (c) removing the softmax from the output layer of the net, which is the simplest solution (no Gaussianization necessary) and gave the best results.
   (d) an explicit Gaussianization [25] is an option, however, it was not used in the TRAP framework.

2. **De-correlation**. HMM's with diagonal covariance matrices "like the features de-correlated". Therefore a PCA is computed on the training data, and then applied to the entire data. Experiments were done on the PCA using raw or normalized covariance matrix.

In addition to those two steps, we can test some processing known from "standard" features, as delta and acceleration coefficient computation, mean and variance normalization, etc. Experiments described at the end of SPINE chapter 8 covered those post-processing tricks.

5.2 How do they look like ?

It is difficult to visualize weights and biases of a trained net. Mean TRAPs were generated to see, if they are consistent with Sharma's results and also if they are consistent among experiments. In addition, the analysis software 'trapalyzer' can produce variance (or better standard-deviation) TRAPs, that tell us, how much variability we can expect at which place of the time trajectory. Some pictures are included at beginnings of SND chapter 6 and reference experiment chapter 7.

5.3 Performance analysis

5.3.1 Phoneme recognition accuracies

are the quickest way to learn, if nets are classifying TRAPs well or bad. Cross-validation set accuracy is the figure to look at both in band-classifier and merger training. Also, phoneme recognition accuracy per class is helpful. Quicknet software can not output this per-class accuracy, but it can be obtain using the in-house 'ffrapalyzer' software.

5.3.2 Phoneme confusion matrices

are the way to see how precisely the net is able to classify, and where does it make most of the errors. Consider number of classes L, and a data set with N frames. We have correct labels for this set, so that we know, that class i has N_i frames. The priors of classes are therefore given:

$$P_l = \frac{N_l}{N} \tag{5.1}$$

For each frame, we have a vector of net outputs giving class posteriors: $\mathbf{x} = [x_1, x_2 \dots x_L]$.

Hard confusion matrix

for each posterior vector, the highest posterior determines the classification of the frame. We can compute how many times a phoneme of correct class i was classified as class j (in ideal case, i would be always equal to j): counts C_{ij}. The hard confusion matrix is then given by a simple division by prior counts:

$$H_{ij} = \frac{C_{ij}}{N_i} \tag{5.2}$$

Ideally, this matrix would be unity (everything correctly classified).

Soft confusion matrix

Rather than taking a decision, this matrix takes into account all the posteriors, and sums them up for each class. Each row of the soft confusion matrix is then defined:

$$\mathbf{S}_{i,:} = \frac{\sum_{\forall i} \mathbf{x}}{N_i} \tag{5.3}$$

where $\sum_{\forall i} \mathbf{x}$ means the sum of all posterior vectors for frames with correct label i. This matrix is therefore going to give more 'blurred' picture of how the posteriors look like for different classes. Ideally, this matrix would be again unity (net each time sure, that it is the correct class and no else).

Variance matrix of posteriors per class

The motivation to compute this matrix was: "if a variance of the posterior for a given correct class is low, it does not matter, if this posterior is high where it should not be – the net works consistently and the merger will take care of it". It is defined by:

$$\mathbf{V}_{i,:} = E\{(\mathbf{x}_{\forall i} - \mu_i)^2\} \tag{5.4}$$

where E denotes the expectation, $\mathbf{x}_{\forall i}$ all posterior vectors for correct class i and μ_i the mean posterior vector for this correct class. Unfortunately, we found this matrix not very representative, as the variances of posteriors depend on their values (higher variances for posteriors $>> 0$ and very low for posteriors closed to 0). The resulting matrix is therefore very similar to the soft confusion one.

Output covariance matrix

Here, we do not use any knowledge of the correct classes, we just compute the correlation of posteriors at the output of the net. The covariance matrix is given by definition:

$$\mathbf{C} = E\{(\mathbf{x}^T\mathbf{x} - \mu^T\mu)\} \tag{5.5}$$

where μ is the global posterior mean. For the visualization and class clustering, we have computed normalized covariance-matrix, with elements:

$$\rho_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}} \tag{5.6}$$

The ideal form of this matrix is again unity (no outputs correlated with each other).

Note on visualization of matrices

Except for the normalized covariance matrix, the visualization suffers from silence class being recognized more precisely than the other classes. The other elements then do not have sufficient resolution. It is then a good idea to visualize the matrices without the row and column corresponding to the silence.

5.3.3 Word error rate of HMM recognizer

The ultimate number while using TRAPs is the word-error rate (WER) of the HMM recognizer using merger-posteriors as features (after some post-processing). This number should be compared to the WER obtained using "classical" features, as MFCC's.

Chapter 6

Basic experiments: Stories–Numbers–Digits

Those experiments were conducted exactly on the same data Jain and Sharma used in their work, the results are therefore directly comparable. See conclusions at the end of this chapter 6.12 for the reasons we migrated toward different set of databases (called "reference" ones).

6.1 Data used

6.1.1 Band-classifier training: OGI Stories

this database is described in Sharma's thesis [32], section 2.2. We disposed already of a pfile with generated band log-energies, and with processed beginnings and ends of files (by flipping). There are 208 sentences, with 1010318 frames (2.8 hours). After selecting TRAPS, the number of frames is 810288 (discarding beginnings and ends and also some TRAPs with unusable labels (see below)). For the training of the net, the utterances are made shorter (cache problems of the SPERT board), their final number is 1620. 1400 are used for the training and 221 for the cross-validation (CV).

6.1.2 Merger training: OGI Numbers

this database is also described in Sharma's thesis [32], section 2.2 and we disposed of a pre-generated pfile as well. There are 3590 sentences and the source pfile contains 1034177 frames (2.87 hours of speech). Utterances contain all the numbers from the database (including "eleven", "ninety", etc).)After again discarding flipped beginnings and ends (all labels are used in Numbers), the total number of frames was 675177 (1.88 hours). The sentences were already short enough not to cause SPERT cache problems, so that no shortening was necessary. 3400 sentences were used for the training, 190 for CV.

6.1.3 HMM recognizer: Digits

Digits (not to confound with TI-digits!) are a subset of Numbers containing only digits from "zero" to "nine" and "oh". Digits are divided into train and test portion, for the HMM recognizer training. For both, we disposed of pre-generated pfiles.

The training part has 2547 sentences, with the original number of frames 713282 (1.98 hours). After discarding beginnings and ends (no label discarding here (no NN was trained on Digits)), the number of frames is 458582 (1.27 hours). The training part overlapped with the Numbers database used for merger training.

		Stories		Numbers	
label	index	count	perc%	count	perc%
d	0	5687	0.70	441	0.06
t	1	19606	2.42	22200	3.28
k	2	12956	1.60	2765	0.40
dcl	3	13761	1.70	521	0.07
tcl	4	27241	3.36	21215	3.14
kcl	5	17250	2.13	6800	1.00
s	6	50978	6.29	39880	5.90
z	7	14635	1.81	7003	1.03
f	8	17131	2.11	28389	4.2
th	9	5162	0.64	11728	1.73
v	10	9675	1.19	14977	2.21
m	11	20948	2.59	38	0.00
n	12	36695	4.53	50424	7.46
l	13	23079	2.85	916	0.13
r	14	20860	2.57	29717	4.40
w	15	15043	1.86	20485	3.03
iy	16	34554	4.26	32362	4.79
ih	17	38111	4.70	16640	2.46
eh	18	22200	2.74	13970	2.06
ey	19	20328	2.51	19085	2.82
ae	20	26146	3.23	162	0.02
ay	21	28054	3.46	53922	7.98
ah	22	49583	6.12	28219	4.17
ao	23	5867	0.72	4027	0.59
ow	24	16639	2.05	47529	7.03
uw	25	11086	1.37	28103	4.16
er	26	15137	1.87	2633	0.38
ax	27	11301	1.39	853	0.12
h#	28	220575	27.22	170173	25.20
total		810288		675177	

Table 6.1: Phoneme coverage in Stories and Numbers

The test part has 2169 sentences, with the original number of frames 843489 (2.34 hours). After discarding beginnings and ends, the number of frames is 626589 (1.74 hours). This set did not overlap with Numbers used for merger training and CV.

6.1.4 Phoneme set

The phoneme set used contains 29 phonemes, and is determined by the phonemes of Numbers:

```
d t k dcl tcl kcl s z f th v m n l r w iy ih eh ey ae
ay ah ao ow uw er ax h#
```

In Stories, with a richer phoneme set, TRAPs carrying the labels not appearing in Numbers are discarded. The band-classifiers are therefore trained on 29 classes as well.

The phoneme coverage in Stories and Numbers (for Stories already only the "correct" labels) is given in Table 6.1. We can see that especially the silence is heavily over-represented. Therefore, some experiments with balanced training were conducted.

6.2 HMM recognizer

was common to all experiments. Jain's setup was used without any changes. The recognizer uses context-independent phoneme models and multiple-pronunciation dictionary. It is built using HTK tools. The phoneme set contains 23 units:
`w ah n ow th r iy f s eh v ih tcl t uw kcl ay ey k ax ao z si`
The pronunciation dictionary contains multiple pronunciation variants for digits "zero" – "nine" and "oh". The steps of training are the following:

1. initialization of models using phonetic transcriptions by `HInit`.
2. 3 iterations of context-independent Baum-Welch re-estimation of the models `HRest`.
3. 5 iterations of context-dependent Baum-Welch re-estimation of the models (embedded-training) `HERest`.

The decoding was performed using `HVite` with cross-word transition log penalty of -25.5 (this number was found optimal by Jain).The scoring was done using standard tool `HResults`. Only 2168 files out of 2169 are used for the scoring. Results are reported in terms of word-recognition accuracy.

The standard **MFCC** coefficients with log-energy, Δ and $\Delta\Delta$ coefficients without cepstral-mean subtraction give **94.07** word accuracy on this setup. This, together with Jain's TRAP baseline accuracy of **93.15** served as comparison numbers.

6.3 The baseline: `base_scripts`

Why: to reproduce Jain's results and make the whole experiment run faster (before, all the processing including the generation of TRAPs was done on Sparc stations which was very slow.)
TRAPS: 101-point, TRAP-based mean and variance normalization.
Classes: 29 phonemes both for band-classifiers and the merger.
Nets: bands: 101–300–29, merger 15×29–300–29
Post-processing: log and PCA.
Results: the following results are provided here as well as for the other experiments:

- final phoneme recognition accuracy on the cross-validation set of Stories while training the band-classifiers for 3 bands (0-th, 5-th and 10-th) on Stories. In tables noted Scv0, Scv5 and Scv10.
- phoneme recognition accuracy while forward-passing the Numbers through the band-classifiers for 3 bands (0-th, 5-th and 10-th). In tables noted Nfwd0, Nfwd5 and Nfwd10.
- final phoneme recognition accuracy on the cross-validation set of Numbers in merger training. Noted Mcv in tables.
- the word recognition accuracy of the HMM-HTK recognizer on Digits, using class posteriors (with post-processing) as features.

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
35.51	40.41	37.75	23.55	31.80	30.60	81.45	**93.36**

Comments: On contrary to Jain, who run the entire experiment on Sparc stations with SPERT boards, great portion of the processing was moved to Linux. The training of nets is indeed very fast on SPERT, but the preparation of the data (especially creation of huge p-files) took horrible time on Sparcs. Now, the data preparation and all forward passes are done on Linux, just the net training is run on SPERT. This results in much smaller run-time (1 week before, 3 days now with almost no data-parallelization).

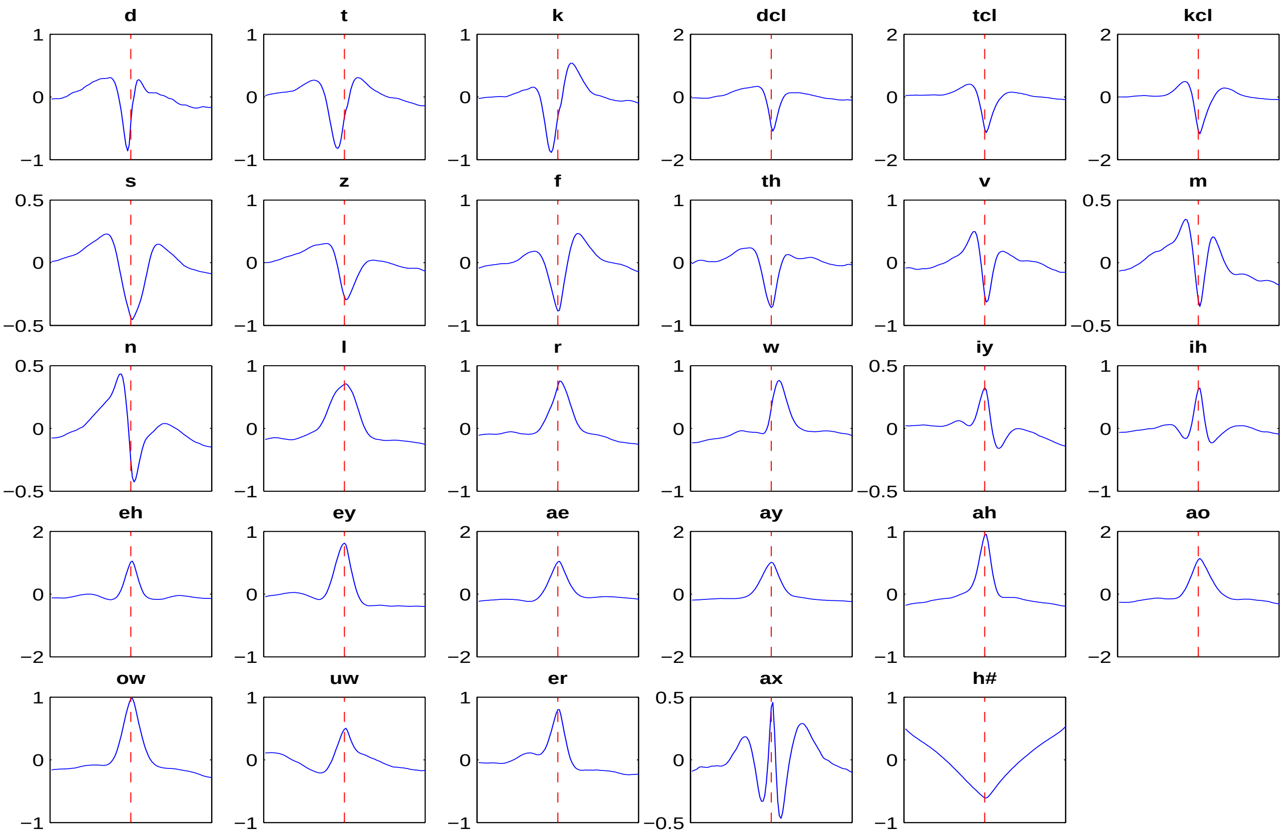

Figure 6.1: Mean TRAPs on Stories

A problem was found in Numbers and Digits databases: the covariance matrix from which PCA was computed (estimated on training part of Digits) was full of NaNs (not-a-number - result of division 0/0).

After tracking the problem down to the original data, it was found, that at one place, the training file for digits contained more than 100 frames with exactly the same values: sentence 1355 frames 140 ... 146, which corresponds to file `/net/data/numbers/release1/speechfiles/41/NU-4132.zipcode.wav`. As creation of TRAPS involves variance normalization, there was a division of 0/0=NaN. The generation of TRAPs and their normalization were corrected, so that if variance=0, the output is just a zero-trap.

It was further found, that when the merger was trained, it also saw those NaN data, as numbers are super-set of digits. Therefore, also data of Numbers were corrected and the merger was re-trained. The result on Digits recognition: 93.36% is naturally better than Jain's number 93.15%, but worse than MFCC baseline 94.07%.

