TRAPS in all senses

Report of post-doc research internship in ASP Group, OGI-OHSU, March – September 2001

Jan P. Černocký

September 25, 2001 — version 1.0

Contents

Chapter 1

Introduction

This document covers the TRAP (TempoRAl Patterns) work I have done during the post-doctoral research internship at OGI in the Anthropic Signal Processing Group of prof. Hynek Hermansky.

At the beginning of my stay, I worked on several other problems:

- features inspired by Lyon's cochlea model
- PLP parameterization for the Aurora task
- some methods of post-processing of features prior to the recognizer (pre- and de-emphasizing features using different powers).

None of those directions was truly conclusive and they are not covered in this report. Interested readers are encouraged to study the README files and contact the author.

TRAPs were the core of my work from May till the end of the stay at OGI. The document is organized as follows: chapter 2 gives generalities on TRAPS, comments on neural network (NN) use and contains the description of how the TRAP experiments were evaluated. Chapter 3 describes experiments on the trinity of databases Stories–Numbers–Digits (SND), used in previous work by Sangita and Pratibha. The following chapter 4 describes experiments performed on the new set of databases with good phonetic coverage, defined by Lukáš. Chapter 5 presents running TRAPS on the Spine task. Chapter 6 contains the conclusions and "todo's". Finally, chapter 7 is intended for people wishing to continue the work on TRAPs using my scripts, C programs, and csh, Matlab and Perl scripts.

Acknowledgments

Many thanks to Hynek Hermansky for having invited me to this post-doc and for many fruitful discussions. Thanks to PhD students: Pratibha, Sachin, Sunil, Lukáš and Andre, for lots of theoretical and technical help and also for lots of fun in the lab and elsewhere. Thanks also to Franta and Petr, especially for lots of fun on hiking trips. Last but not least - to Hanka and Tomášek, for having accompanied me to the US and supported me here all the time.

Chapter 2

Generalities

"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) [7]. If noise is present in the speech signal, it affects the entire feature vector, and impairs the accuracy of the recognizer. Multi-stream approach [6] 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 [4, 5, 3, 1] 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).

2.1 TRAP architecture

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 2.1.1. We need to have phonetic labels for the input data.
- generation of TRAPS. Section 2.1.2 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 2.1.3 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 2.1.4.
- merging net putting all the band-classifier outputs together. Section 2.1.5 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 2.1.6.
- HMM recognizer. We worked with two HMM recognizers (Digits built using HTK and Sphinx on SPINE). Those are described in respective chapters: 3 and 5. 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.

2.1.1 The input: Log energies

are in all experiments computed in a very standard way 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.

Precautions must be taken while playing with TRAPs of different lengths than for whose such source-files have been generated. See documentation for the `trapper` program in the cook-book section of this report.

We used pre-generated phonetic labels:

- converted from Stories and Numbers label-files by Sangita for the Stories-Numbers-Digits (SND) experiments.
- converted and re-mapped from Timit and Stories for the reference setup by Lukáš.
- generated by Sunil 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).

2.1.2 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 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 Sangita 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.

As the files with TRAPs are fairly big, and can be generated very quickly from the original data, they are rarely kept. To save the disk space, they are mostly deleted immediately after the NN training or forward pass.

2.1.3 Band classifiers

Band classifiers (also called TRAP classifiers, small nets, first step, band-posterior estimators, or however you want) classify the TRAPs into phonetic classes, or, in some experiments, into broad phonetic categories. Each band classifier is a 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.

[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.

- the output layer whose size is given by the number of classes. The softmax 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" 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.

2.1.4 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.
- priors are merged into p-files for merger training. One pfiles would generally exceed the physical size limit (2.1 Gbyte), so that 2 pfiles are created and recombined by the NN training software.

2.1.5 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 completely removed.

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

2.1.6 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 3 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: $\text{atanh}(2x + 1)$, where x is the softmax output, which produces more Gaussian 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.

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 5 covered those post-processing tricks.

2.2 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 Sangita'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 can we expect at which place of the time trajectory. Some pictures are included at beginnings of SND chapter 3 and reference experiment chapter 4.

2.3 Performance analysis

2.3.1 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 'ffrapalyzer' software.

2.3.2 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:

$$P_l = \frac{N_l}{N} \tag{2.1}$$

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:

$$H_{ij} = \frac{C_{ij}}{N_i} \tag{2.2}$$

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:

$$\mathbf{S}_{i,:} = \frac{\sum_{\forall i} \mathbf{x}}{N_i} \tag{2.3}$$

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:

$$\mathbf{V}_{i,:} = E\{(\mathbf{x}_{\forall i} - \mu_i)^2\} \tag{2.4}$$

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 $\rightarrow 0$). The resulting matrix is therefore very similar to the soft confusion one.

Net 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:

$$\mathbf{C} = E\{(\mathbf{x}^T\mathbf{x} - \mu^T\mu)\} \tag{2.5}$$

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

$$\rho_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}} \tag{2.6}$$

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.

2.3.3 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 3

Basic experiments: Stories–Numbers–Digits

Those experiments were conducted exactly on the same data Pratibha and Sangita used in their work, the results are therefore directly comparable. See conclusions at the end of this chapter 3.12 for the reasons we migrated toward different set of databases (called "reference" ones).

3.1 Data used

3.1.1 Band-classifier training: OGI Stories

this database is described in Sangita's thesis [6], 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).

3.1.2 Merger training: OGI Numbers

this database is also described in Sangita's thesis [6], 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 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.

3.1.3 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.

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.

3.1.4 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`

		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 3.1: Phoneme coverage in Stories and Numbers

`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 3.1. We can see that especially the silence is heavily over-represented. Therefore, some experiments with balanced training were conducted.

3.2 HMM recognizer

was common to all experiments. Pratibha'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 as optimal by Pratibha).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 Pratibha's TRAP baseline accuracy of **93.15** served as comparison numbers.

3.3 The baseline: `base_scripts`

Section titles contain names of directories for easier orientation in directory structure. The data directory name was `base` here. Beginning with the following experiment, the name of script directory was kept coherent with the data directory. See cook-book section 7.1 to find out more on directory names.

Why: to reproduce Pratibha'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.
- and finally 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 Pratibha, 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).

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 Pratibha's number 93.15%, but worse than MFCC baseline 94.07%.
Conclusions: Pratibha's experiment was successfully reproduced including a little patch. Everything runs faster. Good basis for following experiments.

3.3.1 Baseline – visualization

Mean TRAPs were generated on Stories (5th band) and are shown in Fig. 3.1. They correspond to Sangita's results in [6]. In addition, variance-TRAPs were generated (Fig 3.2) showing the variability of the 101 points for each label.

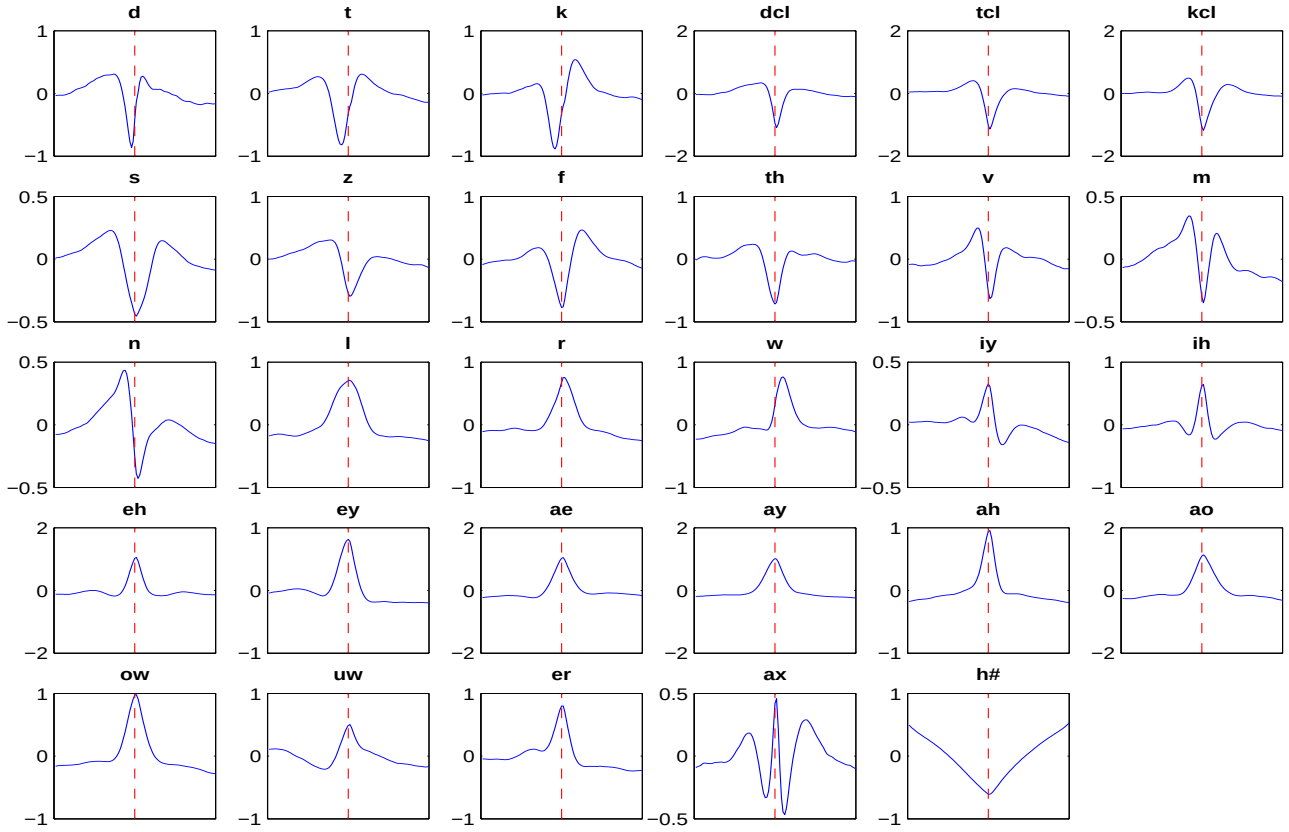

Figure 3.1: Mean TRAPs on Stories

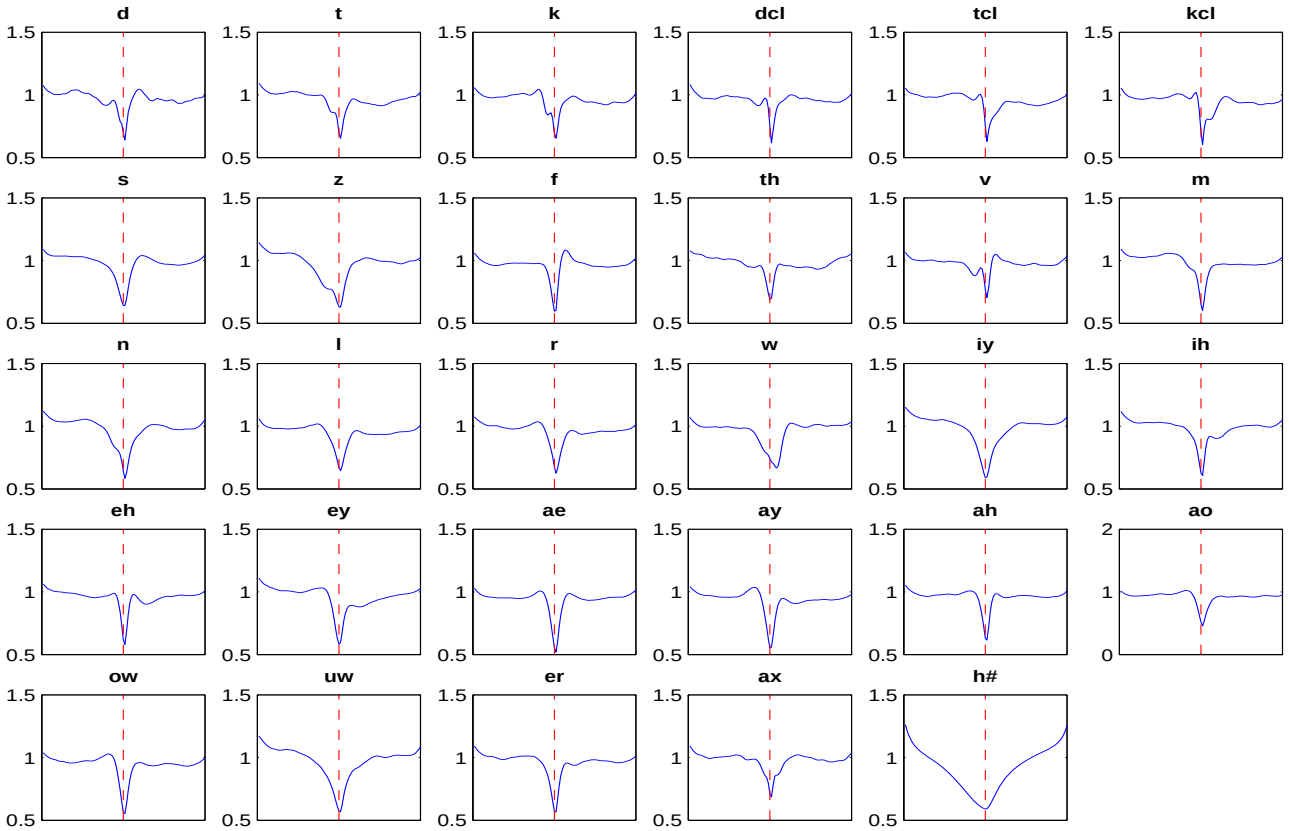

Figure 3.2: Variance (or better standard deviation) TRAPs on Stories

3.4 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 (pilsner)), SPERT still outperforms Pentia for bigger nets (merger): compare 9.50 hours (10 epochs) for SPERT and 11.50 hours (just 9 epochs) for Linux.

3.5 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 4) and SPINE experiments (chapter 5), we returned to the original TRAP-normalization.

3.6 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 3.1.4). 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.

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.

3.7 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.

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

3.8 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 ⇒ t
dcl+d ⇒ d
kcl+k ⇒ 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 3.2), 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.

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

Why: The statistics (section 3.1.4) 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.

3.10 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.

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

3.12 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]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.

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 4

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'.

4.1 Data, phonemes and evaluation

Lukáš 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 part #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 transition 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 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.

4.1.1 Phoneme set

The unique phoneme set was defined by Lukáš as common for Stories and TIMIT (some phonemes had to be mapped). Table 4.1 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 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.

4.1.2 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.

4.2 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.
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 4.1 and 3.1 (mean TRAPs) and 4.2 and 3.2 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 2.3.2) 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 4.3.

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%). The differences in recognition accuracy of the other phonemes would need further analysis.

		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 4.1: Phoneme coverage in TNN-Timit and MNN-Stories