Conclusions: Jain's experiment was successfully reproduced including a little patch. Everything runs faster. Good basis for following experiments.

6.3.1 Baseline – visualization

Mean TRAPs were generated on Stories (5th band) and are shown in Fig. 6.1. They correspond to Sharma's results in [32]. In addition, variance-TRAPs were generated (Fig 6.2) showing the variability of the 101 points for each label.

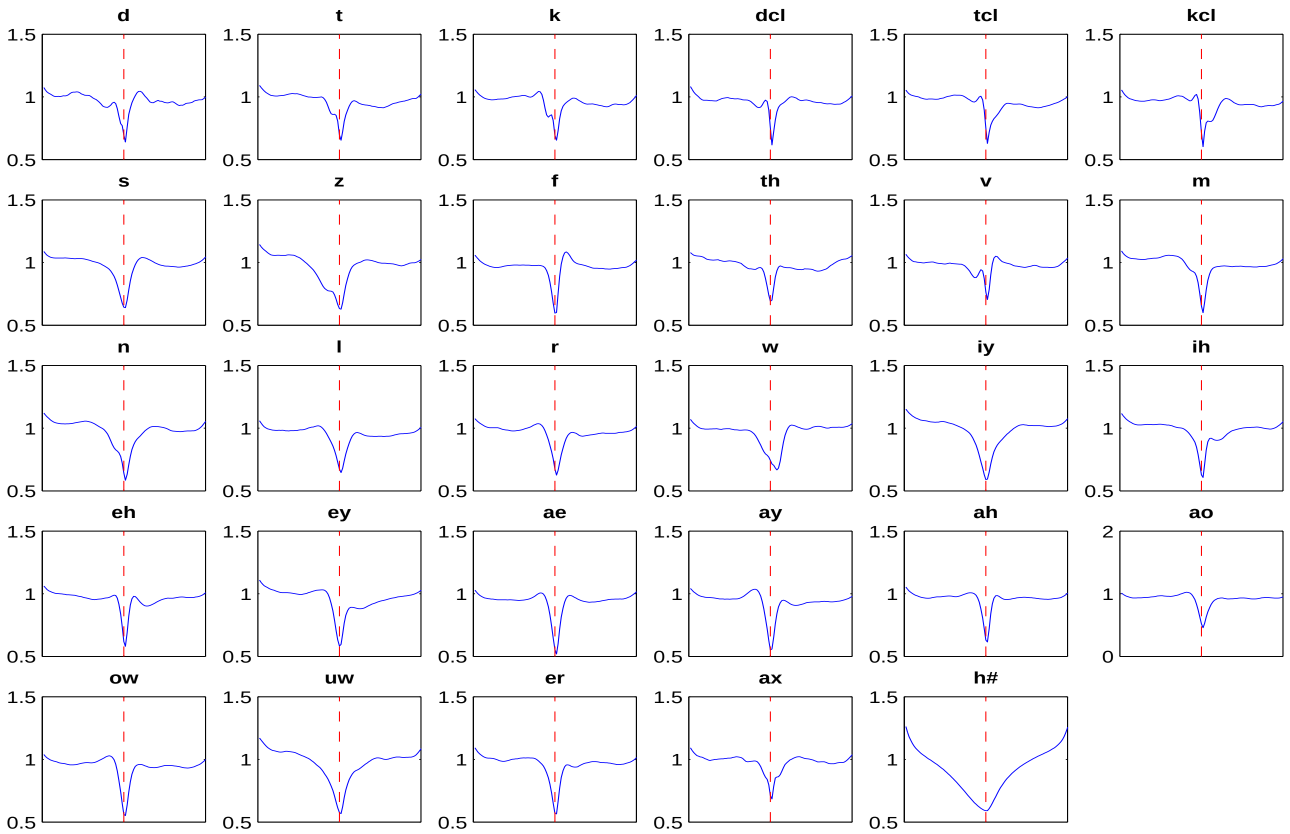

Figure 6.2: Variance (or better standard deviation) TRAPs on Stories

6.4 Running everything on Linux: `linux_base`

Why: running also the training of nets on Linux, to check if the results are comparable to SPERT and to measure the necessary times. The configuration is exactly the same as for `base`, except for the place if net training (SPERT vs. Linux).

TRAPS: 101-point, TRAP-based mean and variance normalization

Classes: 29 phonemes both for band-classifiers and the merger.

Nets: bands: 101–300–29, merger 15×29–300–29

Post-processing: log and PCA.

Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
35.52	41.02	37.77	23.73	31.80	32.61	81.72	**93.17**

Conclusions: Not exactly the same but very comparable result (before 93.36) for training of Linux. While the training of band classifiers was slightly faster on Linux (Pentium II – 500 MHz), SPERT still outperforms Pentia for bigger nets (merger): compare 9.50 hours (10 epochs) for SPERT and 11.50 hours (only 9 epochs) for Linux.

6.5 No Hamming windowing, sentence-based mean and variance normalization: `no_hamming_sent_norm`

Why: In the baseline experiment, Hamming windowing was done for historical reasons. The data is globally mean and variance normalized before nets, so that application of a constant on a single stream is canceled, and does not have any effect.

The sentence-normalization was tested because in the following experiment, we wanted to generate TRAPs on-line by the training software Quicknet. As Quicknet can not do mean and variance normalization of each input vector, it called for testing sentence-based one. Here however, the "old" way of the training was done (all TRAPs generated before) to ensure coherence with previous experiments.
TRAPS: 101-point, sentence-based mean and variance normalization
Classes: 29 phonemes both for band-classifiers and the merger.
Nets: bands: 101–300–29, merger 15×29–300–29
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
35.42	41.62	38.59	29.47	34.94	31.52	82.95	**93.84**

Conclusions: showed that sentence-based normalization gives better results than TRAP-based one. As for "classical" methods, this is probably due to more reliable estimation of mean and variance on the length of a sentence rather than on 101-point TRAP. Unfortunately, this also brings sentence-latency and is not suitable for tasks like AURORA. Sentence-based normalization was tested in several following experiments, but for the Reference traps (chapter 7) and SPINE experiments (chapter 8), we returned to the original TRAP-normalization.

6.6 No Hamming windowing, sentence-based mean and variance normalization, Quicknet generates the TRAPs: `qmk_no_hamming_sent_norm`

QMK in the name of the directories stands for "Quicknet makes kontext".
Why: This experiment was primarily meant to test the creation of TRAPs directly by the NN-training software Quicknet. While generating the TRAPs by Quicknet:

+ the size of generated pfiles is much smaller (actually we select just one stream of features) and the net training is faster, as there is less disk input-output.
– TRAP-based normalization cannot be done, as Quicknet cannot do it dynamically.
– We have a problem with the 'bad' labels (labels appearing in Stories but not in Numbers, see section 6.1.4). Quicknet cannot discard labels, so that an additional processing was necessary at the input, deleting features carrying those 'bad' labels, defining new sentence boundaries in places, where those labels originally were, and re-generating the context for those sentence boundaries[1]. The whole processing was quite complex and messy...
– as a result of new sentence boundaries, it was necessary to re-define the training and cross-validation set for the NN training, resulting in results not fully 100% comparable with the rest.

[1]remember, that a TRAP is not generated for a 'bad' label, bad frames carrying this label are still used for the context.

TRAPS: 101-point, sentence-based mean and variance normalization. Done by Quicknet.
Classes: 29 phonemes both for band-classifiers and the merger.
Nets: bands: 101–300–29, merger 15×29–300–29
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
35.32	42.23	38.48	28.70	35.13	31.82	82.81	**93.88**

Conclusions: Though minor differences in the bands, the overall result was very similar to the previous case, where the TRAPs were generated explicitly (merger 82.95, HTK recognizer 93.84). However, the pre-processing of feature stream is not straightforward, so that this approach was later abandoned.

6.7 Broad phonetic categories in bands `broad_categs_4`

Why: to test if broad phonetic categories in bands can perform well in classification of phonemes and in the word recognition. While band-classifiers were trained with 4 broad phonetic categories, the merger's task was to recognize phonemes.

The phonemes were mapped according to the following table:

phoneme	category
d t k dcl tcl kcl	STOP
s z f th v	FRICATIVE
m n l r w iy ih eh ey ae ay ah ao ow uw er ax	VOCALIC-SONORANT
h#	SILENCE

TRAPS: 101-point, sentence-based mean and variance normalization. Done by Quicknet.
Classes: 4 broad classes for band-classifiers. 29 phonemes for the merger.
Nets: bands: 101–300–4, merger 15×4–300–29
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
74.08	77.90	73.94	69.23	60.45	64.27	68.14	**86.66**

Conclusions: The training and recognition performance in bands obviously increased from phonemes to broad categories. Unfortunately, given the probabilities only for 4 categories per band, the merger is not able to recognize phonemes reliably and this is translated into a big hit on the overall recognition performance. However, this was a very simple experiment and the following points should be worked on:

1. by the limitation of inputs, we are also limiting the number of parameters of the merger. For a fair comparison, the size of hidden layer should be increased.
2. the phonetic categories are quite rough.
3. the merger should be trained to recognize just the categories coming from bands.

6.8 Tying closures with stops `closures_w_stops`

Why: we have seen that broad phonetic categories did not perform very well. A more gentle way to limit the number of classes is to tie just some phonemes. The most obvious is to consider closures and

the explosions as the same phoneme:
tcl+t $\Rightarrow$ t
dcl+d $\Rightarrow$ d
kcl+k $\Rightarrow$ k
Unlike the previous experiment, this tying was done also for Numbers, so that the phoneme sets for band-classifiers and the merger were coherent.
TRAPS: 101-point, TRAP-based mean and variance normalization. Pfiles again generated explicitly, as for the baseline (no TRAP generation using Quicknet).
Classes: 26 phonemes (stops tied with closures) both for band classifiers and the merger.
Nets: bands: 101–300–26, merger 15×26-300–26
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
35.83	40.94	37.27	25.38	33.01	30.46	81.42	**87.74**

Conclusions: We have seen comparable results with the baseline in bands and also during the merger training (merger CV accuracy for baseline was 81.45 – just slightly better than in this experiment). However, in the HMM recognizer word accuracy, we have seen a big hit (from 93.36 to 87.74). This is probably due to the fact, that the HMM recognizer uses the closures (see section 6.2), that we have linked with corresponding stops here. The models for closures then can not be reliably trained. A lesson from this is that the phoneme set used for TRAPs should be the most coherent possible with what we use for the HMM recognizer.

6.9 Balanced training #1: limiting the silence `less_silence_4x`

Why: The statistics (section 6.1.4) show clearly that the amount of data available for training different classes is very unbalanced. This and the following experiments aim at the balancing of training data. The balancing was done only for the band classifiers - the merger training was left intact, with the full phoneme set and full amounts of data for each class.

The most represented class is the silence. Here, we limited the number of silence TRAPs to $\frac{1}{4}$, so that its proportion approaches the most represented phoneme ('s' with 6.29%). After this limitation, the proportion of 's' was 7.91% while that of silence was 8.55%.
TRAPS: 101-point, TRAP-based mean and variance normalization. Pfiles again generated explicitly, as for the baseline (no TRAP generation using Quicknet).
Classes: 29 phonemes both for band classifiers and the merger.
Nets: bands: 101–300–29, merger 15×29-300–29
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
20.52	27.07	24.05	21.09	28.23	30.33	81.58	**92.70**

Conclusions: In bands, the CV and Numbers recognition accuracies are lower than for the baseline. This should not however be considered a hit, as we have limited the number of occurrences of class (silence), which is mostly recognized correctly. We see more coherence between the Stories and Numbers 'band-results' here. The merger CV result is even slightly better than the baseline 81.45. However, the HMM recognizer shows a slight hit from 93.36. This would suggest that limiting the silence is not a good step (silence is quite an important phoneme in the recognition). A thorough study of phoneme confusion matrix of the HMM recognizer would however be needed to prove this hypothesis.

6.10 Balanced training #2: suppressing the silence `no_sil_in_bands`

Why: Even in the previous experiment, we still saw a lot of silence. We wanted to test, how the *suppression* of silence in bands affects the phoneme recognition accuracy and the HMM recognizer WER.

The silence was deleted only from Stories. The configuration of merger training was kept, including the silences in Numbers.
TRAPS: 101-point, TRAP-based mean and variance normalization. Pfiles again generated explicitly, as for the baseline (no TRAP generation using Quicknet).
Classes: 28 (no silence) for band classifiers. 29 (including silence) for the merger.
Nets: bands: 101–300–29, merger 15×29-300–29. Here, the number in bands should have been 28 and the input of the merger 15×28. However, 29 was accidentally left in the scripts. The band classifiers therefore saw no data for class 28: the corresponding neuron in the output layer was therefore trained to produce always 0.
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
15.70	22.34	19.99	(8.36)	(14.92)	(11.30)	81.96	**92.77**

Conclusions: Band results for Stories confirm, that the main class responsible for the numbers around 30% on CV Stories was the silence. When it is completely suppressed, the numbers drop to 15-22%. The band-accuracies on Numbers are low, because the silence (the most represented class) could never be correctly recognized (remember: there was (accidentally) a neuron for silence, but trained to produce 0 all the time). The merger CV accuracy is however better that that of the baseline (81.45). This is unfortunately not translated in the improvement of HMM recognizer accuracy, confirming our previous conclusion, that the silence is necessary.

6.11 Balanced training #3: real balancing of classes `balance_stories`

Why: the limitation or suppression of silence was a playing just with one class. Here, the training data of *all* classes were balanced using a generalized down-sampler[2].

After the balancing, the amount of data was 149726 TRAPs, which was 5.4× less than for baseline (810288). The amounts of data for Numbers (merger training) are not changed.
TRAPS: 101-point, TRAP-based mean and variance normalization.
Classes: 29 for both band classifiers and the merger.
Nets: bands: 101–300–29, merger 15×29-300–29.
Post-processing: log and PCA.
Results:

Scv0	Scv5	Scv10	Nfwd0	Nfwd5	Nfwd10	Mcv	HTK W. accur.
14.09	20.22	18.11	14.51	23.41	26.13	81.35	**92.60**

Conclusions: If the band-classifier nets were well trained here, we should see similar performance on the recognition of Numbers data. Unfortunately, we do not reach the baseline's performance, which suggests badly trained nets. The CV-accuracy of the merger is just as good as the baseline and we

[2]Standard down-sampling requires integer decimation factor d. The samples taken must satisfy: $i \bmod d = 0$, where i is the sample-index. Generalized down-sampling allows for fractional decimation factors d. The indices of retained samples must satisfy: $i - d\left\lfloor \frac{i}{d} \right\rfloor \in [0, 1)$. Example: the "classical" decimation with $d = 5$ would produce indices 0 5 10 15 20 25 30 35 40 45 50 55. The fractional decimation with $d = 5.3$ produces 0 6 11 16 22 27 32 38 43 48 53 59. We can see, that the samples are not equi-distant, which is not hurting us. The amounts of data can be perfectly balanced using fractional decimation.

have a .7% hit in HMM recognizer. We can conclude, that the band-nets should be trained with all the available data.

6.12 Stories-Numbers-Digits: Conclusions

We have reproduced the baseline experiment with satisfactory results and performed some other experiments with broad phonetic classes and balancing of the training data. The results were represented in terms of band-classifier cross-validation accuracy, phoneme recognition accuracy on Numbers, CV accuracy when training the merger and finally of word recognition accuracy of the HMM recognizer.