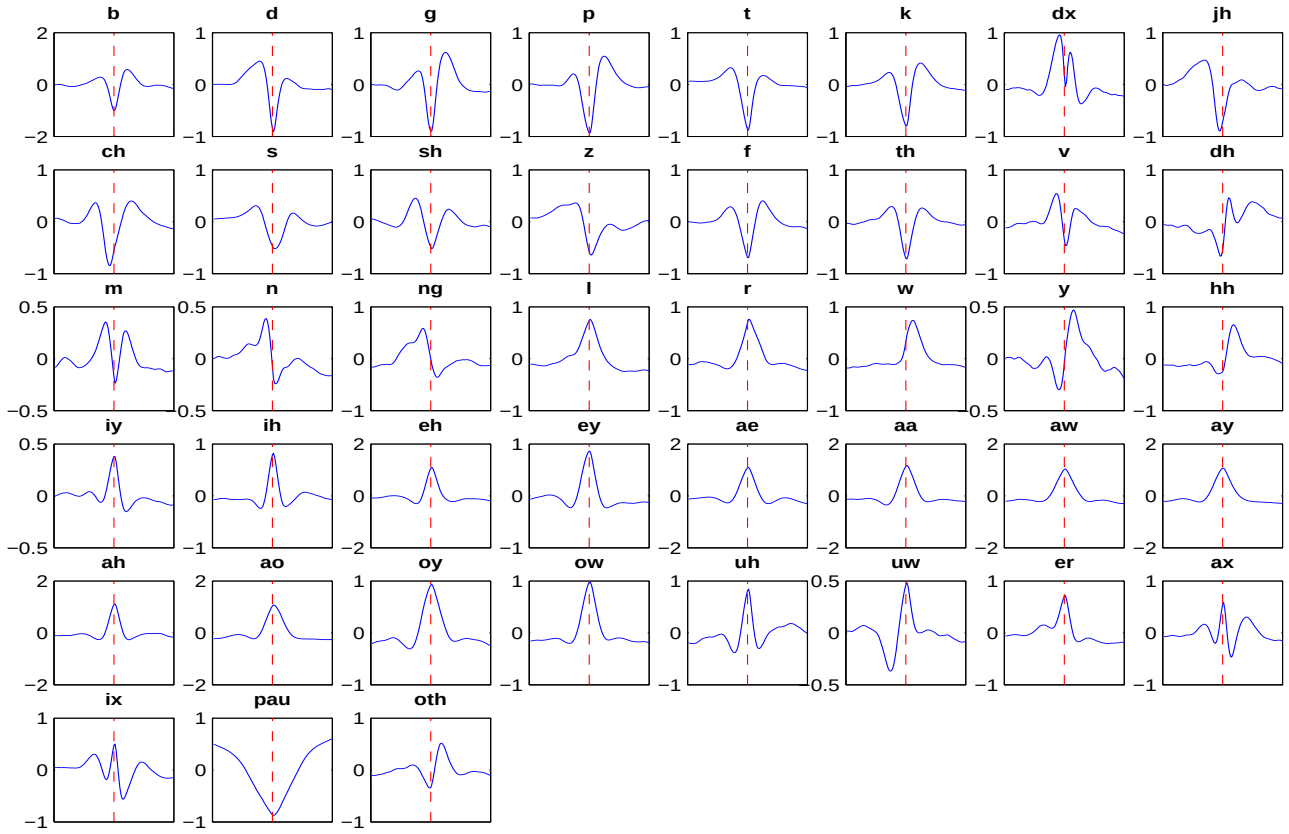

Figure 4.1: Mean TRAPs on TNN-Timit

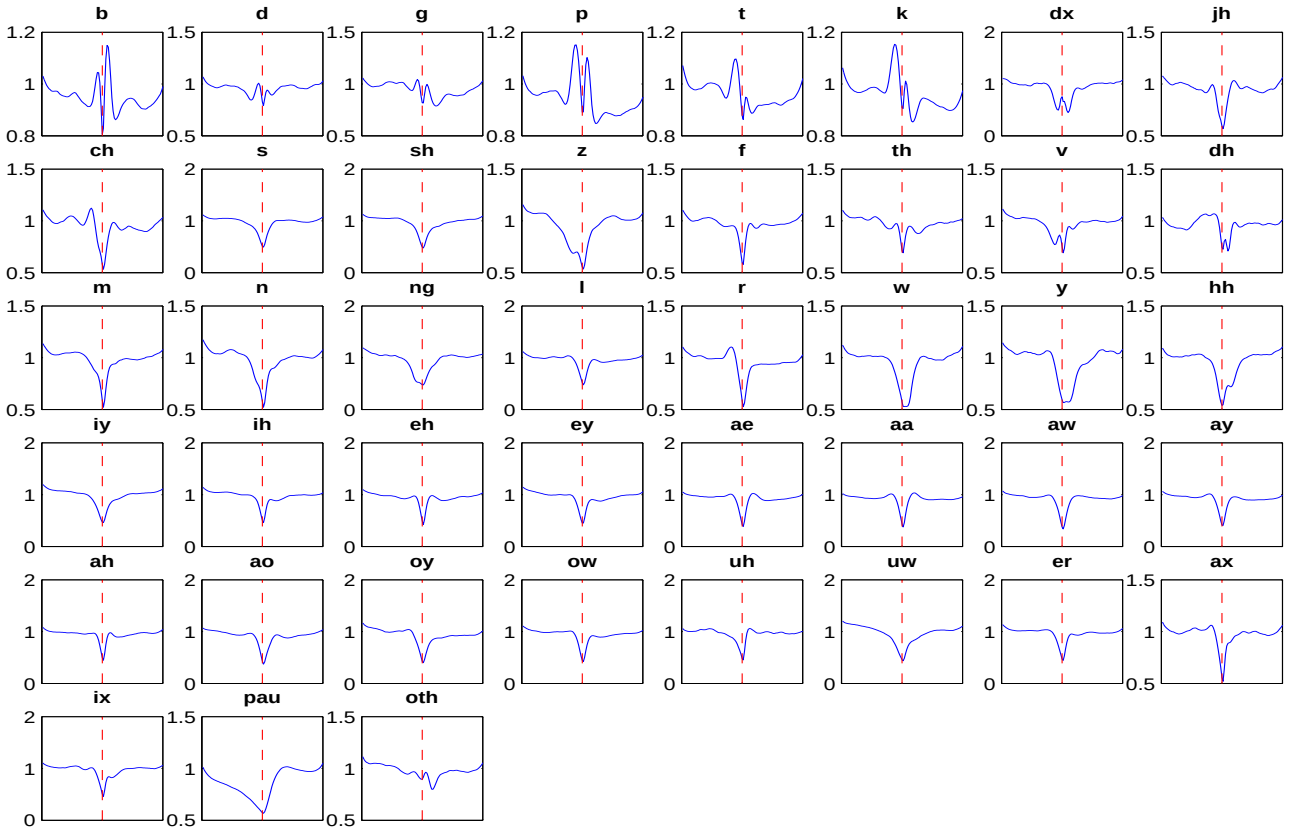

Figure 4.2: Variance (or better standard deviation) TRAPs on TNN-Timit

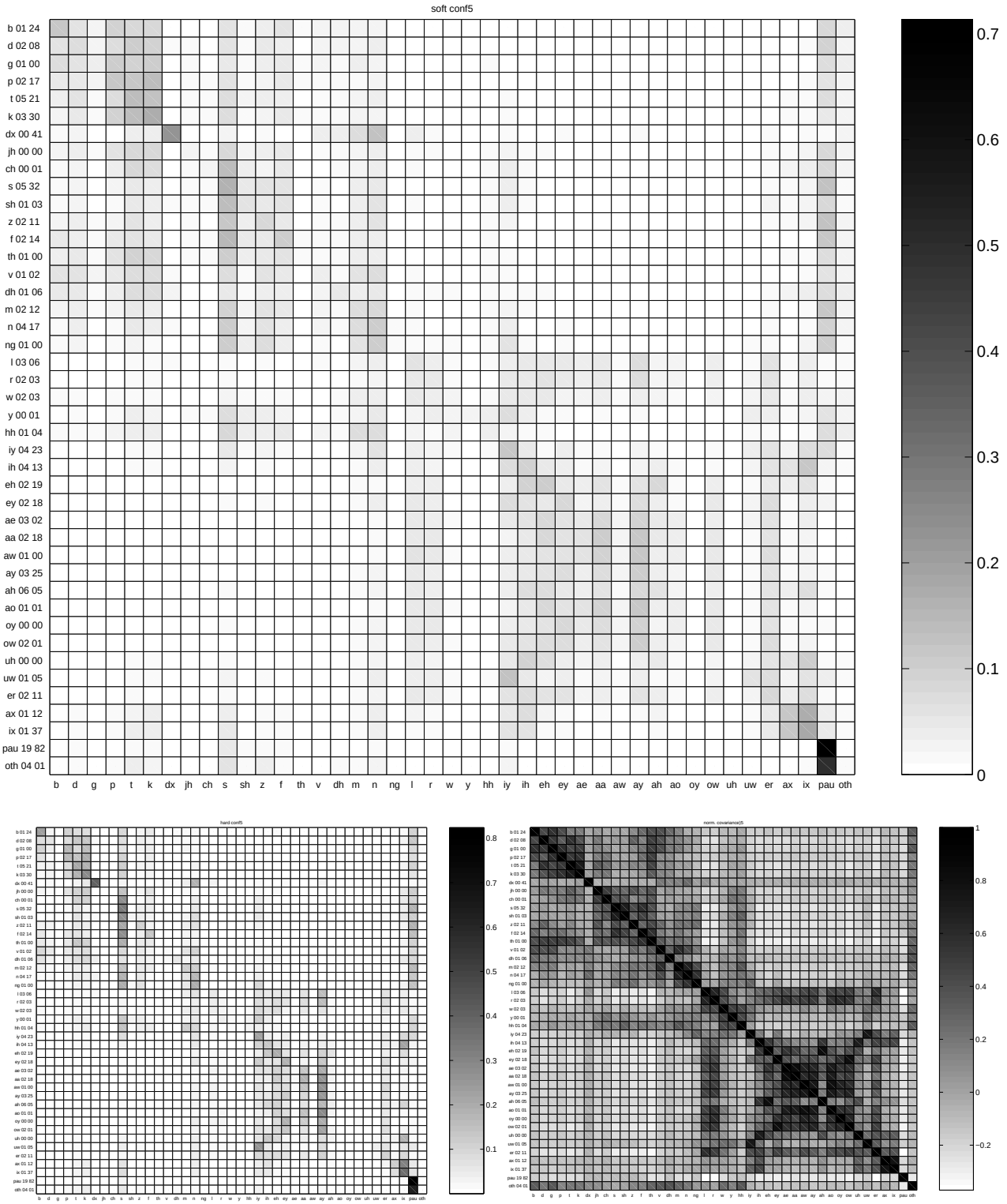

Figure 4.3: Soft, hard and normalized covariance matrix of band #5 (reference TRAPs–baseline experiment).

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 4.3, 4.4 and 4.5.

Similar matrices were generated also at the output of the merger – see Figure 4.4.

4.3 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 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 Sachin'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.

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 3.7, 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.

4.4 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:

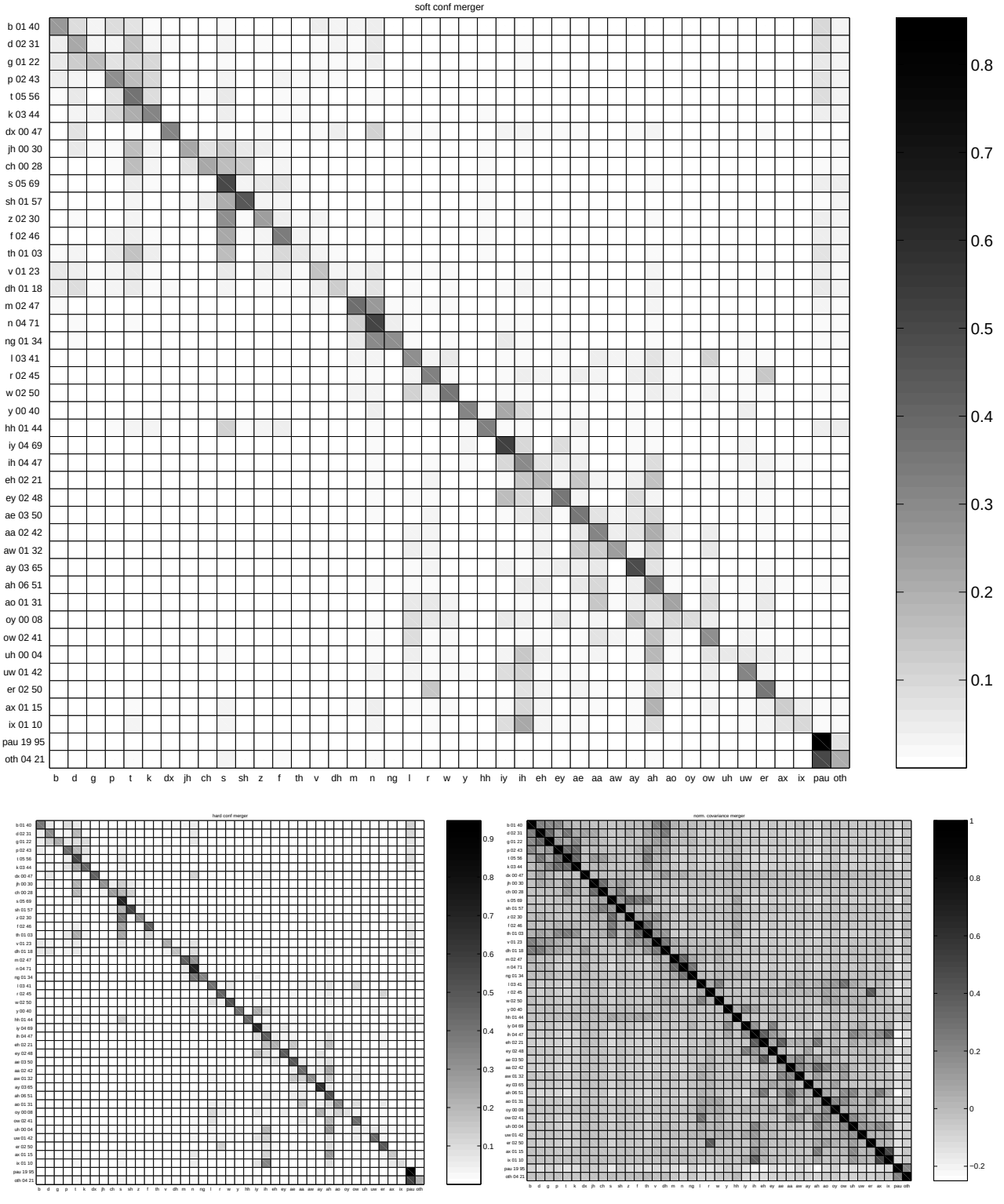

Figure 4.4: Soft, hard and normalized covariance matrix at the merger output (reference TRAPs–baseline experiment).

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 4.5. 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.

4.5 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:

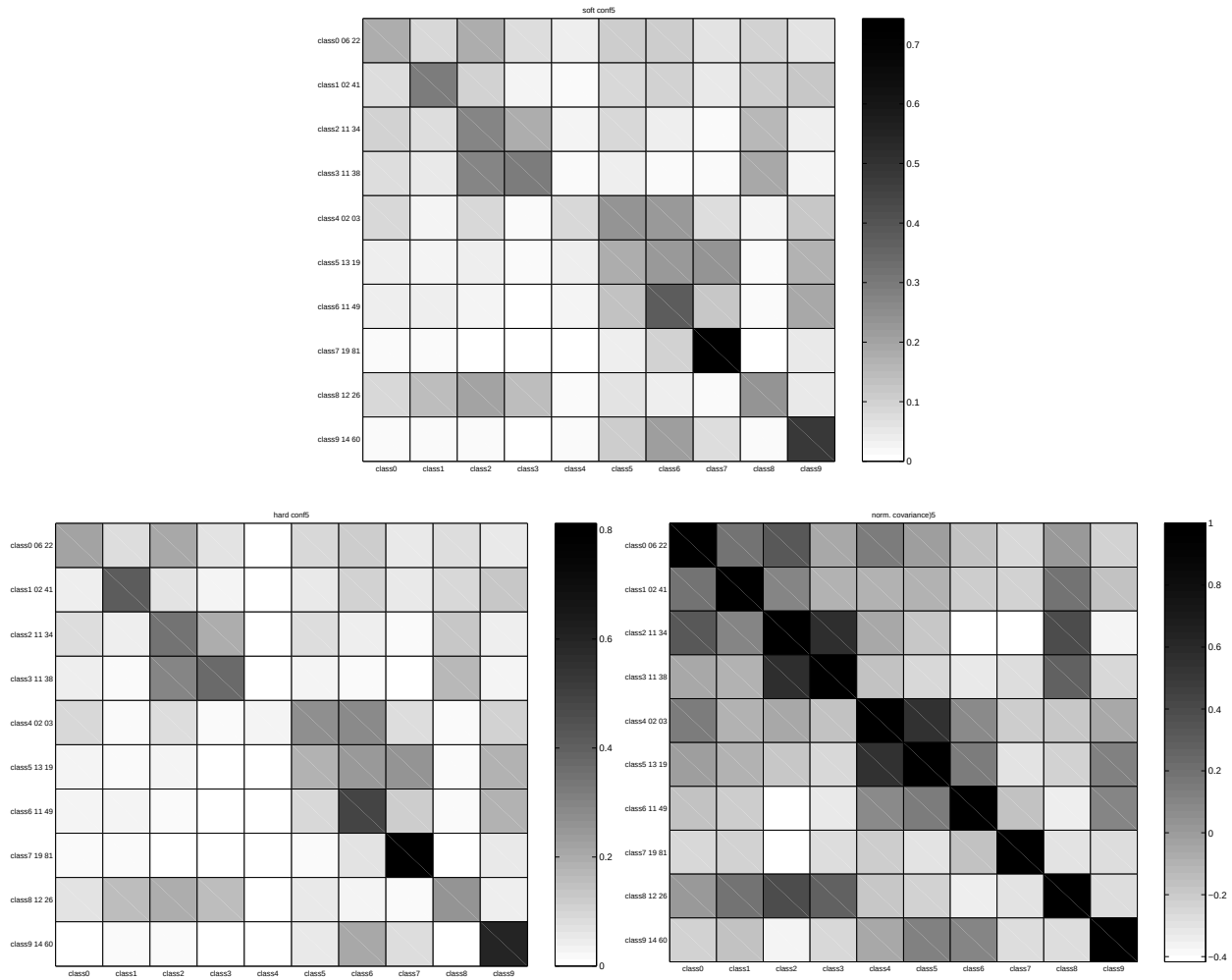

Figure 4.5: Soft, hard and normalized covariance matrix of band #5 (reference TRAPs–10 automatically generated classes).