There are however the following problems with the SND experiments:

1. the phoneme set for band-classifier training is not full and contains only the 29 phonemes present in Numbers. There are lots of 'bad labels' we have to discard.
2. Although Stories provide good phonetic coverage, Numbers contain a very limited vocabulary, so that the phonemes appear all the time in the same context. This may heavily bias the phoneme recognition accuracies.
3. Global numbers are biased by the distribution of data among classes. We need more detailed analysis of what is happening in bands and in the merger.

Some of those issues were addressed in the 'Reference' experiments described in the following chapter.

Chapter 7

Reference experiments: Timit and Stories

The reasons for switching to this experimental setup from SND were:

1. to have a coherent phoneme set for both band-classifier and merger training.
2. to dispose of phonemes in various context for both band-classifier and merger training.

Also, some visualization tools were produced for this setup allowing to see the confusion matrices and to do detailed per-class analysis. Those experiments are called 'Reference-TRAPs'.

7.1 Data, phonemes and evaluation

Lukáš Burget defined 4 data-sets:

part	purpose	sub-division	source	amount	comment
1	eventually for LDA training	train	Stories	165 min	-
2	Band-classif. training	train	TIMIT	106 min	-
		CV	TIMIT	20 min	-
3	Merger training	train	Stories	145 min	same data as for part 1
		CV	Stories	20 min	-
4	phoneme HMM recognizer	train	TIMIT	84 min	-
		test	TIMIT	49 min	different data from part 2

only parts #2 and #3 were used in the current experiments. If performed, recognition tests were done on SPINE (see next chapter).

The part for band-classifier training and CV is often mentioned as 'tnn' (trap-neural-network), while the part serving for merger training and CV is often called 'mnn'.

For TRAPs, it is good to have long signal files, so that we have less transitions on boundaries (and hence less problems with 'border' TRAPs). For Stories (mnn), this is not a problem, as the signal files are long. For TIMIT, Lukáš Burget concatenated the signals to one big signal file per speaker. To ensure context variability, the 2 sentences repeating for all speakers (`sa1` and `sa2`) were not included.

For **tnn-timit** we disposed of 783866 frames (including 50+50 flipped at the beginning and end of each file), resulting in 752966 TRAPs. We had 260 training sentences and 49 CV ones, totaling in 309 sentences. As in the previous setup, it was necessary to make the sentences shorter to fit in SPERT's cache. After some tests in the baseline experiment, the size of a sentence was chosen to be 1450, resulting in 599 'new' sentences for the training and 82 'new' sentences for the CV.

For **mnn-stories** we disposed of 1007823 frames (including 50+50 flipped at the beginning and end of each file), resulting in 987023 TRAPs. We had 183 training sentences and 25 CV ones, totaling in 208 sentences. As in the previous setup, it was necessary to make the sentences shorter to fit in SPERT's cache. The size of a sentence was chosen to be 500, resulting in 1267 'new' sentences for the training and 239 'new' sentences for the CV.

7.1.1 Phoneme set

The unique phoneme set was defined by Lukáš Burget as common for Stories and TIMIT (some phonemes had to be mapped). Table 7.1 gives the coverage of those phonemes on both data-sets. We can see, that sufficient number of examples is provided for all phonemes in both sets.

Unfortunately, even here we could not avoid some 'strange' or 'bad' label. Lukáš Burget has left some places in his label files void, as they contained phonemes not common to the 2 databases: namely the epinthetic closure 'epi' and glottal onset and stop 'q'. We have mapped all those to the 'other' label (number 42), as there can be no gaps in label files associated to frames. Only then we realized that the acoustics of 'epi' and 'q' is very different. Therefore, some experiments had to be re-done discarding the 'oth' label.

7.1.2 Evaluation

no HMM recognizer was trained at the top of Reference TRAPs. The evaluation of results was based on what we have seen during the training and cross-validation of nets. The following results are provided:

- final phoneme recognition accuracy on the cross-validation set of tnn-timit while training the band-classifiers for 3 bands (0-th, 5-th and 10-th) on Stories. In tables noted Tcv0, Tcv5 and Tcv10.
- phoneme recognition accuracy while forward-passing Stories through the band-classifiers for 3 bands (0-th, 5-th and 10-th). In tables noted Sfwd0, Sfwd5 and Sfwd10.
- final phoneme recognition accuracy on the cross-validation set of Stories in merger training. Noted Mcv in tables. This was the ultimate number.

7.2 The baseline: `base`

Why: baseline experiment with the same TRAP generation and net configuration as for SND, to assess the phoneme recognition accuracy in bands and at the output of the merger, and to do class-based analysis.
TRAPS: 101-point, TRAP-based mean and variance normalization.
Classes: 43 phonemes (including 'other') both for band-classifiers and the merger.
Nets: bands: 101–300–43, merger 15×43–300–43
Post-processing: no post-processing, remember that there was no HMM recognizer at the end.
Results:

Tcv0	Tcv5	Tcv10	Sfwd0	Sfwd5	Sfwd10	Mcv
25.58	30.24	27.01	22.92	26.66	23.39	**50.04**

Conclusions: The CV accuracies in bands are lower than in the SND setup, which is understandable, as we have much less silence in TNN-Timit than in the original Stories (14% here versus 27% before). What is more shocking is the accuracy after training the merger: from 80% for the SND experiment, we go down to mere 50%. A smaller proportion of silence in MNN-Stories (19% versus 25% before in Numbers) can be blamed, but is not solely responsible for 30% hit. The *variability of contexts* is probably the factor responsible for this huge difference.

		TNN-TIMIT		MNN-Stories	
label	index	count	perc%	count	perc%
b	0	12465	1.65	13282	1.34
d	1	16835	2.23	19615	1.98
g	2	7001	0.92	7186	0.72
p	3	20223	2.68	18984	1.92
t	4	32531	4.32	47976	4.86
k	5	28006	3.71	29794	3.01
dx	6	3550	0.47	3381	0.34
jh	7	4132	0.54	3461	0.35
ch	8	4676	0.62	4652	0.47
s	9	45753	6.07	51472	5.21
sh	10	11460	1.52	7662	0.77
z	11	20497	2.72	14817	1.5
f	12	15426	2.04	17348	1.75
th	13	4429	0.58	5311	0.53
v	14	8130	1.07	9906	1.00
dh	15	5874	0.78	8499	0.86
m	16	15896	2.11	22497	2.27
n	17	27044	3.59	42616	4.31
ng	18	4871	0.64	7352	0.74
l	19	24045	3.19	26155	2.64
r	20	17925	2.38	20942	2.12
w	21	9064	1.20	14974	1.51
y	22	3451	0.45	4181	0.42
hh	23	7518	0.99	8106	0.82
iy	24	29613	3.93	35051	3.55
ih	25	21910	2.90	38274	3.87
eh	26	20393	2.70	22505	2.28
ey	27	19132	2.54	20705	2.09
ae	28	20357	2.70	27182	2.75
aa	29	18649	2.47	24118	2.44
aw	30	7752	1.02	9648	0.97
ay	31	20343	2.70	28962	2.93
ah	32	13644	1.81	54794	5.55
ao	33	16089	2.13	13526	1.37
oy	34	3863	0.51	1831	0.18
ow	35	13871	1.84	17055	1.72
uh	36	2723	0.36	2573	0.26
uw	37	12595	1.67	12390	1.25
er	38	26161	3.47	20415	2.06
ax	39	12258	1.62	11384	1.15
ix	40	25046	3.32	5178	0.52
pau	41	104555	13.88	191146	19.36
oth	42	13210	1.75	40117	4.06
total	43	752966		987023	

Table 7.1: Phoneme coverage in TNN-Timit and MNN-Stories

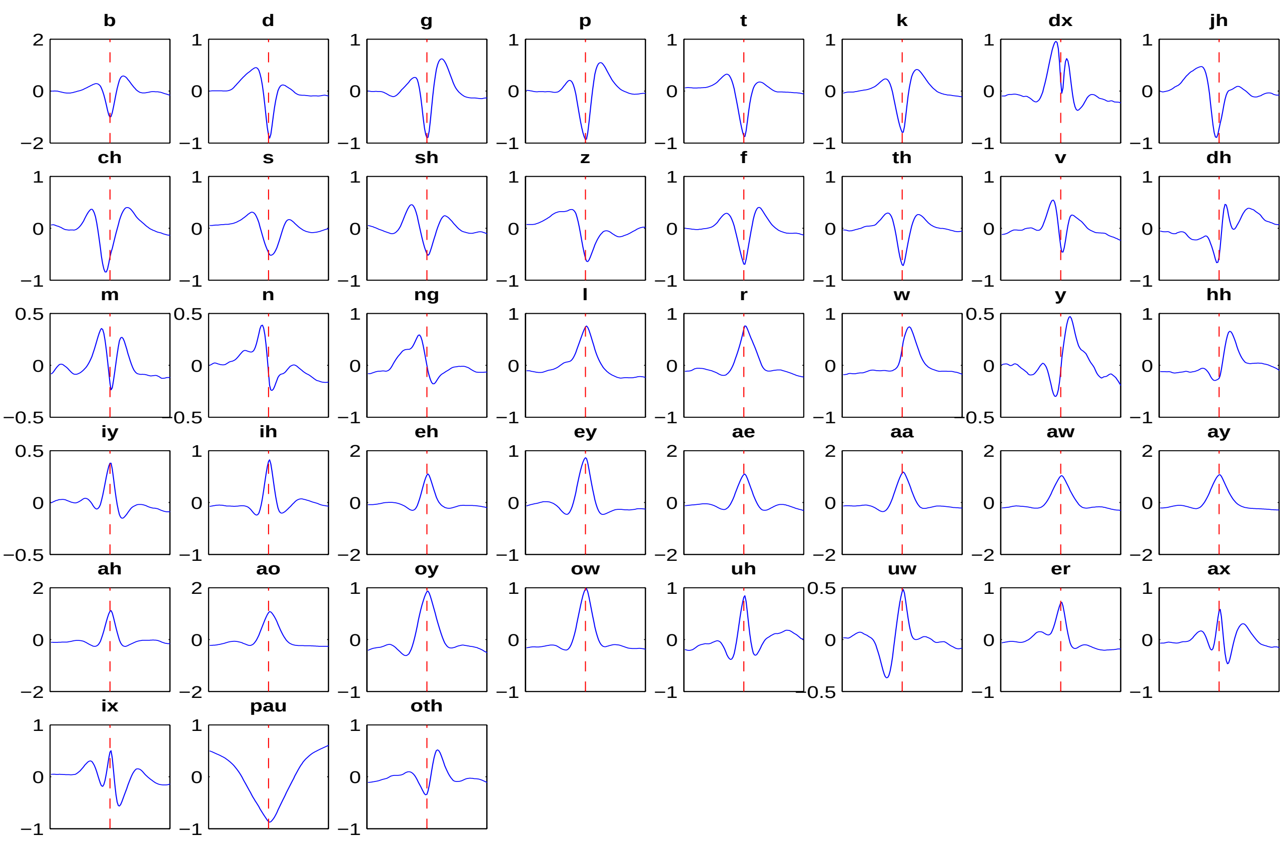

Figure 7.1: Mean TRAPs on TNN-Timit

Visualization and further conclusions: On this baseline experiment, new visualization and analysis tools were tested. First, the mean and variance TRAPs were computed (for the 5th band) to see, if they are coherent with what we have seen previously on SND. Comparison of figures 7.1 and 6.1 (mean TRAPs) and 7.2 and 6.2 shows, that we were probably not mistaken in the generation and selection of TRAPs - the shapes for phonemes carrying the same label are similar.

To asses the performance of TRAPs in bands, hard and soft confusion matrices and output normalized covariance matrices (see section 5.3.2) were computed for each band, based on the MNN-Stories data forward-passed through the band-classifiers. For band #5, they can be seen in Figure 7.3.

On all three matrices we can see, that the phonemes form clusters similar to broad phonetic categories. It is impossible to include all the figures for all the bands in this report, but it is interesting to see, how certain phonemes (especially liquids) "travel" among classes from band to band.

The y-axis of figures is completed by two important numbers: the first is the percentage of occurrences of the given phoneme in MNN-Stories while the second is the recognition accuracy ('hit-rate') of this phoneme. Not surprisingly, we see, that the silence is hit in most cases (82%).

Normalized covariance and soft-confusion matrices were used for automatic generation of broad phonetic categories in experiments:
`classes_bands_7_aut`, `classes_bands_10_aut` and `sconf_classes_bands_10_aut` – see sections 7.3, 7.4 and 7.5.

Similar matrices were generated also at the output of the merger – see Figure 7.4.

7.3 Automatically generated 7 broad classes: `classes_bands_7_aut`

Why: we have seen that the phoneme recognition performance in bands is quite poor. Broad phonetic classes should be better recognized, providing more reliable information to the merger. The classes can

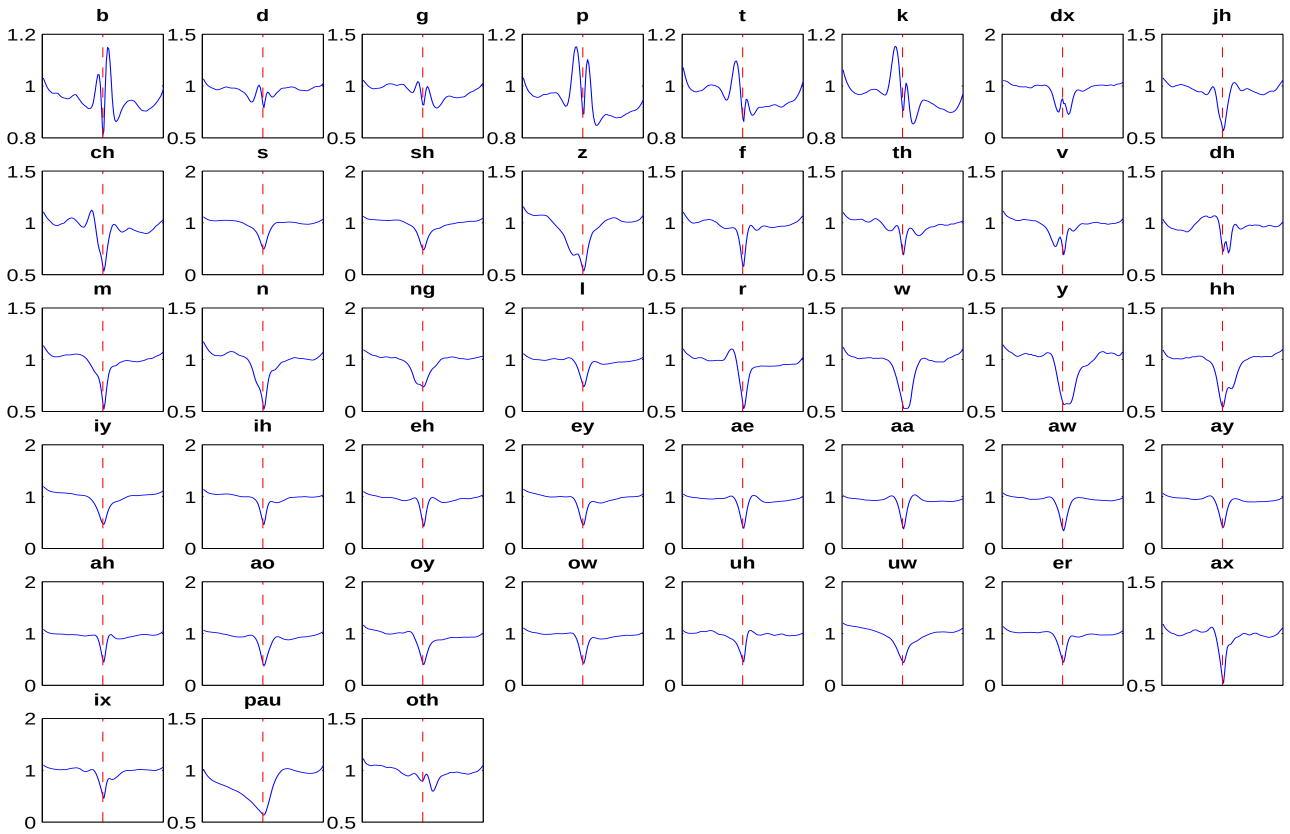

Figure 7.2: Variance (or better standard deviation) TRAPs on TNN-Timit

be generated using a-priori phonetic knowledge, or by automatic clustering on some of the confusion matrices. Moreover, they can be uniform for all the bands or band specific. In this experiment, we have chosen to generate the classes automatically based on the normalized covariance matrix in each band.