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.

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

Why: quite ugly confusion patterns of the 'other' class (Fig 4.3) lead us to investigate, what the 'other' label really was. A discussion with Lukáš made it clear, that this label should not be considered one class but rather discarded (see section 4.1.1 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.
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...

4.7 2-Band TRAPs – adjacent bands `2_band_51_300_300`

Why: Pratibha 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%.

4.8 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.

4.9 2-Band TRAPs – a bit of brute force: `2_band_1_skip_101_500_1000_no_oth`

Why: This 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.

4.10 Reference TRAPs – Conclusions

TRAP experiments were conducted on a set of databases providing, in our opinion, more reliable 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, or whatever appropriate) 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.

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 5

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 at CMU into speech and silence regions. The recognizer used – SPHINX – came also from CMU.

5.1 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 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 (Sachin), the complete-silence parts detected (Petr) and finally, new signal files were generated (Honza). 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

5.2 Labels and training sets

The labels were produced last year at ICSI [2], 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.

5.3 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 5.1 presents the two mappings.

Statistics computed in the train_a portion of the training set (improved data) in Table 5.2 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.

5.4 Experiments on original data

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

Only selected experiments are described in this report, see file:
`file:/u/honza/OGI/SPINE/results_html/spine2000.html`
for the description and results of all (sometimes quite crazy) experiments. The numbers in names of section correspond to experiment numbers in this table.

5.4.1 The MFCC baseline: `htk_mfcc13_0_d_a_z` – 1-5

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 o 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) 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.

5.4.2 TRAPs from Stories and Numbers `traps_merger_on_numbers_1b_101_nn_300_300` – 10

Why: first experiment with TRAPs on spine, all nets were taken from SND experiment `qmk_no_hamming_sent_norm` (see section 3.6.
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:

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 5.1: Phoneme mapping 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 5.2: Phoneme coverage in SPINE: original ICSI 56 phonemes and re-mapped small set of 34 phonemes.

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.

5.4.3 Merger trained on SPINE
`traps_merger_on_spine_1b_101_nn_300_300` – **9**

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 3.6).
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).

5.4.4 Merger trained on SPINE - small set of 34 labels
`traps_merger_on_34_spine_1b_101_nn_300_300` – **11**

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 3.6).
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.

5.4.5 Everything trained on SPINE - small set of 34 labels
`traps_all_on_34_spine_1b_101_nn_300_300` – **12**

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.

5.5 Experiments on "improved" data

As mentioned before, the army files were corrected and used in further experiments.

5.5.1 MFCC baseline `idata_htk_mfcc13_0_d_a_z` – 19

Why: first test with improved data using classical MFCC features. Same MFCC computation as in experiment `htk_mfcc13_0_d_a_z`, section 5.4.1.
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.

5.5.2 TRAPs from Stories and Numbers `traps_idata_baseline -- 20`

Why: to test the same approach as in the TRAP baseline for original data: take nets from Stories and Numbers and just forward-pass 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 3.3.
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:

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.

5.5.3 Nets from Reference experiments `traps_idata_tnn_mnn` – 21

Why: section 3.12 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 4.2.
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:

[1] if the previous file did not 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
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). we have seen a small, but noticeable

5.5.4 Band-nets from Reference experiments, merger on SPINE `traps_idata_tnn_merger_on_34_spine` – 25

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 4.2.
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 provides 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.

5.6 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 (remember the merger has an output softmax-nonlinearity) followed by a PCA). It was inspired by Sunil's works on the post-processing of feature-net and by the article of Dan Ellis [1] 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 those experiments.

5.6.1 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 5.1.

After the log (actually minus log, fig 5.2), 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 5.3), 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 5.3. Resulting histograms shown in figure 5.2 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. .

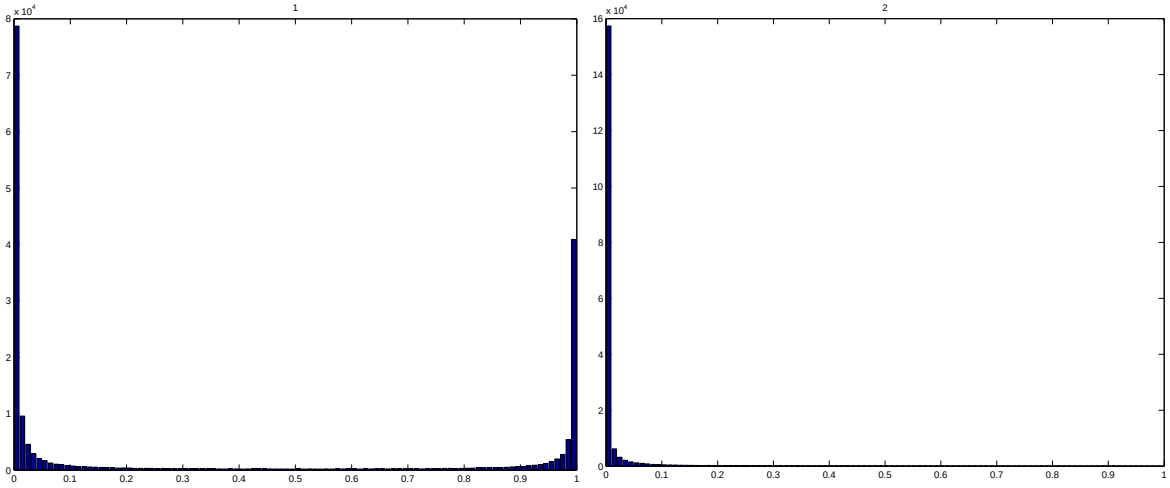

Figure 5.1: Histograms for softmax output of the merger: 'sil' and 'aa' outputs

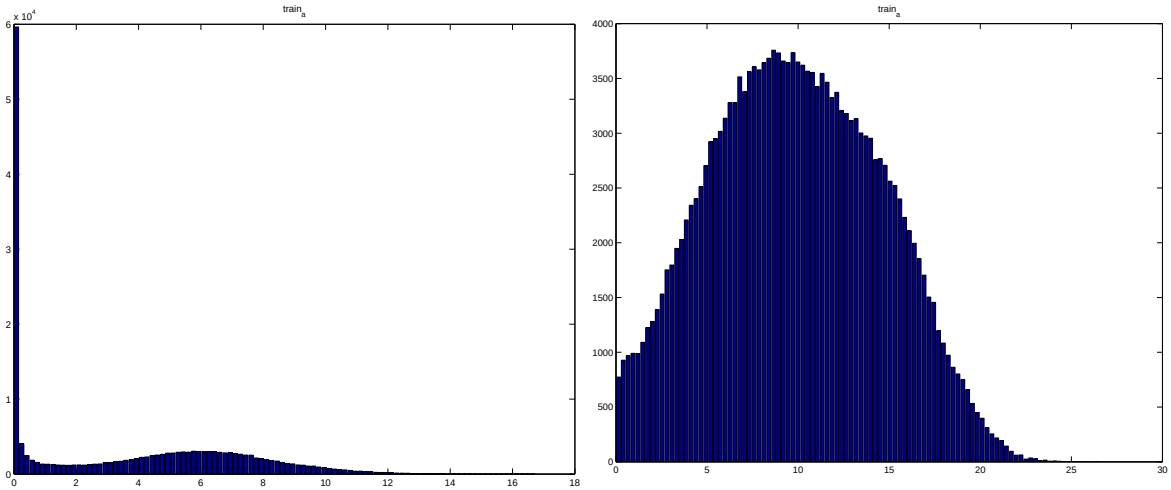

Figure 5.2: Histograms for log of softmax output of the merger: 'sil' and 'aa' outputs