The number of classes per band was set to 7. For the class computation, we used Kajarekar's function `make_phn_clusters2.m` using Matlab functions `pdist`, `linkage`. The distance used is 'cityblock' and the linkage type is 'ward'.

The generated clusters for bands 0, 5 and 10 are summarized in the following table:

band	class0	class1	class2	class3	class4	class5	class6
0	b d g jh z v oth	dx dh hh	p t k ch s sh f th	pau	l r iy ey ae aa aw ay ao oy ow uw er	ih eh ah uh ax ix	m n ng w y
5	jh ch s sh z f th	pau	l r ey ae aa aw ay ao oy ow er	ih eh ah uh	b d g p t k	dx v dh m n ng y hh oth	w iy uw ax ix
10	dx jh ch dh	s sh z f y hh	b d g p t k	pau	th v m n ng l w oth	r ih eh ey ae aa aw ay ah er	iy ao oy ow uh uw ax ix

We can again observe "traveling" of certain phonemes among clusters.

TRAPS: 101-point, TRAP-based mean and variance normalization.

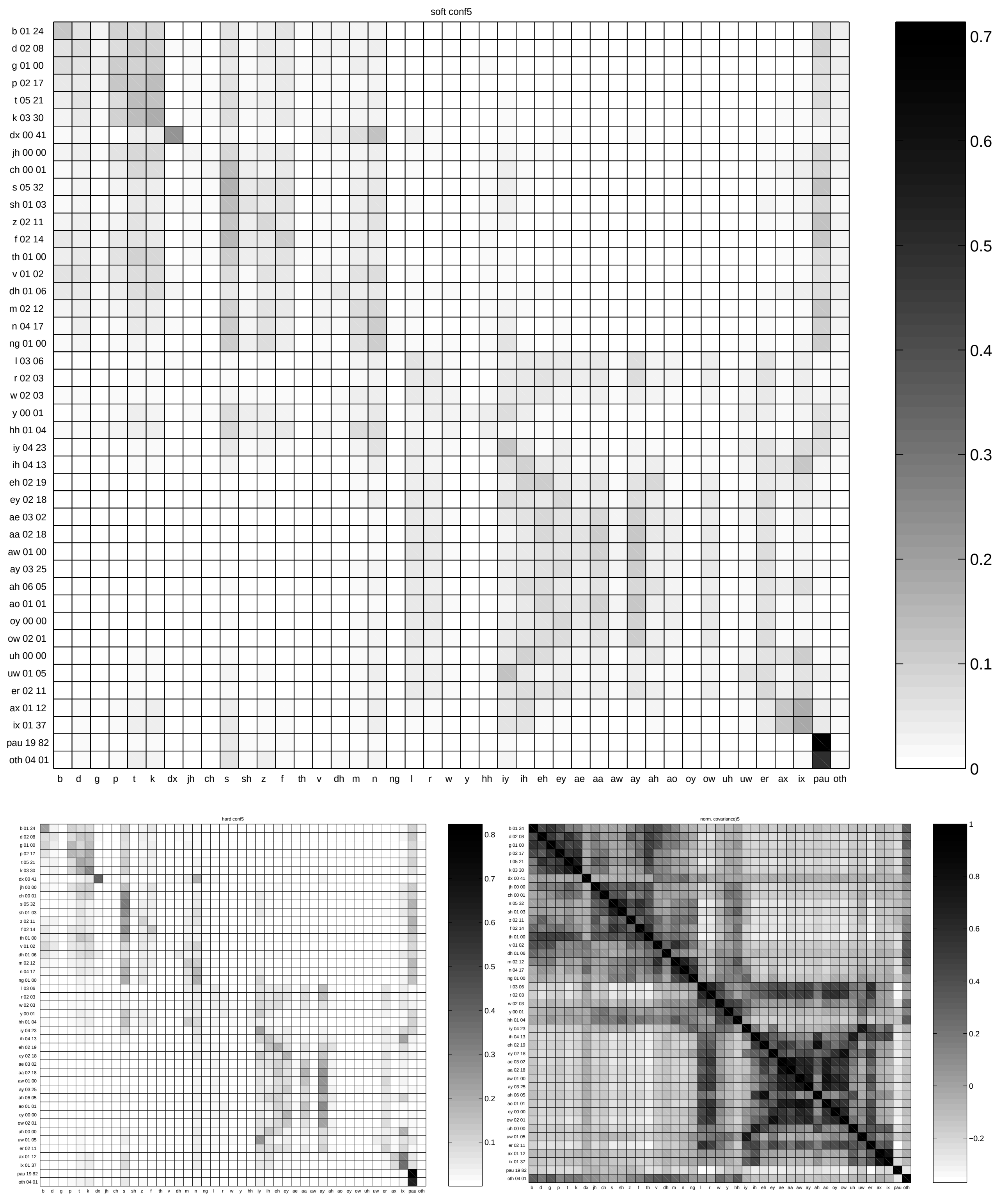

Figure 7.3: Soft, hard and normalized covariance matrix of band #5 (reference TRAPs–baseline experiment).

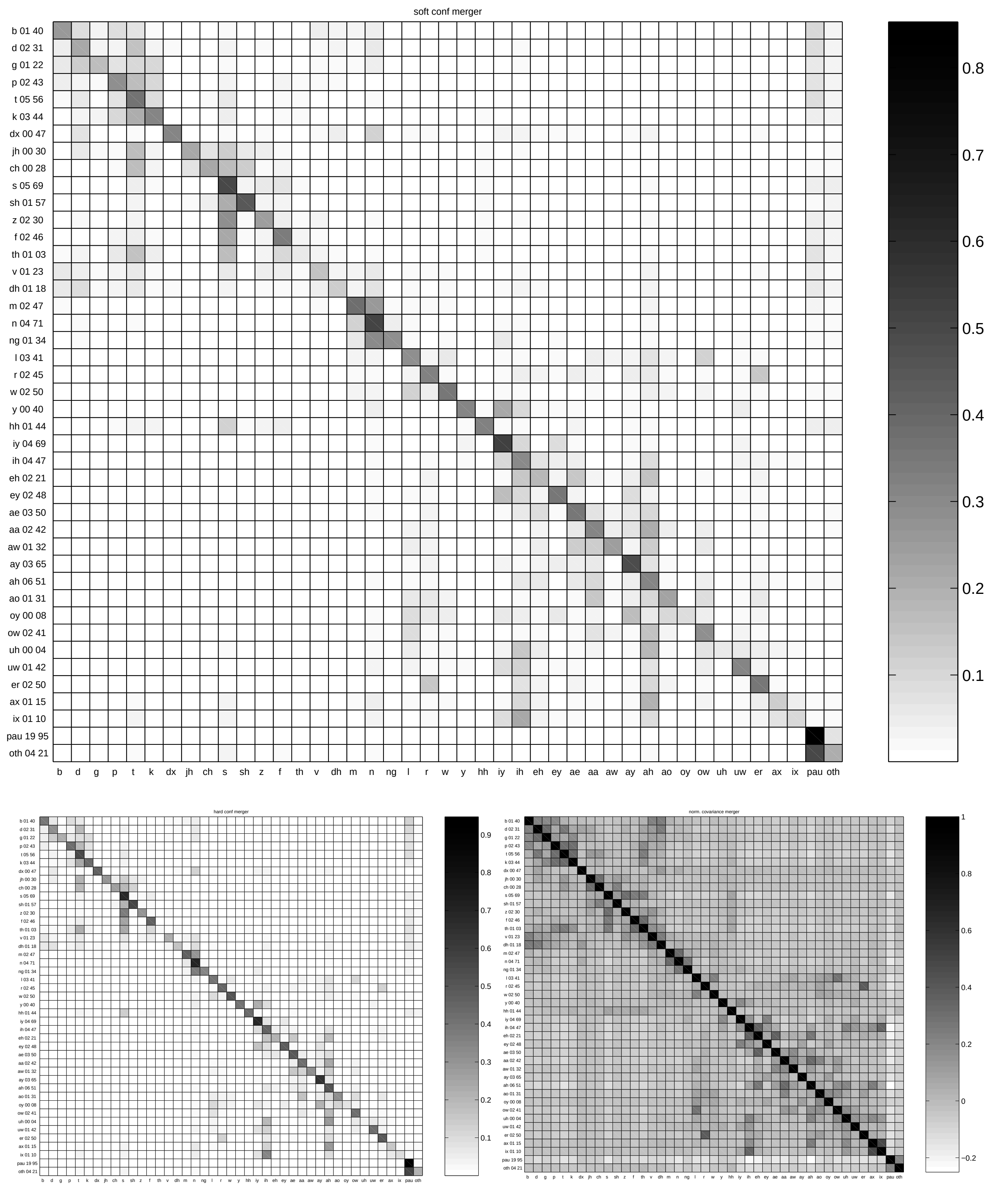

Figure 7.4: Soft, hard and normalized covariance matrix at the merger output (reference TRAPs–baseline experiment).

Classes: bands: 7 broad classes per band, generated automatically from normalized covariance matrix. merger: full set of 43 phonemes (including 'other').
Nets: bands: 101–300–7, merger 15×7–300–43
Results:

Tcv0	Tcv5	Tcv10	Sfwd0	Sfwd5	Sfwd10	Mcv
64.52	66.01	59.64	41.83	51.62	49.93	**46.44**

Conclusions: The band accuracies are not comparable with the previous setup, as we have lower number of broader classes. Obviously, the accuracy is higher. At the output of the merger, we have 4% hit. As it was mentioned already in section 6.7, by limiting the number of classes per band, we have also limited the number of merger's parameters. A fair comparison would require increasing the size of the hidden layer. Also, a simpler experiment with uniform classes for all bands should be conducted.

7.4 Automatically generated 10 broad classes: `classes_bands_10_aut`

Why: same reasons as for the previous experiment, but increasing the number of classes. 10 broad classes per band were generated using the same procedure as before. The generated clusters for bands 0, 5 and 10 are summarized in the following table:

band	class0	class1	class2	class3	class4	class5	class6	class7	class8	class9
0	m w y	n ng	l r ao oy	iy ey ae aa aw ay ow uw er	ih eh ah uh	ax ix	b d g jh z v oth	dx dh hh	p t k ch s sh f th	pau
5	w iy uw	ax ix	l r ey ow er	ae aa aw ay ao oy	dx ng y hh	v dh m n oth	jh ch s sh z f th	pau	ih eh ah uh	b d g p t k
10	r ae aa aw ay er	ih eh ey ah	s z f	sh y hh	iy ao oy ow uw	uh ax ix	dx jh ch dh	b d g p t k	pau	th v m n ng l w oth

TRAPS: 101-point, TRAP-based mean and variance normalization.
Classes: bands: 10 broad classes per band, generated automatically from normalized covariance matrix. merger: full set of 43 phonemes (including 'other').
Nets: bands: 101–300–10, merger 15×10–300–43
Results:

Tcv0	Tcv5	Tcv10	Sfwd0	Sfwd5	Sfwd10	Mcv
55.17	56.70	52.17	38.61	44.65	42.29	**48.22**

Conclusions: A more thorough analysis was done for band results in this experiment – confusion matrices for band #5 can be seen in Fig 7.5. The numbers accompanying the figures clearly tell that there are still huge differences in the recognition accuracy per class and that the confusion matrices are far from diagonal.

For the final number, we obtained better number than for 7 classes (as expected), but the baseline accuracy was not reached. Same comments apply for the number of parameters of the merger as for the previous experiment – it would be more fair to increase the size of the hidden layer.

7.5 Automatically generated 10 broad classes, based on soft confusion matrix: `sconf_classes_bands_10_aut`

Why: previous 2 experiments based the clustering on the normalized covariance matrix. This matrix is derived without any knowledge about the correct labels, but just by looking at the correlation of net's outputs. We hypothesized, that generation of classes based on one of the confusion matrices would bring more coherent classes and hence better final accuracies. The soft confusion matrices were used to generate 10 classes per band. The generated clusters for bands 0, 5 and 10 are summarized in the following table:

band	class0	class1	class2	class3	class4	class5	class6	class7	class8	class9
0	b d g jh v	dh hh	iy ih ah uh er	eh ey ae aa aw ay ao oy ow uw	m n ng	l r w y	dx	ax ix	p t k ch s sh z f th	pau oth
5	b d g p t k	v dh	jh ch s sh f th	z m n ng hh	y iy uw	ax ix	l r w ih eh ey ah uh er	ae aa aw ay ao oy ow	dx	pau oth
10	jh ch	ax ix	dx	sh iy ow uw	ih ey ah ao oy uh	s z f l w y hh	th v dh m n ng	pau oth	r eh ae aa aw ay er	b d g p t k

TRAPS: 101-point, TRAP-based mean and variance normalization.
Classes: bands: 10 broad classes per band, generated automatically from soft covariance matrix. merger: full set of 43 phonemes (including 'other').
Nets: bands: 101–300–10, merger 15×10–300–43
Results:

Tcv0	Tcv5	Tcv10	Sfwd0	Sfwd5	Sfwd10	Mcv
56.48	58.86	53.36	41.77	51.86	45.05	**48.35**

Conclusions: Improvement in bands which is unfortunately not translated to a big improvement at the output of the merger: just a fraction of % compared to the results with the normalized covariance matrix. The classes however look more "reasonable" than those generated using the normalized covariance matrices. Therefore, the selection of classes based on soft confusion matrix appears to be a good choice. All the comments of previous two experiments apply also here.

7.6 Baseline with discarded 'other' label: `base_no_oth`

Why: quite ugly confusion patterns of the 'other' class (Fig 7.3) lead us to investigate, what the 'other' label really was. A discussion with Lukáš Burget made it clear, that this label should not be considered one class but rather discarded (see section 7.1.1 describing the phoneme set of reference experiments). The 'other' label was here discarded both for bands and the merger training and CV, making the phoneme set size 42.
TRAPS: 101-point, TRAP-based mean and variance normalization.

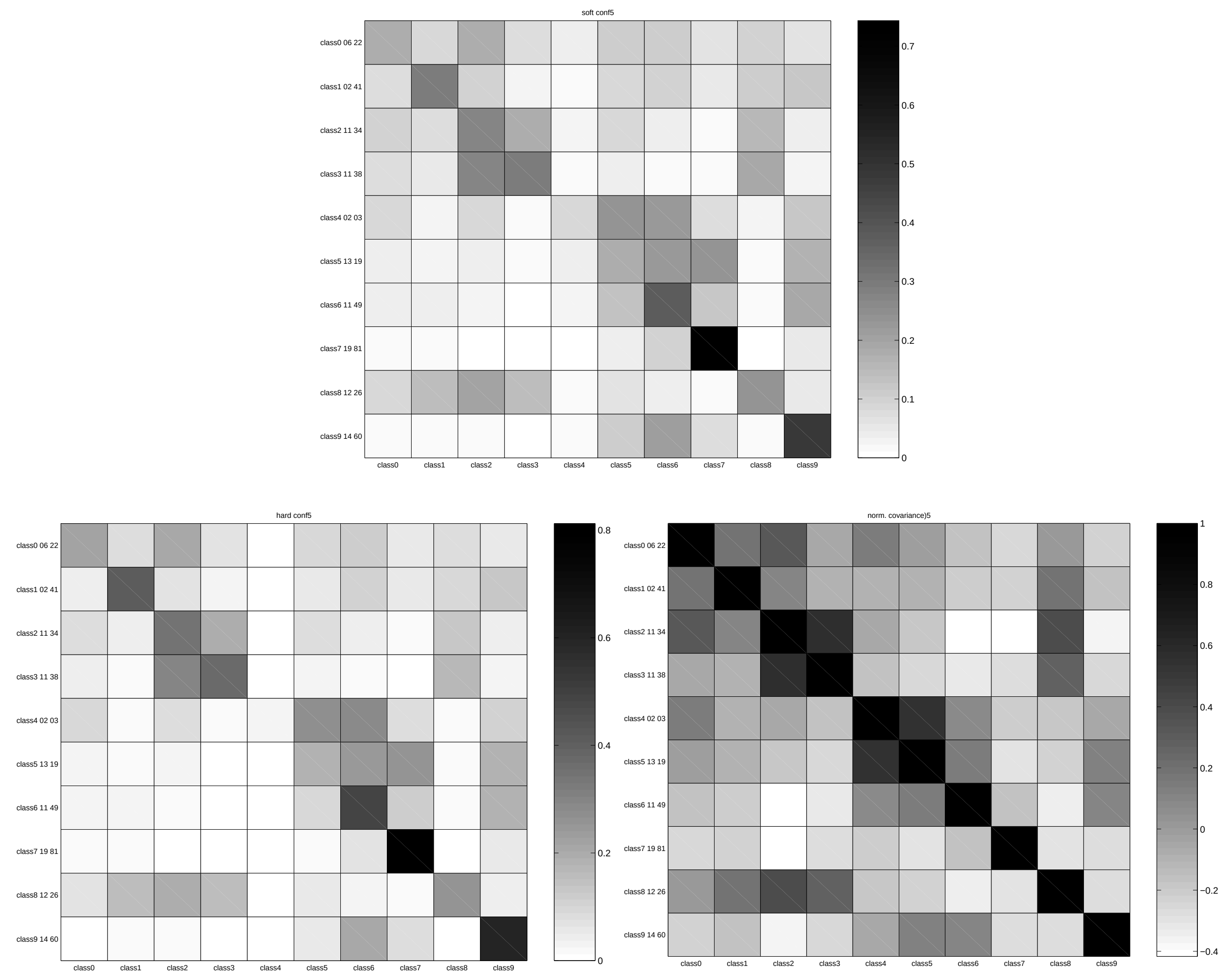

Figure 7.5: Soft, hard and normalized covariance matrix of band #5 (reference TRAPs–10 automatically generated classes).

Classes: 42 phonemes without 'other' both for band-classifiers and the merger.
Nets: bands: 101–300–42, merger 15×42–300–42
Results:

Tcv0	Tcv5	Tcv10	Sfwd0	Sfwd5	Sfwd10	Mcv
26.38	31.05	27.83	23.95	28.13	24.90	**52.66**

Conclusions: we have seen a systematic >1% improvement in bands, and more than 2.5% improvement at the output of merger. We should therefore discard the 'other' label from the experiments. A more general conclusion is that one should be *very careful* about the phoneme set used, check and re-check...

7.7 2-Band TRAPs – adjacent bands `2_band_51_300_300`

Why: Jain reported positive results with generating the bands for 2 bands. Theoretically, this approach is supported by the studies of co-modulation masking. We wanted to test 2-band TRAPs on the reference setup.

To ensure comparable number of net parameters with the previous experiments, we have made the TRAPs shorter- just 51-frames instead of original 101. This brings the size of a 2-band TRAP to

102, which is almost the same as 101. In this experiment, the 2 bands were adjacent, e.g. 0-1, 1-2, . . . 13-14. The total number of couples was 14.
TRAPS: 51-point, 2-band, TRAP-based mean and variance normalization, separately in each channel.
Classes: 43 phonemes including 'other' both for band-classifiers and the merger. Unfortunately, this experiment was done prior to the discovery that the 'other' label was hurting.
Nets: bands: 102–300–43, merger 14×43–300–43
Results:

Tcv0-1	Tcv5-6	Tcv10-11	Sfwd0-1	Sfwd5-6	Sfwd10-11	Mcv
29.01	34.01	31.47	24.89	29.57	26.35	**50.74**

Conclusions: we see a nice improvement from isolated bands to couples of bands. At the output of the merger, the improvement is however not spectacular: just 0.7%.

7.8 2-Band TRAPs – band skipping `2_band_1_skip_51_300_300`

Why: we should remember that the input features for TRAPs are log energies at the Bark filter-bank. The frequency characteristics for adjacent bands overlap, so that the 2 band outputs are necessarily correlated. Therefore, we conducted an experiment with couples of bands with the skipping of one band. The couples are: 0-2, 1-3, . . . 12-14, and their total number is 13.
TRAPS: 51-point, 2-band, 1-band skip, TRAP-based mean and variance normalization, separately in each channel.
Classes: 43 phonemes including 'other' both for band-classifiers and the merger. Unfortunately, this experiment was done prior to the discovery that the 'other' label was hurting.
Nets: bands: 102–300–43, merger 13×43–300–43
Results:

Tcv0-2	Tcv5-7	Tcv10-12	Sfwd0-2	Sfwd5-7	Sfwd10-12	Mcv
31.63	35.01	33.16	25.38	30.10	27.53	**51.14**

Conclusions: again an improvement in bands and again a slight improvement at the output of the merger.

7.9 2-Band TRAPs – a bit of brute force: `2_band_1_skip_101_500_1000_no_oth`

Why: This was a "last-cry" experiment breaking the logical suite which would be:

1. removing the 'other' label for 2-band TRAPs, letting the configuration intact.
2. lengthening the TRAPs to their original length (101 points).
3. increasing the number of parameters of one of the nets (band-classifier or merger), then both.

Due to the limited time, we did all the changes together, the hidden layer size for band-classifiers was increased to 500 neurons, that for the merger to 1000 neurons. This had two bad consequences:

1. we had to train both band-classifiers and the merger on Linux, SPERT board cache memory was exhausted.
2. the merger training time was very long (almost 3 days).

TRAPS: 101-point, 2-band, 1-band skip, TRAP-based mean and variance normalization, separately in each channel.
Classes: 42 phonemes without 'other' both for band-classifiers and the merger.
Nets: bands: 202–500–42, merger 13×42–1000–42
Results:

Tcv0-2	Tcv5-7	Tcv10-12	Sfwd0-2	Sfwd5-7	Sfwd10-12	Mcv
31.50	35.95	34.41	26.89	31.45	29.20	**54.04**

Conclusions: neat improvement in bands, and what is the best, a 4% improvement at the output of the merger. As it was mentioned at the beginning of this section, the question "Which of the changes is responsible for most of the improvement" is open.

7.10 Reference TRAPs – Conclusions

TRAP experiments were conducted on a set of databases providing, in our opinion, more reliable and realistic assessment of their performance than Stories–Numbers–Digits. It was found, that on a database with phonemes occurring in varying context, the phoneme recognition accuracy at the output of the merger is rather around 50% than 80% (SND experiments).

Experiments with *automatically generated broad phonetic classes* showed that the overall phoneme recognition accuracy is inferior to baseline results. The following issues are open:

- as mentioned, limitation of number of classes in bands implies the limitation of merger parameters. For a fair comparison, the size of hidden layer should be increased to match the number of parameters of the baseline.
- there are many ways to generate the classes: manual creation, and automated creation uniform for all bands should be tested.
- we need not necessarily have the same number of classes per band. Instead of a "hard" number, a variable number based on a distance measure (or mutual information) could be used.
- finally, a merger could be trained not to recognize phonemes but also broad phonetic classes. Those probabilities (after post-processing) could serve as input to an HMM recognizer [17].

2-band traps have shown a huge potential in performance. While testing 51-point TRAPs, we should remember that the normalization of input features was TRAP-based - the estimation of mean and variance was therefore on 2× less data than for the baseline. This calls for a normalization using a different window, or whole sentence.

In SND experiments, *sentence-based normalization* provided good results (the mean and variance are more reliably estimated). This approach was not tested with the reference setup, those experiments should be completed.

Chapter 8

TRAPS on SPINE

SPINE (Speech in Noisy Environments) is an evaluation run by the Naval Research Laboratory. The task a medium-sized vocabulary recognition on several military environments. The training and evaluation data from 2000 were used to assess performances of our features. These data come as stereo-recordings, but we disposed of data pre-segmented into speech and silence regions at CMU [33]. The recognizer used – SPHINX – came also from CMU [28].

8.1 The data

The **training data** consists of 140 conversation (each has 2 channels) completed with 18 directories with DRT (Diagnostic Rhyme Test) words with added noises. There are 15847 files (utterances) in the training set.

The **evaluation data** consists of 120 conversations (each with 2 channels). 12 first conversations were selected as the short evaluation set, including 1353 files. Most of the results reported here were obtained on this short set. There are 13265 files in the evaluation set.

"Improving" the data: we have found, that on side of conversations from ARMY scenarios (the noise one) contained files very unsuitable for any temporal processing (including TRAPS): a total silence (signal values oscillating around zero), followed by a sharp transition to noise and speech, and the same in inverse at the end of file. Visualization has shown very sharp edges in temporal trajectories coming from bands, that were hurting the performance of temporal LDA and TRAP methods. Those files were listed, the complete-silence parts detected, and finally, new signal files were generated. 1879 files were corrected in the training set and 1437 in the evaluation one. Experiments were done also on the original data, this report covers both data-sets.

While working with the improved data, we have also defined a new strategy to select the context for first 50 and last 50 frames in each utterance. Before, the 50 first and 50 last frames of each file were always flipped to create the necessary context. Here, we have taken the 50 left extra frames from the previous file of the same side of conversation, and similarly for the right extra 50 frames. "Improving" of the data together with this processing alleviated a lot of unpleasant artifacts at the edges of files.

The amounts of data in original and "improved" training and evaluation data are summarized in the following table:

set	orig-frames	orig-hours	improved-frames	improved-hours
train	3067237	8.52	3017499	8.38
eval	2540578	7.06	2497009	6.94

8.2 Labels and training sets

The labels were produced in 2000 at ICSI [7], unfortunately, out of 15847 training files, 804 were not labeled (they were actually left out as an evaluation set, but the labels were never produced). For the training of the nets, the files in the training set were randomized.

The training set was further divided into the following parts:

set	# files	comment
train_a	7500	served for the training of merger
train_b	8347	the labeled portion of train_b (7543 files) served for band-classifier training

The label files were available as strings of labels per frame. When shortening ("improving") the army files, care was taken to re-synchronize the labels.

8.3 Phoneme sets

The labels from come ICSI come with the set of 56 phonemes:
b d g p t k dx bcl dcl gcl pcl tcl kcl jh ch s sh z zh f th v dh m em n nx ng en l el r w y hh hv iy ih eh ey ae aa aw ay ah ao oy ow uh uw er axr ax ix h# q

There are however only 41 context-independent phonemes the SPHINX system uses:
+NOISE+ SIL AA AE AO AW AX AY B CH D DH E EH ER EY F G HH I II JH K L M N O OO OW P R S SH T TH U V W Y Z ng

Effort was done to map the 56 phonemes to something more similar to SPHINX phoneme set. Two mappings were made, one from 56 ICSI classes to 38 (SPHINX set without O, E, and +NOISE+), and then further limitation to 34 classes. Table 8.1 presents the two mappings.

Statistics computed in the train_a portion of the training set (improved data) in Table 8.2 show, that some phonemes are heavily under-represented in the ICSI 56 set, and some have even zero occurrences. This is improved for the small set with 34 phonemes, where we at least enough training examples for all classes.

8.4 Experiments on original data

those experiments were conducted prior to correcting the 'bad' army files.

8.4.1 The MFCC baseline: `htk_mfcc13_0_d_a_z`

Why: The very first experiment was done using MFCC features computed by HTK (the HCopy tool). There were 13 MFCC coefficients including c_0 (not the log energy). Delta and acceleration coefficients were computed using default window sizes (context of 2 frames on both sides). The mean was subtracted from the waveform on utterance basis. Original 16 kHz data were used with no down-sampling. The Mel-filterbank had 24 channels. Utterance-based cepstral mean subtraction (CMS) was done.

Results: are reported in terms of word error rate (WER) in percents for the complete (rarely) and small (always) evaluation sets, for context-independent (CI) and context-dependent fully tied (CD) models of Sphinx. The most important number is the WER with the final CD models for the small evaluation set.

CI 12utt	CI full	CD 12utt	CD full
76.1	77.5	**37.9**	42.9

The result 37.9% on the short set served as comparison number for all following experiments.

ICSI 56 → reduced 38		reduced 38 → small 34	
b	B	SIL	SIL
d	D	AA	AA
g	G	AE	EH
p	P	AO	AO
t	T	AW	AW
k	K	AX	EH
dx	D	AY	AY
bcl	B	B	B
dcl	D	CH	CH
gcl	G	D	D
pcl	P	DH	DH
tcl	T	EH	EH
kcl	K	ER	ER
jh	JH	EY	EY
ch	CH	F	F
s	S	G	G
sh	SH	HH	HH
z	Z	I	I
zh	SH	II	I
f	F	JH	JH
th	TH	K	K
v	V	L	L
dh	DH	M	M
m	M	N	N
em	M	OO	U
n	N	OW	OW
nx	N	P	P
ng	ng	R	R
en	N	S	S
l	L	SH	SH
el	L	T	T
r	R	TH	TH
w	W	U	U
y	Y	V	V
hh	HH	W	W
hv	HH	Y	Y
iy	II	Z	Z
ih	I	ng	ng
eh	EH		
ey	EY		
ae	AE		
aa	AA		
aw	AW		
ay	AY		
ah	AX		
ao	AO		
oy	AO		
ow	OW		
uh	U		
uw	OO		
er	ER		
axr	ER		
ax	AX		
ix	I		
h#	SIL		
q	SIL		

Table 8.1: Phoneme mappings for SPINE.