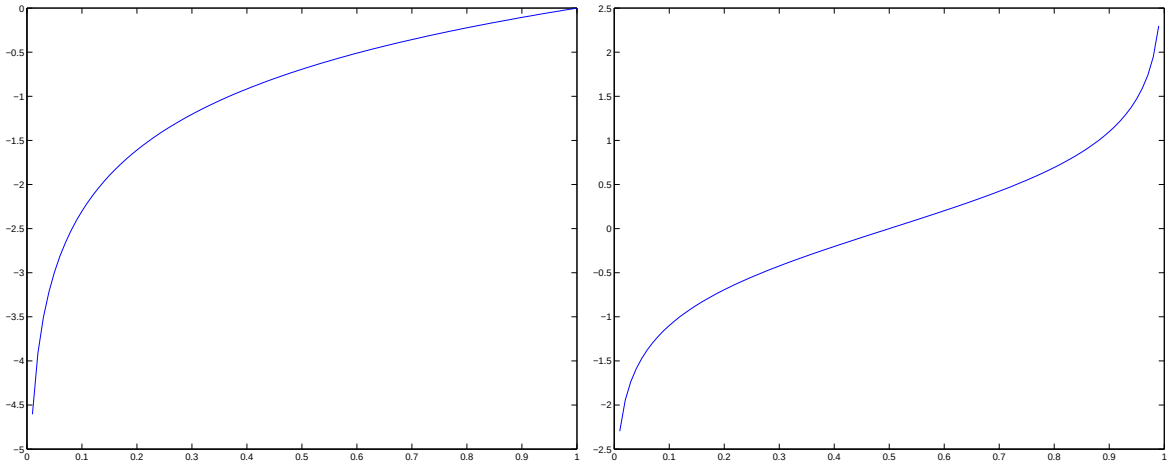

Figure 5.3: Log and atanh functions to post-process the merger softmax outputs

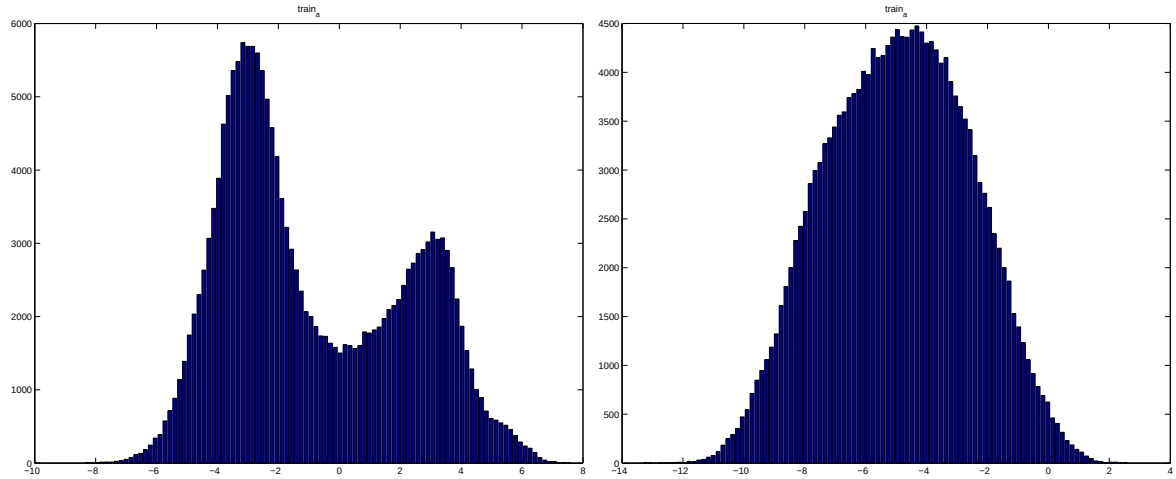

Figure 5.4: Histograms for atanh of softmax output of the merger: 'sil' and 'aa' outputs

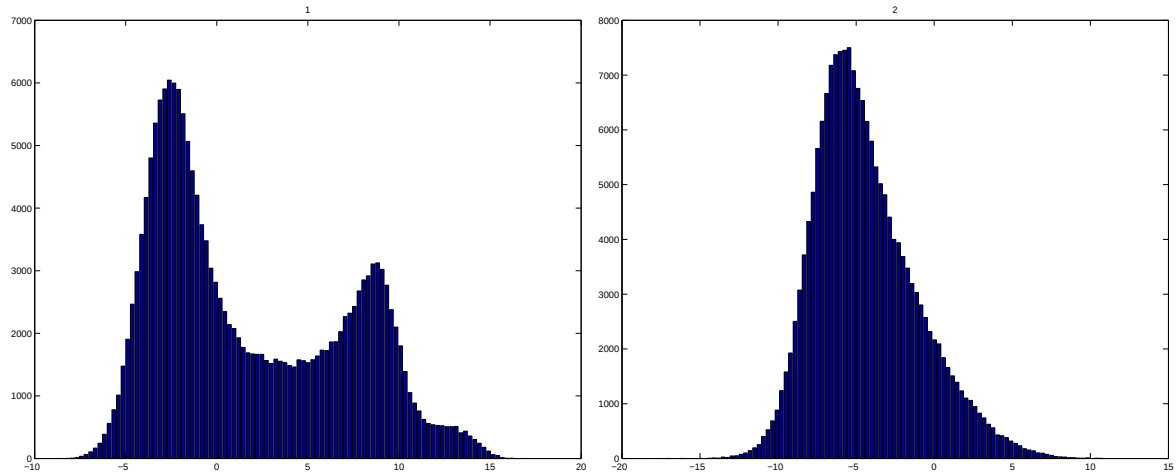

Figure 5.5: Histograms for linear output of the merger: 'sil' and 'aa' outputs

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 5.5 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 probabilities (Sunil), atanh is the choice.

5.6.2 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:

$$\rho_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}} \tag{5.1}$$

   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 followed by mean norm.	68.4	**42.7**
`traps_itmo34s_fwdlin_d_dd_pca34_mvn`	PCA followed by mean and var. norm.	68.3	43.7
`traps_itmo34s_fwdlin_d_dd_mn_pca34`	mean norm. followed by 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.

5.7 Brute force on SPINE

As the last cry we have done and experiment summarizing all the good things 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, and good Gods of speech recognition would be favorable (and forgivable) to us.

5.8 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 Pratibha show, that especially for 2-band TRAPs, this should be a very promising way.
- SPHINX recognizer was found to give worse performance that the system of Yonghong Yan and Chaojun Liu. Lukáš is working on bringing the TRAPs to work with this system.

Chapter 6

Grand conclusions

- Although outperforming 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 *labels* seems to play crucial rôle in the TRAP work. In sections 5.4.3 and 5.4.4, 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 nets 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, but only in the HTML table), 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 decorrelate 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.

Chapter 7

The cook-book

This chapter gives some instructions for the people wanting to make use of my scripts, C-programs, etc. Comments are welcome.

7.1 Directory structure

I always keep a separate directory for scripts, programs, README's and additional (not too voluminous) info, and for the data. I got used to it in Brno, where we do not have enough capacity to backup the data disks. If you backup the scripts, you can always re-generate the data, even if you loose them...

- SND experiments script directory: `/u/honza/OGI/TRAPS`. The experiment subdirs DO NOT begin by 'traps' (this is a bit confusing...). The directory with recognition scripts for the Digit database is `/u/honza/OGI/RECO/exps`. Data directory is `/net/pilsner/u0/honza/TRAPS`. In the data directory, subdirs `on_stories`, `on_numbers`, and `on_digits` denote the database run. Experiment subdir names are coherent with those in the script directory.
- Reference TRAPs script directory is `/u/honza/OGI/REF_TRAPS`. Data directory is `/net/cernahora/u0/honza/REF_TRAPS`. In the data directory, subdirs `src`, `tnn` and `mnn` stand for source files (log energies, labels, etc), band-classifier training and merger training.
- SPINE is more tricky. The TRAP generation script directory is also `/u/honza/OGI/TRAPS`. Subdirs for SPINE experiment DO BEGIN with 'traps'. The directory containing SPHINX-related stuff is `/u/honza/OGI/SPINE`. The data directory is `/net/pilsner/u0/honza/SPINE/SPHINX/Experiments` with some links from `/net/cernahora`.

In the data directories the following structure can be found:

- `PFILES` - as the name says, for storing PFILES.
- `FFRapout` - phoneme (or broad classes) posterior files, ICSI ff.rap-format.
- `Log` - logs of the net training and forward passes.
- `Norms` - files to normalize features before NN training or forward pass. Computed prior to training.
- `Weights` - result of NN training.
- additional directories if needed.

After an experiment is finished and you are sure you will not need any pfiles or ff-rapfiles anymore, everything in `PFILES` and `FFRapout` can be deleted. It is however wise to keep logs, norms and weights (they do not take too much disk-space).

In SND experiments, you will find those additional directories under `on_digits`:

- `train,test` - carrying HTK-features for the HMM recognizer.
- `models,models.*` - HMM parameters for the initialization and context-free Baum-Welch re-estimation (HInit, HRest).
- `hemodels.*` - HMM parameters and context Baum-Welch re-estimation (HERest).

The recognition results directory is kept in some temporary directory (`/tmp/something`) and only the result of scoring is stored here (files `score-*`). After an experiment is finished, all the features and models can be deleted, as it is quite quick to re-produce them (less than 1 hour). Only the score file should be kept.

In SPINE experiments, these following subdirs appear in data directories:

- `Feat` - features for SPHINX in CMU format.
- `model_parameters`
- `model_architecture`
- `bwaccumdir`
- `trees`
- `logdir` - the log files of SPHINX are VERY BIG. It is wise to delete them once the experiment is finished.
- `Results` - recognition results and results of scoring. Keep this directory.
- `c_scripts` - scripts for running SPHINX. Keep this directory.

7.2 Environment variables

A bit of relief ... nothing needs to be set to run my scripts. Some paths need to be set in order to run software on SPERT boards. Look into `/u/honza/.cshrc`.

7.3 README's

Are the FIRST thing you should look at when you want to re-produce an experiment. As my memory does not work very well (and never ever did), I use to put everything I find useful into README's (some call it grapho-mania):

- introductions to experiments, summarizing the information, file locations, scripts, etc. someone other did and I used.
- a few lines of motivation for each experiment.
- names of directories for each experiment (very important!).
- everything you need to run the experiment, mostly in the form: `nice +20 script.csh >& /tmp/script.log &` Those lines can just be copied to a terminal window by copy-paste and the script is fired.
- results and comments.

The README's are sometimes quite messy[1], so use of a key-word search is recommended when you want to find something. The 'grep' command is also quite handy.

Location of the main README's (there are more of them, just look at the names or use grep if you think they may contain something interesting):

- for SND experiments: `/u/honza/OGI/TRAPS/README_TRAPS`
- for reference TRAPs: `/u/honza/OGI/REF_TRAPS/README_REF_TRAPS`
- for TRAPs on SPINE: `/u/honza/OGI/TRAPS/README_SPINE_TRAPS` describes the feature generation. For running SPHINX recognizer (including the results), look at `/u/honza/OGI/SPINE/README_SPINE`.

As the running of experiments is documented in README's, it is not covered in this report.

7.4 Notes on compiling C-programs

- In programs I have written, I use some functions to ease me the life (file opening, allocation, some variable printing). There exist surely professional libraries to do all that, but I am quite happy with my little functions. Whenever you see `#include "common.h"` in the source-code, you should copy or link files `common.[ho]` from `/u/honza/C_PROGS` to the program directory or compile `/u/honza/C_PROGS/common.c`.

[1] and worse ... they may contain very explicit vulgar words if something did not work...

- if there is a Makefile in the dir, the safest way to compile is `make target_executable`. It will not mess-up the other programs in the directory.
- if there is no Makefile, mostly the compiler command is given in a comment somewhere at the beginning of the source. If not, use your common sense, I am not using any fancy stuff in C :-)

7.5 Notes on scripts

If you want to design an experiment, those are the criteria to take to select the right directory to copy the scripts from and modify:

1. select the experiment which is closer to your one. The least changes you make, the less probability of introducing new bugs.
2. in case there are several directories with similar experiments, choose the most recent one. Hopefully the scripts will be cleaned from rubbish and even more hopefully they will be optimized to save disk space, run faster, be more funny, etc.

7.6 Trapper

is a program to generate TRAP's. It is located in `/u/honza/OGI/TRAPS/C/trapper`. This section is actually a copy of the README you can find there. Trapper allows for much faster and more flexible creation of pfiles with TRAPs, than in the original work of Pratibha and Sangita. It is placed in a pipeline between pfile_print and pfile_create. See TESTING for many examples of trapper's use.

7.6.1 Input:

-i filename or '-'
pfile with features per band and somehow flipped or extended beginning and end of each sentence. Usually we flip first 50 and last 50 frames (for 101-point traps) or take those 50 frames from the previous and next file. '-' for std input (default).

-b 0 for ascii, 1 (default) for binary native input.
As it is difficult to read directly pfile (c++), will use BINARY output from pfile print (big endian - this is in the doc ! verified it uses native input !). Our program will get only the features for bands we want + sentence and frame numbers + labels. Will give also option to work w/ ascii input, as the old progs.

-s filename
optional sentence and frame number patching file (needed for training on SPERT which does not like too long sentences.). Ascii. one pair of sentence and frames nos. per line. if nothing, no patching.

-I filename
optional input label patching file - patching directly the input labels, not after trap selection. Ascii. one label per line. if nothing, no patching.

-p filename
optional label patching file - trap level. ascii. One label per line. Providing also a possibility to say that we want to zero all labels (good just for fwd pass with a network alien to out labels, or when we do not have any labels at all) - give filename of patching file '0' if you want that.

-d filename
optional down-sampling file which is handy when we want to balance the set for NN training. For each label, a float down-sampling coefficient must be provided. The least represented class should have down-sampling coefficient of 1.0. If you want to know, how is fractional down-sampling done, look in the code. This file can be also used for blacklisting certain labels (for example silence or noise - give a label down-sampling factor 0 if you do not want it at all).

-M filename
Matrix filename if traps are to be post-processed by LDA or PCA. Default NULL (no post-multiplication). Rows of the matrix have to contain base vectors. The number of bases to retain is controlled using the -y switch. The matrix-file can contain more bases, but those $\geq$y are not used.

!!! The steps are done in the following order (they will rarely be used all together, but who knows:

original label patching → limitation to valid labels → label patching → down-sampling → sentence and frame number patching → matrix multiplication.

7.6.2 Configuration

-P number of bands
yes, trapper can generate multi-band traps. Default 1. If P>1, the normalization is always done independently in each band, and the traps from bands are concatenated.

-V number of valid labels.
This is an OBLIGATORY parameter (actually the only one :-). All frames with label $\geq V$ will be used as context, but traps will not be generated for them !

Example: phoneme set has 29 labels, 'sil' is the last (28). If you want to generate traps also for silence, put -V 29. If not, put -V 28 and silence frames will be used only as context. Similar effect can be obtained by using a blacklisting file with 0.0 factor for the labels we do not want.

-x number of valid patch labels
If a trap-level label patch file is used (-p), you must specify the size of its label set by -x. Example: original label set has 43 phonemes, we want to patch it to 10 broad phonetic classes. -V 43 -p patchfile -x 10

-h 0 for no Hamming, 1 for Hamming
optional Hamming windowing of all input bands. Hamm. windowing comes after all normalizations. Default 1.

-m 0 for no mean norm, 1 for mean norm
optional mean normalization. Default 1.

-v 0 for no var norm, 1 for var norm
optional variance normalization. Var norm can not be done w/o mean norm. Default 1.

-S 0 for sentence-based norm off, 1 for sentence-based norm on.
Mean and var are computed over the whole sentence (only over the 'correct') frames from LeftSkip + LeftContext to end - RightSkip - RightContext. then ALL frames are normalized. Each band is always treated separately. Default 0.

-T 0 for trap-based norm off, 1 for trap-based norm on.
Mean and var are computed for each TRAP, then the norm is done on this trap. Each band is always treated separately. Default 1.

!!! You can specify -S 1 -T 1, but it is the same as -S 0 -T 1. Just think about a local normalization following a global one ...

-L left context
important parameter telling how many frames from the left context are we going to take into the trap. Default 50.

-R right context
important parameter telling how many frames from the right context are we going to take into the trap. Default 50. You see, that the default are 101-point symmetrical traps.

-l LeftSkip

-r RightSkip
optional manual selection of beginning and end point for each utterance. This is handy if we have generated the input pfile with a context of 50 frames on the left and 50 on the right of each sentence, but want to experiment with shorter traps. Default 0,0.

Example: the pfile was generated to be used with 101-point symmetrical traps, so that there are 50 extra vectors at the beg and 50 extra vectors at the end. Suppose you want to experiment with 71-pt asymmetrical traps with left context of 40 and right context of 30. You should not use the first 10 vectors and last 20 vectors of each sentence. Use the following parameters:
-L 40 -R 30 -l 10 -r 20

-y number of valid bases
Number of vectors you want to retain if PCA or LDA is applied. Obligatory if -M used.

7.6.3 Output

-o filename or '-'
output file or stdout. Default '-'.