ICSI 56				small 34			
label	index	count	perc%	label	index	count	perc%
b	0	4180	0.29	SIL	0	511016	35.6
d	1	8493	0.59	AA	1	34745	2.42
g	2	9276	0.64	AO	2	22098	1.53
p	3	4057	0.28	AW	3	13272	0.92
t	4	15584	1.08	AY	4	24355	1.69
k	5	22328	1.55	B	5	9788	0.68
dx	6	3891	0.27	CH	6	5938	0.41
bcl	7	5608	0.39	D	7	29147	2.03
dcl	8	16763	1.16	DH	8	6414	0.44
gcl	9	9381	0.65	EH	9	101017	7.03
pcl	10	15144	1.05	ER	10	27798	1.93
tcl	11	41865	2.91	EY	11	41184	2.86
kcl	12	23121	1.61	F	12	19908	1.38
jh	13	7024	0.48	G	13	18657	1.29
ch	14	5938	0.41	HH	14	13907	0.96
s	15	55121	3.84	I	15	87384	6.08
sh	16	6360	0.44	JH	16	7024	0.48
z	17	16762	1.16	K	17	45449	3.16
zh	18	0	0	L	18	21446	1.49
f	19	19908	1.38	M	19	29843	2.07
th	20	10114	0.7	N	20	45413	3.16
v	21	4120	0.28	OW	21	47799	3.33
dh	22	6414	0.44	P	22	19201	1.33
m	23	29792	2.07	R	23	26125	1.82
em	24	51	0	S	24	55121	3.84
n	25	45304	3.15	SH	25	6360	0.44
nx	26	0	0	T	26	57449	4
ng	27	16240	1.13	TH	27	10114	0.7
en	28	109	0	U	28	29611	2.06
l	29	18265	1.27	V	29	4120	0.28
el	30	3181	0.22	W	30	21342	1.48
r	31	26125	1.82	Y	31	9329	0.64
w	32	21342	1.48	Z	32	16762	1.16
y	33	9228	0.64	ng	33	16240	1.13
hh	34	13813	0.96				
hv	35	94	0				
iy	36	43979	3.06				
ih	37	35630	2.48				
eh	38	18015	1.25				
ey	39	41184	2.86				
ae	40	33105	2.3				
aa	41	34745	2.42				
aw	42	13272	0.92				
ay	43	24355	1.69				
ah	44	28912	2.01				
ao	45	22004	1.53				
oy	46	195	0.01				
ow	47	47799	3.33				
uh	48	1176	0.08				
uw	49	28435	1.98				
er	50	17772	1.23				
axr	51	10026	0.69				
ax	52	20985	1.46				
ix	53	7775	0.54				
h#	54	511016	35.6				
q	55	0	0				
total	56	1435376			34	1435376	

Table 8.2: Phoneme coverage in SPINE: original ICSI 56 phonemes and re-mapped small set of 34 phonemes.

8.4.2 TRAPs from Stories and Numbers `traps_merger_on_numbers_1b_101_nn_300_300`

Why: first experiment with TRAPs on SPINE, all nets were taken from SND experiment `qmk_no_hamming_sent_norm` (see section 6.6).
TRAPS: 101-point, sentence-based mean and variance normalization.
Classes: 29 phonemes both for band-classifiers and the merger (same set as for Stories and Numbers).
Nets: bands: 101–300–29, trained on Stories, 15×29–300–29, trained on Numbers.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

CI 12utt	CI full	CD 12utt	CD full
75.5	-	**55.1**	61.3

Conclusions: quite very bad, almost 17% hit from the MFCC baseline.

8.4.3 Merger trained on SPINE `traps_merger_on_spine_1b_101_nn_300_300`

Why: we tried to "approach" the TRAPs to the target task here by re-training the merger on ICSI labels for SPINE. The band-classifiers were still from Stories:
`qmk_no_hamming_sent_norm`, section 6.6.
TRAPS: 101-point, sentence-based mean and variance normalization.
Classes: 29 phonemes both for band-classifiers. 56 ICSI phonemes for the merger.
Nets: bands: 101–300–29, trained on Stories, 15×29–300–56, trained on train_a set of SPINE using ICSI label files.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

CI 12utt	CI full	CD 12utt	CD full
65.0	-	**53.7**	59.9

Conclusions: 2% improvement against the experiment with merger trained on numbers. Still very far from MFCC (37.9).

8.4.4 Merger trained on SPINE - small set of 34 labels `traps_merger_on_34_spine_1b_101_nn_300_300`

Why: the analysis has shown that some of the ICSI labels are not present at all and some are under-represented. Here, we trained the merger using the smaller phoneme set, where all the phonemes have reasonable number of occurrences. The band-classifiers were still from Stories (`qmk_no_hamming_sent_norm` section 6.6).
TRAPS: 101-point, sentence-based mean and variance normalization.
Classes: 29 phonemes both for band-classifiers. 34 phonemes (small set) mapped from 56 ICSI phonemes for the merger.
Nets: bands: 101–300–29, trained on Stories, 15×29–300–34, trained on train_a set of SPINE using re-mapped ICSI label files.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

CI 12utt	CI full	CD 12utt	CD full
62.4	-	**47.9**	52.1

Conclusions: huge improvement over the first TRAP experiment on SPINE, showing again, how phoneme set is important. Unfortunately still quite far from the MFCC baseline.

8.4.5 Everything trained on SPINE - small set of 34 labels `traps_all_on_34_spine_1b_101_nn_300_300`

Why: a natural step was to train also the band-classifiers on SPINE data. The labeled portion of train_b set was used for this.
TRAPS: 101-point, sentence-based mean and variance normalization.
Classes: 34 phonemes (small set) mapped from 56 ICSI phonemes both for band-classifiers and the merger.
Nets: bands: 101–300–34, trained on labeled portion of train_b set of SPINE using re-mapped ICSI label files. Merger 15×34–300–34, trained on train_a set of SPINE using re-mapped ICSI label files.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

CI 12utt	CI full	CD 12utt	CD full
62.0	-	**49.7**	55.0

Conclusions: surprisingly, we have seen a hit from the previous result. Band-classifiers taken on another database (Stories) perform better than band-classifiers trained on the target data. We should however remember that the original army files (containing the sharp 'complete silence'–noise transitions) were used here. One can suppose that a net trained using those 'sharp-edged' data would perform worse than that trained on data without those edges.

8.5 Experiments on "improved" data

The army files were corrected and used in the following experiments.

8.5.1 MFCC baseline `idata_htk_mfcc13_0_d_a_z`

Why: first test with improved data using classical MFCC features. Same MFCC computation as in experiment `htk_mfcc13_0_d_a_z`, section 8.4.1.
Results:

CI 12utt	CI full	CD 12utt	CD full
72.4	-	**36.7**	

Conclusions: improvement of more than 1% over the previous baseline (37.9) confirming, that the improvement of army files aids also a feature extraction not based on temporal trajectories at all. 36.7 is a new baseline number for experiments on the improved data.

8.5.2 TRAPs from Stories and Numbers `traps_idata_baseline`

Why: to test the same approach as in the TRAP baseline for original data, we have taken nets from Stories and Numbers and just forward-passed the SPINE data through them to get the features. With the improved data however, we used uniquely TRAP-based mean and variance normalization. The nets were taken from the baseline experiment on SND, see section 6.3.
TRAPS: 101-point, TRAP-based mean and variance normalization. Nice handling of left 50 and right 50 frames of each file: taken from the previous and next file(s) from the same side of the conversation[1].
Classes: 29 phonemes both for band-classifiers and the merger (same set as for Stories and Numbers).
Nets: bands: 101–300–29, trained on Stories, 15×29–300–29, trained on Numbers.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

[1]if the previous file did not contain enough data for 50 frames, it was extended by the next previous file, and so forth. Same for the next file. Previous file to the first file in the conversation was the last one. Next file for the last file in the conversation was the first one (similar to a circular buffer).

CI 12utt	CI full	CD 12utt	CD full
78.4	-	**54.7**	

Conclusions: we have seen a small, but noticeable improvement from 55.1% of the previous TRAP-baseline on the original data. Verified, that the correction of army files helps also the TRAPs.

8.5.3 Nets from Reference experiments `traps_idata_tnn_mnn`

Why: section 6.12 summarizes the objections we had to nets trained on Stories and Numbers. Here, the nets trained on the Reference setup (band-classifiers on TIMIT and merger on Stories) were to be tested. The net weights were linked from the reference baseline experiment `base`, described in section 7.2.
TRAPS: 101-point, TRAP-based mean and variance normalization. Nice handling of left 50 and right 50 frames of each file.
Classes: 43 phonemes of the reference setup (including the questionable 'other' label) both for band-classifiers and the merger.
Nets: bands: 101–300–43, trained on tnn-timit, 15×43–300–43, trained on mnn-stories.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

CI 12utt	CI full	CD 12utt	CD full
70.8	-	**50.0**	

Conclusions: *big improvement* from previous results showing that the nets trained on the reference setup are better than those from Stories and Numbers for a task, where phonemes occur in different context (which is indeed the case of SPINE).

8.5.4 Band-nets from Reference experiments, merger on SPINE `traps_idata_tnn_merger_on_34_spine`

Why: as before, retraining the merger on the target task should aid the recognition performance. The merger was trained on the train_a set using already the small phoneme set containing 34 phonemes. Band-classifier nets were still from the reference baseline experiment `base`, described in section 7.2.
TRAPS: 101-point, TRAP-based mean and variance normalization. Nice handling of left 50 and right 50 frames of each file.
Classes: 43 phonemes of the reference setup (including the questionable 'other' label) for band-classifiers. 34 phonemes (mapped from 56 ICSI phonemes) for SPINE.
Nets: bands: 101–300–43, trained on tnn-timit, 15×43–300–34, trained on train_a set of SPINE.
Post-processing: log and PCA. The PCA matrix was computed on SPINE (train_a set).
Results:

CI 12utt	CI full	CD 12utt	CD full
60.0	-	**45.8**	

Conclusions: another spectacular improvement. Confirmed that band-classifiers trained on the reference setup and retraining of the merger on the target task provide reasonable performance. At this point, we have done some study of the distribution of output features; the next section describes our experiments on the post-processing of merger output.

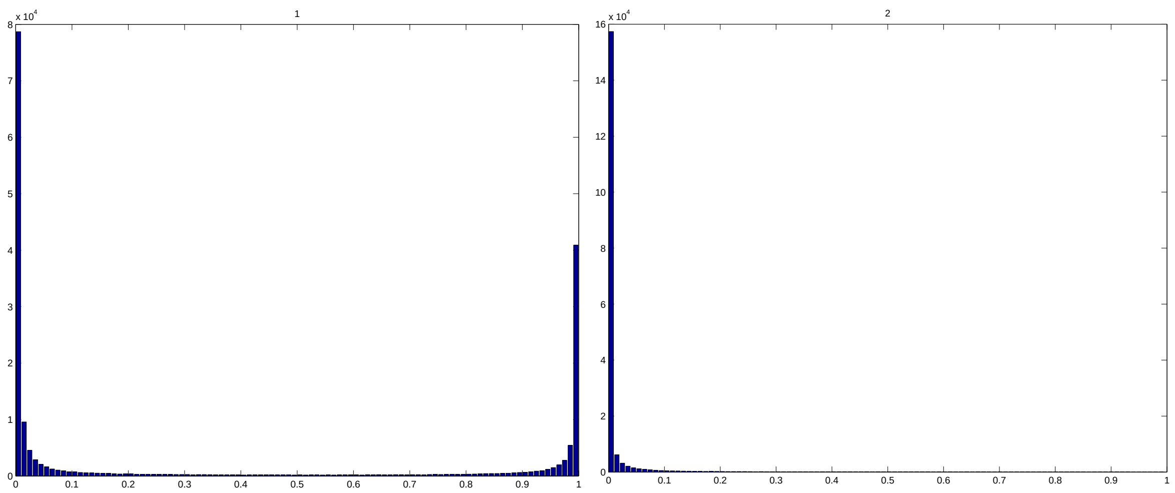

Figure 8.1: Histograms for softmax output of the merger: 'sil' and 'aa' outputs

8.6 How to post-process merger output

this section addresses the post-processing of merger output which was quite neglected in the previous work (just log of merger output [2] followed by a PCA). It was inspired by Sivadas's works on the post-processing of feature-net and by the article of Dan Ellis [6] on a similar topic.

We know that the HMM recognizer "likes the data Gaussian and de-correlated". We have therefore addressed the Gaussianization of the data, and some aspects of the PCA. From classical features, we also know that the adding of velocity and acceleration parameters aid, as well as mean and variance normalization. We have therefore tested those approaches together with PCA. The output probabilities of the merger from `traps_idata_tnn_merger_on_34_spine` (previous section) were used in all these experiments.

8.6.1 Processing of merger outputs

In all the previous experiment there was a soft-max non-linearity in the output layer of the merger. When we visualize the histograms for output probabilities, we see a very peaky distributions: bimodal with peak at 0 and 1 for the silence (most represented class) and unimodal for the other classes. See figure 8.1.

After the log (actually minus log, fig 8.2), the distributions become more Gaussian, but with an artifact at 0, corresponding to the original peak at 1.

When we look at log function (left panel of fig 8.3), we see that the expanding "tail" processes well the parts around zero, but fails to shape the region around 1. Therefore, we looked for a function having "tails" at both zero and one and found hyperbolic arcus-tangens $\text{atanh}(2x-1)$ to be a suitable candidate – see right panel of fig 8.3. Resulting histograms shown in figure 8.2 are nicer than for the log.

Looking at histograms and judging if they are nice or not nice is funny, but the recognition accuracies are more important. We have therefore run a recognition experiment for all of those features. The outputs were processed by a simple PCA to de-correlate.

[2]remember the merger has an output softmax-nonlinearity

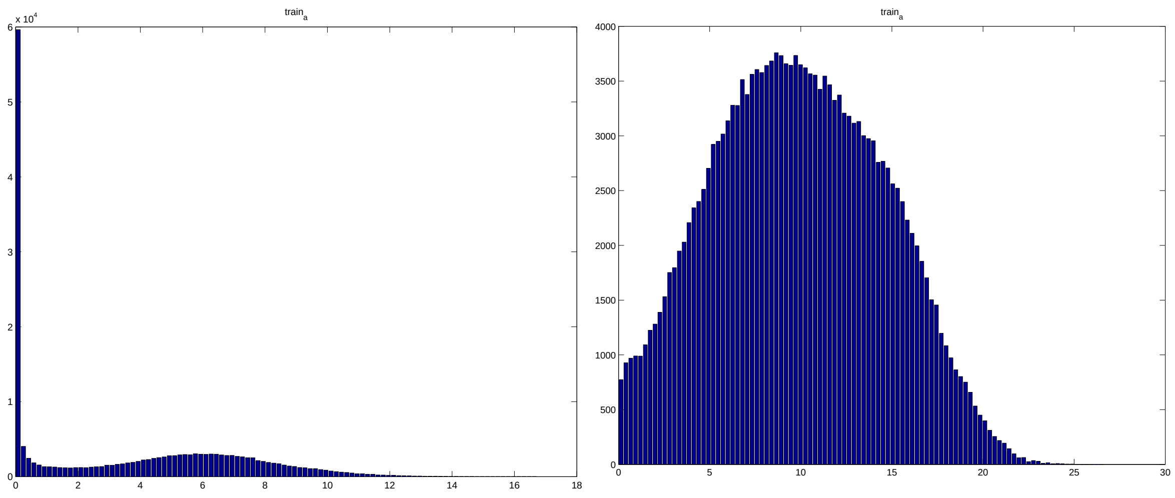

Figure 8.2: Histograms for log of softmax output of the merger: 'sil' and 'aa' outputs