-B 0 for ascii, 1 for binary Big Endian output.
Default 1. The bin. output has to be Big Endian, as pfile_create wants it so.

7.6.4 Diagnostic output (stderr)

diagnosis and some stats.

7.7 Trapalyzer and related Matlab scripts

as the name suggests, it is intended for analyzing TRAPs. Located in `/u/honza/OGI/TRAPS/C/trapalyzer`. Computes the means and vars of traps, and is able to output some sample traps so that we see, what do we want to classify. It works on the text output of a different program for creation of traps (which is usually piped into the pfile_create). The current version can not work with binary input, so if used with trapper, you should not forget the -B 0 switch. Trapalyzer can make itself the traps, as would do Quicknet with the window_extent switch.

7.7.1 Input

-i input text file (can be - for stdin (default))
The text file should contain data in the format accepted by pfile_create, that is
`sent_number frame_number ...trap... label`

7.7.2 Configuration

-P size of input
length of input traps with explicit context, 1 when the context is should be created by trapalyzer. Default 101.

-L left context
0 if explicit context, ¿0 if context is to be created. Default 0.

-R right context
0 if explicit context, ¿0 if context is to be created. Default 0.

-V number of valid labels
traps with labels $\geq$ this number will be discarded. Default 29.

-n number of examples.
The program allows to select a couple of examples of TRAPs and store them in a text file. Good for visualization. If 0, no examples are written. Default 10.

-k step in taking examples
each k-th example taken. Default 1.

7.7.3 Output

-e prefix of filename with examples number of label will be appended. In examples, the direction of P is written first, then the context. One example per line. Default `/tmp/traps/ex`

-m filename where means and stds will be written.
In means and stds, the direction of P is written first, then the context. Default `/tmp/traps/mv`. The file will contain:

- global mean trap
- mean traps per class
- global std trap
- std traps per class.

-s filename where stats will be written
number of traps per label and percentage. Default '-' (stdout).

7.7.4 Visualization of trapalyzer output

In the program directory, the Matlab script `see_traps_1_pic.m` would plot the mean traps, std-traps and frequency responses of traps to three plots. It can also visualize some examples. A better version of this script is in `/u/honza/OGI/REF_TRAPS/tools/see_traps_1_pic_43.m`. To those scripts, it is necessary to provide a file with the phoneme set, so that the script knows which labels to put on the top of plots.

7.8 FFrapalyzer and related Matlab scripts

FFrapalyzer is intended to analyze posterior probability files (ff.rap). It resides in `/u/honza/OGI/TRAPS/C/ffrapalyzer`. Originally, the analysis of ffrap-files was done in Matlab, but the reading of ffrap-files is too slow there. FFrapalyzer produces matrices and stats in text-form, which can be then visualized by some Matlab scripts.

FFrapalyuzer needs the posterior probability vectors and "true" labels and it provides the following information (theoretically explained in section 2.3.2):

- hard confusion matrix.
- soft confusion matrix.
- output variance matrix per reference class.
- covariance matrix of outputs.
- statistics of coverage of classes, and statistics of correctly recognized vectors (which can be compared to Quicknet forward pass log-files – the global accuracy should be the same as shown in the log-file).

7.8.1 Input

-i input rap file.

-l input label file
text, one label per line.

7.8.2 Configuration

-n size of label set in the label file
can be different from the rap-file for non-square confusion matrices.

7.8.3 Outputs

-H filename for hard confusion matrix.

-S filename for soft confusion matrix.

-V filename for variance matrix.

-C filename for covariance matrix
as explained in the theoretical part, does not look at the reference labels.

-s filename where stats will be written.

7.8.4 Running ffrapalyzer and visualization of its output

FFrapalyzer was used mainly to analyze data in reference experiments. Many of the script directories under `/u/honza/OGI/REF_TRAPS` contain:

- `conf_matrices_mnn_bands.csh` script for computing the matrices for all bands. Outputs are stored to `/net/cernahora/u0/honza/REF_TRAPS/mnn/experiment/Confusion`.
- `show_conf_matrices_mnn_bands.m` visualizes the matrices.

- `conf_matrice_mnn_merger.csh` computes the matrices for the merger output.
- `show_conf_matrice_merger.m` visualizes the matrices.

It should be fairly simple to modify those scripts to your needs.

7.9 Label file tools

7.9.1 Label mapping

Often it is necessary to map a label set onto a different one using a map file, for example while working with broad phonetic categories. Our label files are streams of numeric labels. My format of the map-file is:

```
source_label_number target_label_number
% comments ...
src_label src_number target_label target_number
...
```

(an example can be found in `/u/honza/OGI/TRAPS/phone/reducedset29_to_4_classes.janmap`). The script to apply this mapping is `/u/honza/OGI/SPINE/phone/map_can_do_multiple.pl`, which can even do a splitting of one label into two new ones (in case we want to map for example 'ay' to 'aa' and 'y').

7.9.2 Label file analysis

script `/u/honza/OGI/REF_TRAPS/tools/label_anal.pl` does an analysis of labels in a label file. First parameter of this script must be a phoneme list in the format:

```
number label
...
```

(look for example to `/u/honza/OGI/TRAPS/phone/reducedset29.phset` for an example). There should be no extra spaces in the phoneme list. The other inputs to the file are the label file(s) that should be analyzed. The script would work with standard input, as well.

7.10 SPINE

7.10.1 Feature generation

The basis for feature generation for SPINE is the ffrap-file with posteriors from the merger. In the basic setup, there is just a PCA computed on a part of SPINE training set, which is then applied to decorrelate the entire data. In the final experiments on SPINE, different methods of post-processing were tested. The actual procedure differs from experiment to experiment and is always described in `/u/honza/OGI/TRAPS/README_SPINE_TRAPS`. A lot of post-processing was done in Matlab, and there handy Matlab-functions were created:

- in `/u/honza/matlab`:
  - `read_sphinx.m` for reading a sphinx feature file into a matrix.
  - `write_sphinx.m` to do the inverse.
  - `load_many_sphinx.m` to load many Sphinx files according to a list. Good for computing the histograms of the data. There are no special functions to compute the histograms, just peek in `/u/honza/OGI/TRAPS/README_SPINE_TRAPS` how to get them.
  - `sphinx2cov.m` to get a covariance matrix out of a list of Sphinx files. Good if you intend to play with PCA on different types of features and you have the base features in Sphinx files.
- lots of small Matlab-scripts in `/u/honza/TRAPS/traps_itmo34s_fwdlin*`, for example to apply the PCA, to perform sentence-based mean and variance normalization, etc, etc.

The resulting features are in the `Feat` sub-directory of the data directory. Sub-directories for different training sessions should be created first using the command
`mkdir 'cat /u/honza/OGI/SPINE/jan_lists/spine2000.dirs'`

7.10.2 Running the recognizer

Binaries for SPHINX are in `/net/pilsner/u0/honza/SPINE/SPHINX/{s3decode,s3trainer}`. To run Sphinx recognizer on the generated data, you need to copy the directory `c_scripts` from one of my data directories in `/net/pilsner/u0/honza/SPINE/SPHINX/Experiments` (`traps_itmo34s_fwdlin` would be probably the best) and do the following changes:

- set correct data path and feature vector size in `variables.def`, `decode/silent-decode-ci-12utt.csh` and `decode/silent-decode-cd-12utt.csh`.
- run the training: `runall.csh` and look what happens, especially in directory logdir.
- once context-independent models are ready (directory `logdir/02.cd_untied` created, you can run CI-decoding: `decode/silent-decode-cd-12utt.csh`.
- once the whole training is over (CD models created), run `decode/silent-decode-ci-12utt.csh`.

Bot decoding scripts work on the short evaluation set (12 conversations only). Be aware, that SPHINX hangs up sometimes, leaving the process run with 99% CPU usage. Check periodically, if the log-files grow.

7.10.3 Scoring

Results from the CI and CD decoding would be in the directory `Results`, files `recog.SPHINX3.cd_continuous_12utt.txt` and `recog.SPHINX3.ci_continuous_12utt.txt`. To score, go to `/net/pilsner/u0/honza/SPINE/SCORING/scoring_linux` and fire `score_short.csh` with the full path to the txt file as parameters. You will see lots of huge result files created in the `Results` directory. I personally prefer to look at the end of `*.lur` files to get the word error rate (first percentage in the line beginning with "Set Sum/Avg").

Bibliography

[1] 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.

[2] 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.

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

[4] 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.

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

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

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