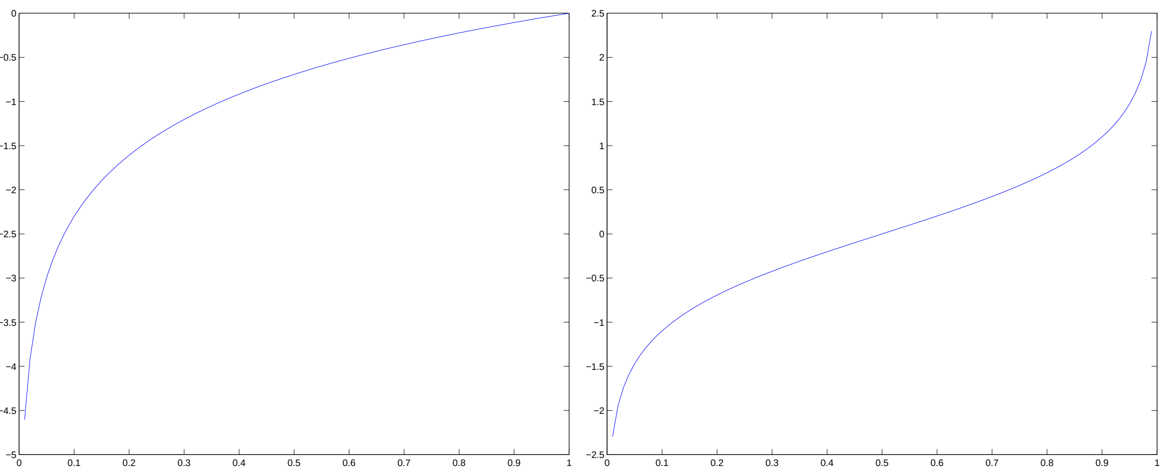

Figure 8.3: Log and atanh functions to post-process the merger softmax outputs

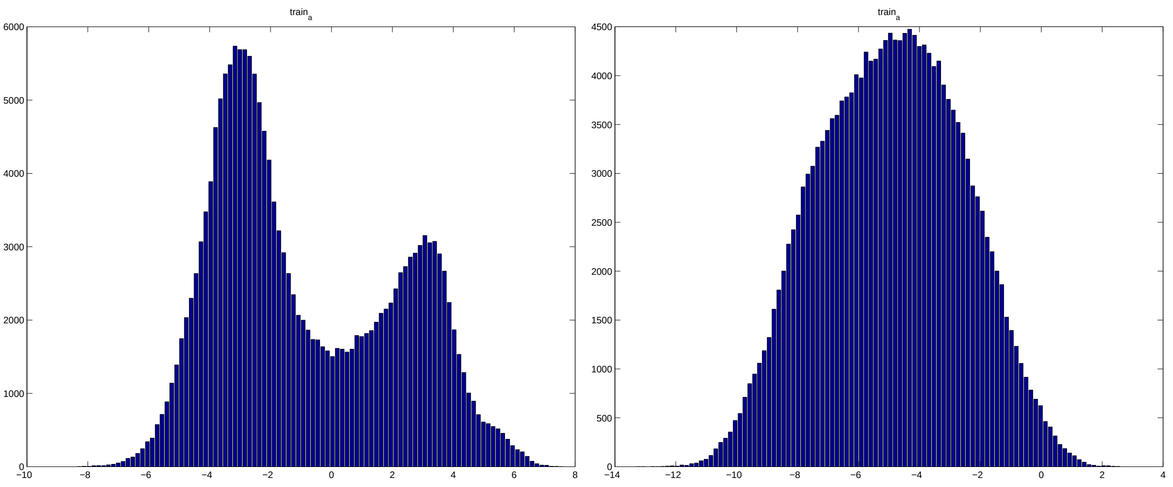

Figure 8.4: Histograms for atanh of softmax output of the merger: 'sil' and 'aa' outputs

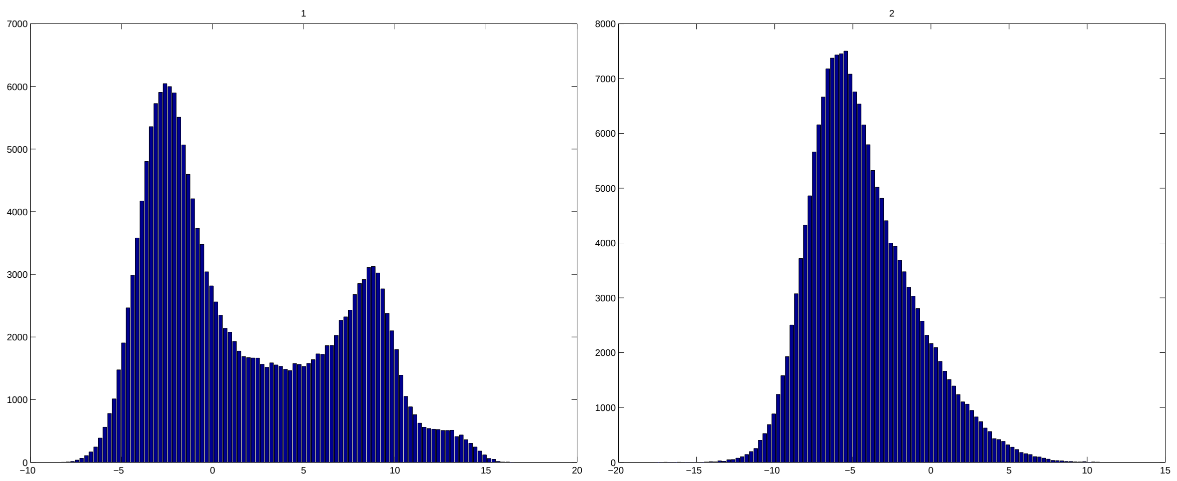

Figure 8.5: Histograms for linear output of the merger: 'sil' and 'aa' outputs

description	PCA	experiment	CI 12utt	CD 12utt
raw outputs, no log	NO	`traps_itmo34s_no_log`	89.1	crash
	YES	not done	-	-
log	NO	`traps_itmo34s_raw`	76.1	54.8
	YES	`traps_idata_tnn_merger_on_34_spine`	60.0	**45.8**
atanh	NO	`traps_itmo34s_atanh`	74.6	53.0
	YES	`traps_itmo34s_atanh_pca`	59.8	**44.7**

The results show, that the atanh produces not only nicer histograms, but also better WER.

The last method tried was to remove the softmax from the output layer of the net and use just the linear outputs. The histograms can be see in fig 8.5 and the recognition performances are summarized below:

description	PCA	experiment	CI 12utt	CD 12utt
linear outputs	NO	`traps_itmo34s_fwdlin`	68.0	49.6
	YES	`traps_itmo34s_fwdlin_pca`	59.2	44.5

The results are the best so far seen. The removing of softmax was used in the following experiments. However, the result is not far from hyperbolic arcus-tangens, so that for approaches relying on combining probabilities, atanh is the choice.

8.6.2 Further tricks on merger outputs

After the study of non-linearity processing, we were interested by additional post-processing of merger outputs. The following steps were tested:

1. adding delta and acceleration parameters. Those were computed on the 9-frame context.
2. PCA. We have experimented also with normalized PCA, where the rotation matrix is computed not from the original, but from the *normalized covariance matrix*. Its elements are given:

$$\rho_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}} \tag{8.1}$$

   where c_{ii} and c_{jj} are the variances of features. The denominator of equation above is actually the product of standard deviations. The normalized PCA can physically be done in 2 ways:

   (a) the data are globally variance normalized, and an ordinary PCA is computed.

(b) the PCA rotation matrix is computed from normalized covariance matrix, when applying this matrix, the data are first divided by standard deviations.

The dimensionality of the output data was always kept at 34, even if delta and acceleration parameters were used. PCA directions corresponding to 34 greatest eigen-values were selected.

3. sentence-based mean and variance normalization of the final features.

Below we summarize the results of those experiments, which, we agree, are not well theoretically founded, but rather justified by "people do it with MFCC's so why shouldn't it work here…". The base for all this processing were the linear outputs of the net (without softmax in the output layer).

Delta and acceleration coefficients added

experiment	PCA	CI 12utt	CD 12utt
`traps_itmo34s_fwdlin_d_dd`	none	80.5	57.9
`raps_itmo34s_fwdlin_d_dd_pca34`	normalized	63.8	**42.8**
`traps_itmo34s_fwdlin_d_dd_dyn_pca`	standard	63.3	43.5

We can conclude, that the normalized PCA brings better results than the standard one. This is probably due to the fact, that as all the features are globally variance normalized, their dynamic ranges are no more important, the PCA can concentrate only on the de-correlation of features.

Sentence-based mean and variance normalization

We have experimented with the mean and variance normalization before and after the PCA (which retained only 34 directions):

experiment	description	CI 12utt	CD 12utt
`traps_itmo34s_fwdlin_d_dd_pca34_mn`	PCA + mean norm.	68.4	**42.7**
`traps_itmo34s_fwdlin_d_dd_pca34_mvn`	PCA + mean and var. norm.	68.3	43.7
`traps_itmo34s_fwdlin_d_dd_mn_pca34`	mean norm. + PCA	65.7	42.9

The best results are obtained by mean normalization only, followed by the PCA. Other experiments should be conducted with retaining more PCA directions, unfortunately, we could not do them in the time given.

8.7 Brute force on SPINE

As the last cry we have done an experiment summarizing all the improvements we have seen before, and going beyond by the increasing of hidden layers of all nets. This experiment features long (101-point) 2-band traps with one band skip, with band-classifiers trained on TIMIT containing 500 hidden neurons (before 300), with a merger trained on train_a part of SPINE with 500 neurons in the hidden layer (we wanted to train a hidden layer with 1000 neurons, but the training would last forever…). Most powerful post-processing was deployed, too.

TRAPS: 101-point, 2-band, 1-band skip, TRAP-based mean and variance normalization, separately in each channel.

Classes: 42 phonemes from Reference experiments, without 'other' for band-classifiers. 34 phonemes (mapped from 56 ICSI phonemes) for SPINE.

Nets: bands: 202–500–42, trained on tnn-timit, 13×42–500–34, trained on train_a set of SPINE.

Post-processing: removed softmax and PCA (`traps_i2b_mo34s_fwdlin_pca`) and deltas and accelerations, normalized PCA and sentence-based mean normalization:
`traps_i2b_mo34s_fwdlin_d_dd_pca34_mn`. The PCA matrix was computed on SPINE (train_a set).

Results:

	CI 12utt	CI full	CD 12utt	CD full
`traps_i2b_mo34s_fwdlin_pca`	63.0	-	43.8	-
`traps_i2b_mo34s_fwdlin_d_dd_pca34_mn`	63.0	-	**41.2**	-

Conclusions: as expected, we have seen the best numbers here. Here we have stopped the experiments and are praying that the ROVERing on SPINE, Chaojun's system.

8.8 TRAPs on SPINE: Conclusions

In the last experiments on SPINE, only TRAP-based mean and variance normalization were used. In the SND experiments, we have seen some improvement when going from TRAP based to sentence-based normalization. This should be tested on SPINE. We can go even beyond: we know, that one speaker is always at one side of a conversation, so that the normalization could be conversation and channel based. That would provide us with more reliable estimates of means and variances.

The training of band-classifiers on SPINE was abandoned, as they did not perform so good as those trained on other database. This approach was unfortunately no more tested in the "improved" data, which should make the band-classifier training more consistent. Recent results of Jain show, that especially for 2-band TRAPs, this should be a very promising way.

Chapter 9

TRAP summary

9.1 Conducted experiments

Although being comparable MFCC's on the small vocabulary task with context-independent phonemes, TRAPs seem to have hard time to reach the performance of "classical" features on LVCSR task using context-dependent tied models. The recent experiments however show promising approaching of TRAP performance to the MFCC (41.2 versus 36.7% WER on SPINE).

The availability and quality of *labels* seems to play crucial role in the TRAP work. In sections 8.4.3 and 8.4.4, we have seen a dramatic improvement from 53.7 to 47.9% just by re-mapping the ICSI 56 phoneme set to a smaller one containing 34 phonemes. We are however still far from optimum, as we tune the TRAPs to context-independent phonemes (often not the same, as the recognizer is using) while the LVCSR systems use CI-models just for the initialization. It is however difficult to train any net aiming at the discrimination of classes finer than CI-phonemes due to their big numbers (e.g. 2600 tied states in SPHINX).

We have done experiments at the output of the merger, but similar care should be taken while *processing the band-classifier outputs* for the input to the merger. Currently, a log of softmax output is taken. We have tested (experiment not documented in the report), that log performs better than taking just the raw output. It would be worth to investigate if a hyperbolic arcus-tangens or removing the softmax do not bring similar improvement as at the output of the merger.

We use the neural nets as a black box, without changing their architecture (which is determined by Quicknet), number of hidden layers (Quicknet supports just 1), learning strategies, etc etc. There is certainly a potential of improvement here.

PCA applied at the output of all the processing is the simplest and probably also the worst way to de-correlate the features for HMM recognizer. LDA and newly MLLT are certainly of interest. As it was already mentioned, the mean normalization could be done on conversation and channel basis for SPINE, where this information is available.

9.2 Current and future work on TRAPs

The TRAP framework is currently being investigated by the research groups of OGI Portland, VUT Brno and by ICSI Berkeley[1].

9.2.1 TRAPs in AURORA and VAD

Pratibha Jain is the main person working on TRAPs nowadays. Her results include:

[1]Note that the chief "ideologist" of those approaches, Hynek Hermansky, has an appointments at OGI, ICSI, and VUT Brno, so this combination is not surprising :-)

- a study of manner-trained TRAPs (manner categories including "vocalic/approximant", "nasal", "fricative", "flap", "stop" and "silence") aiding the robust feature extraction on AURORA-2002 database (Aurora-2 and Aurora-3 tasks). The feature extraction named QIO (Qualcomm-ICSI-OGI) [1] includes noise suppression by Wiener filtering in power-spectral domain, Mel-spectral analysis, linear-discriminant analysis and voice-activity detection (VAD) on the terminal side. At the server, the features are on-line mean and variance normalized, and Δ and $\Delta\Delta$ features are appended. Frames are dropped according to the VAD decision from the terminal.

  To this feature-stream, Jain is adding scaled-down version of TRAPs described in the previous chapter: instead of 101-frame context, only 19 frames are used (9 left and 9 right) in order to satisfy the total algorithmic latency imposed by AURORA. 19-point TRAPs are mean and variance normalized, and further reduction is obtained by projecting them on 10 PCA basis. The nets are also scaled down (10 inputs, 25 hidden units and 6 outputs), and another dimensionality reduction takes places before merging the streams from bands. Jain reports 1.0-1.5% absolute improvement over the QIO features. In her opinion, further improvement of results should be obtained by using more representative data to train the classifiers (classifiers were trained on US-English only, while Aurora test data were multi-lingual).

- Brian Kingsbury with Pratibha Jain and Andre Adami have obtained very promising results using TRAPs for voice-activity detection in a large-vocabulary task [23]. SPINE-2001 task was similar to SPINE-2000, described earlier in this work (section 8.1). The data were recorded in simulated military environments, and came with long portions of silence between utterances. The task of great importance was therefore the separation into speech and silence regions.

  Three methods of VAD were compared while during Jain's summer internship at IBM: classical *HMM-GMM* detector based on short-term energy and degree of voicing, *TRAPS-M* system, where band classifiers and merger were trained to recognize three classes (voiced, unvoiced, silence) and the output of merger was post-processed by a simple median-filtering, and *TRAPS-H*, where the merger was used to produce likelihoods for the Viterbi search. All three methods were tested with the sophisticated IBM recognizer including Vocal-tract-length normalization (VTLN) and speaker adaptive training (SAT) making use of feature-space maximum likelihood linear regression (FMLLR). The results have shown, that TRAPS-H system outperformed HMM-GMM and TRAPS-M by 1% in word-error rate.

9.2.2 LDA-TRAPs and Merger-only

František Grézl is making significant progress in TRAPs, too. In [10], he compares "classical" TRAPs with LDA-TRAPs and "merger-only" systems.

The LDA-TRAP system is shown in Fig 9.1. Instead of passing the full 101-point trajectory to a band classifier, a dimensionality reduction preserving the discriminability is done first using LDA. We expect better generalization properties of such a classifier, because for this configuration, the size of hidden layer is relatively bigger than in the "classical case" (101-300).

Another proposed modification consists in removing completely the band-classifiers following the LDA (Fig. 9.2). The idea behind is that LDA is already trained to discriminate phonemes so that the band-classifiers are no more necessary. The advantage of this approach is that there is just one neural net to train.

The comparison of all three systems is done on the baseline TRAP experiment (section 8.5.2) with the word-error rate of 54.7%. The TRAP-LDA system provides accuracy of 53.8% while the "merger only" gives 47.1%. Note that those improvements were obtain without additional "tricks", as final feature mean and variance normalization, Δ and $\Delta\Delta$ computation, etc. This gives a lot of room for further improvements.

The comparison of "standard" TRAPs and TRAP-LDA systems was carried out on Aurora-2 task (only US-English – TI Digits with added noises). Here, the best results were obtained with

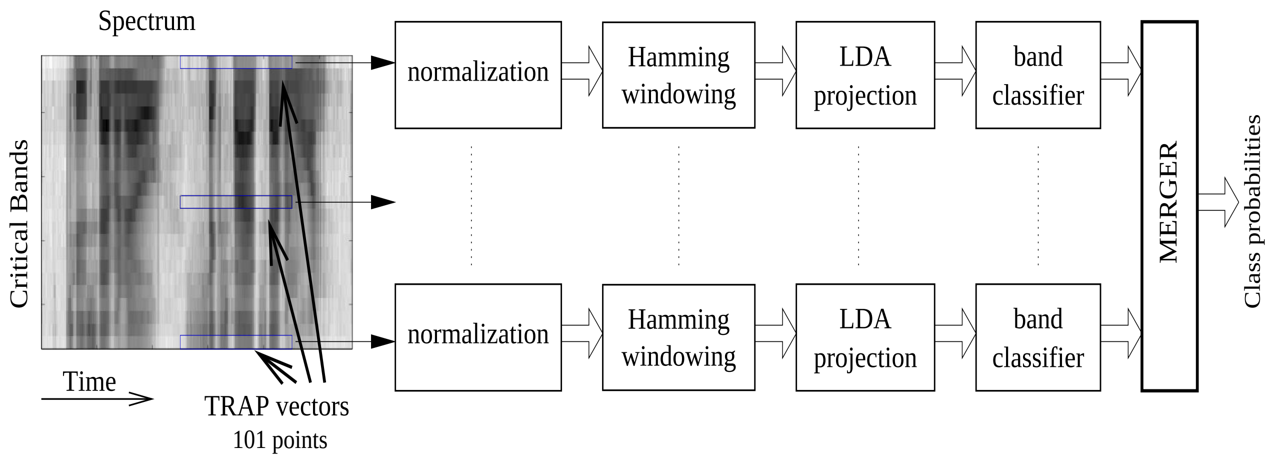

Figure 9.1: Block diagram of LDA-TRAP system

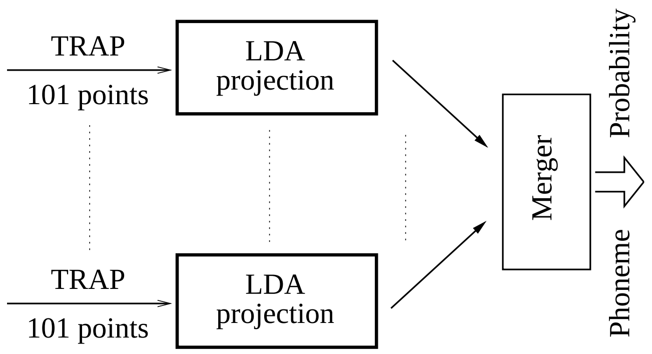

Figure 9.2: Block diagram of Merger only system

TRAP-LDA system. This study also includes different options of TRAP normalization: none, mean normalization (MN), mean and variance normalization (MVN), concluding that while "none" deteriorates the results while using LDA, both MN and MVN with LDA provide better results than their non-LDA counterparts.

An interesting generalization of LDA projection of TRAPs (or TRAPs in pairs of frequency bands, as Grézl also proposes) is a filtering of time-frequency "blocks" using 2-dimensional filters. Kleinschmidt and Gelbart in [24] propose filtering by Gabor filters which can be "tuned" to different rates and different directions in the time-frequency plane. The authors use those features to train a neural network and use the output features in parallel to QIO feature extraction [1]. The comparison is done with the QIO system, or with a completion of QIO by a neural net trained on mel-frequency band energies (so called Tandem system, [12]). The Gabor features outperform both QIO and QIO+Mel_Spec_Tandem on both Aurora-2 (US English) and Aurora-3 (multilingual) tasks.

9.2.3 Reliable phoneme recognizer and M4

TRAP-based approaches are also going to be investigated by the VUT Brno group in the framework of Multi-Modal Meeting Manager project[2] sponsored by the EU. Here, our task is to come up with algorithms for reliable phoneme detection, which should aid several speech-processing tasks:

[2]`http://www.dcs.shef.ac.uk/spandh/projects/m4/`

- large vocabulary continuous speech recognition (LVCSR) for producing meeting transcripts – here, the phonemic information may lead to better recognition of proper names, or become itself the base of LVCSR.
- speaker ID and tracking – statistics of sequences of phonemes should be characteristic for different speakers (some would for example very often say "actually"). We are able to compute those statistics even without knowing the underlying words.
- language and foreign accent ID. Phono-tactic models are quite a common approach to language ID necessitating reliable derivation of phoneme information. For foreign accent ID, it is known, that some speakers would consistently replace a phoneme by a different one (for example the sonorized 'r' for Slavic speakers of English). Doing such detection may aid to adapt the pronunciation dictionary of a speech recognition system.

For solving the task of reliable phoneme detection, it is possible to use directly the posteriors of a NN-based system. Hynek Hermansky has proposed the following steps to post-process the "trails" (by "trail" several estimates of the same phoneme class are meant):

1. give higher confidence estimates from the centers of trails.
2. impose minimum durations on trails (presumably at least 40-50 ms for each phoneme (including the "short" ones) if TRAPS are used for probability estimations.
3. substitute estimates in shorter trails by nearest "reasonable" classes, to make the short trails longer.

The general assumption is that people produce between 10-20 phonemes/sec, more or less uniformly distributed in time.

Another approach was proposed by Petr Schwarz and Pavel Matějka (both currently at OGI):

1. a conventional HMM-based phoneme recognizer is used to generate lattices of candidate phonemes.
2. for each candidate phoneme in the lattice (the lattice determines its starting and end times, eventually spreading across the HMM states), a scoring of *one* temporal trajectory with the center in the center of hypothesized phoneme is performed. The hope is that the nets will work with more or less static patterns, while in the "classical" TRAP approach, they had to cope with many possible shifts of the same temporal trajectory.
3. the obtained likelihoods will be used to re-score the lattices and to find the best path.

So far, we have conducted early experiments on the manually aligned data (TIMIT), training TRAP band-classifiers only on the trajectories in centers of phonemes. Then, on another set, a classification was performed in similar way (only trajectories in centers of phonemes). We have shown, that for this classification, the net trained on centers performs better in terms of phoneme recognition accuracy than the one trained with all possible shifts (even if this later one has "seen" approximately 10 times more training data). We have made a preliminary conclusion, that the temporal synchronization of TRAPs is helping the classifiers and that this approach should be pursued.

Chapter 10

Conclusions

As each of the experimental parts ended by a conclusive chapter (4 and 9), those global conclusions will be quite brief.

We have shown, that the feature extraction is of great importance in nowadays speech recognition systems, and that quite a simple operation, as filtering of temporal trajectories using 101-tap filters, can bring important increase in recognition performance (2% absolute improvement over the baseline of 90% indeed *is* a very important increase). But even more interesting than this is the verification of the basic idea: it is possible to train filters on labeled speech data, those filters resemble to the ones obtained previously by studies of human auditory periphery, and they even improve recognition accuracy! Of course, the experiments conducted are far from complete[1] and it should be for example appended by a thorough study of LDA-filter behavior in different types of noise.

On the other hand, TRAP experiments have not yet shown brilliant improvement over MFCC's, and for some tasks (SPINE) they are well behind. An attempt to excuse ourselves would be stating that researchers needed more than 20 years to come up with today's form of MFCC's with their Δs and $\Delta\Delta$s, and that TRAP efforts started only two-three years ago. Also, there is quite a number of questions in TRAP system design (summarized in section 9.1) ranging from label selection to neural net architecture. We believe however that we are on a very promising way and that temporal trajectories and non-linear classifiers will have a significant role in future speech feature-extraction algorithms.

To conclude this work, I would like to express my great will to continue in the speech processing research and teaching, a field that I consider very interesting, challenging from the scientific point of view but also bringing quite a lot of fun. I hope to be a good tutor to my pre- and post-grad students. I hope to take part in international projects, bringing not only a huge amount of know-how and lots of work, but also visits of nice places and meeting smart and funny people. I sincerely believe that Czech republic will get new possibilities by joining the EC — by the work of myself, and of my group, I would like to make a contribution to the success of our country in the common Europe.

[1]But is a set of experiments ever complete?

Bibliography

[1] A. Adami, L. Burget, S. Dupont, H. Garudadri, F. Grezl, H. Hermansky, P. Jain, S. Kajarekar, N. Morgan, and S. Sivadas. Qualcomm-ICSI-OGI features for ASR. In *Proc. ICSLP 2002*, Denver, Colorado, USA, 2002.

[2] H. Bourlard. Nouveaux paradigmes pour la reconnaissance robuste de la parole. In *Proc. XXIèmes Journées d'Étude sur la Parole*, pages 263–272, Avignon, France, June 1996.

[3] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach.* Number 247 in Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1994.

[4] L. Burget. Concepts of the dissertation. Technical report, Brno University of Technology, Inst. of Radioelectronics, April 2001.

[5] S. B. Davis and P. Mermelstein. Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech & Signal Processing*, 28(4):357–366, 1980.

[6] D. P. W. Ellis and M. J. Reyes Gomez. Investigations into tandem acoustic modeling for the Aurora task. In *Proc. Eurospeech 2001*, Aalborg, Denmark, September 2001.

[7] D. W. P. Ellis, R. Singh, and S. Sivadas. Tandem acoustic modeling in large-vocabulary recognition. In *Proceedings of ICASSP'01*, Salt Lake City, Utah, USA, May 2001.

[8] D. Gibbon, R. Moore, and R. Winski, editors. *EAGLES Handbook on Spoken Language Systems.* Mouton de Gruyter, 1997.

[9] B. Gold and N. Morgan. *Speech and audio signal processing.* John Wiley & Sons, 2000.

[10] F. Grézl. Concepts of the dissertation. Technical report, VUT Brno, Faculty of Information Technology, April 2002.

[11] H. Hermansky. Human speech perception: Some lessons from automatic speech recognition. In V. Matoušek, P. Mautner, P. Mouček, and K. Taušer, editors, *Proc. of 4th International Conference Text, Speech,Dialogue - TSD 2001*, number 2166 in Lecture notes in artificial intelligence, pages 187–196, Železná Ruda, Czech Republic, September 2001. Springer Verlag.

[12] H. Hermansky, D. P. W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional HMM systems. In *Proc. ICASSP 2000*, Turkey, 2000.

[13] H. Hermansky and N. Morgan. RASTA processing of speech. *Trans. on Speech & Audio Processing*, 2(4):578–589, 1994.

[14] H. Hermansky, S. Sharma, and P. Jain. Data-derived nonlinear mapping for feature extraction in HMM. In *Proc. Workshop on automatic speech recognition and understanding*, Keystone, December 1999.

[15] M. J. Hunt. A statistical approach to metrics for word and syllable recognition. *J. Acoust Soc. Am.*, 66(S1)(S35(A)), 1979.

[16] P. Jain and H. Hermansky. Improved mean and variance normalization for robust speech recognition. In *Proc. ICASSP 2001*, Salt Lake City, Utah, USA, May 2001.

[17] P. Jain, H. Hermansky, and B. Kingsbury. Distributed speech recognition using noise-robust MFCC and TRAPS-estimated manner features. In *Proc. ICSLP 2002*, Denver, Colorado, USA, 2002.

[18] J. Jan. *Digital filtering, analysis and restoration of signals (in Czech).* Brno University of Technology, 1997.

[19] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, 1998.

[20] J. Kafka. Linear discriminant analysis in recognition of Czech (in Czech), Diploma thesis. Technical report, Brno University of Technology, Faculty of Electrical Engineering and Communication, June 2002.

[21] S. Kajarekar. *Analysis of Variability in Speech with Applications to Speech and Speaker Recognition.* PhD thesis, OGI School of Science and Engineering, Oregon Health & Science University, Portland, Oregon, July 2002.

[22] M. Karafiát and J. Černocký. Context dependent hidden Markov models in recognition of Czech. In *Proc. Radioelektronika 2002*, pages 37–40, Bratislava, Slovakia, May 2002.

[23] B. Kingsbury, P. Jain, and A. Adami. A hybrid HMM/TRAPS model for robust voice activity detection. In *Proc. ICSLP 2002*, Denver, Colorado, USA, 2002.

[24] M. Kleinschmidt and D. Gelbart. Improving word accuracy with Gabor feature extraction. In *Proc. ICSLP 2002*, Denver, Colorado, USA, 2002.

[25] P. Matějka, P. Schwarz, M. Karafiát, and J. Černocký. Some like it Gaussian .... In P. Sojka, I. Kopeček, and K. Pala, editors, *Proc. of the 5th International Conference on Text, Speech and Dialogue—TSD 2002*, Lecture Notes in Artificial Intelligence LNCS/LNAI 2448, pages 321–324, Brno, Czech Republic, Sep 2002. Springer-Verlag.

[26] P. Motlíček and J. Černocký. All-pole modeling based feature extraction for AURORA3 DSR task. In *submitted to ICASSP 2003*, Hongkong, 2003.

[27] K. H. Muthusamy, R. A. Cole, and B. T. Oshika. The OGI multilanguage telephone speech corpus. Technical report, Center for Spoken Language Understanding, OGI, Portland, Oregon, USA, 1999.

[28] P. Placeway, S. Chen, Maxine Eskenazi, U. Jain, V. Parikh, B. Raj, M. Ravishankar, R. Rosenfeld, K. Seymore, M. Siegler, R. Stern, and E. Thayer. The 1996 HUB-4 SPHINX-3 system. In *Proc. 1997 ARPA Speech Recognition Workshop*, 1997.

[29] L. Rabiner and B. H. Juang. *Fundamentals of speech recognition.* Signal Processing. Prentice Hall, Engelwood Cliffs, NJ, 1993.

[30] P. Schwarz. Continuous speech recognition: spelled letters (in Czech), diploma thesis. Technical report, Brno University of Technology, Faculty of Electrical Engineering and Computer Science, Brno, 2001.

[31] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky. Feature extraction using non-linear transformation for robust speech recognition on the Aurora database. In *Proc. ICASSP 2000*, Turkey, 2000.

[32] S. R. Sharma. *Multi-stream approach to robust speech recognition.* PhD thesis, Oregon Graduate Institute of Science and Technology, October 1999.

[33] R. Singh, M. L. Seltzer, B. Raj, and R. M. Stern. Speech in noisy environments: robust automatic segmentation, feature extraction, and hypothesis combination. In *Proc. ICASSP 2001*, Salt Lake City, Utah, USA, May 2001.

[34] H. van den Heuvel et al. SpeechDat-East: Five multilingual speech databases for voice-operated teleservices completed. In *Proc. EUROSPEECH 2001*, Aalborg, Denmark, September 2001.

[35] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland. *The HTK book.* Entropics Cambridge Research Lab., Cambridge, UK, 1996.