**UNIVERSITY OF SZEGED**
**Doctoral School of Computer Science**
Szeged

**KU LEUVEN**
**Arenberg Doctoral School**
Leuven

# Noise Robust Automatic Speech Recognition Based on Spectro-Temporal Techniques

**György Kovács**

Supervisors:
Prof. dr. ir. D. Van Compernolle
Dr. L. Tóth

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Szeged and the degree of Doctor in Engineering of KU Leuven

2017

# Noise Robust Automatic Speech Recognition Based on Spectro-Temporal Techniques

**György KOVÁCS**

Supervisory committee:

Prof. dr. ir. D. Van Compernolle, supervisor
Dr. L. Tóth, co-supervisor
Prof. dr. ir. H. Van hamme, assessor
Prof. dr. J. Csirik, assessor

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Szeged and the degree of Doctor in Engineering of KU Leuven

2017

# Preface

"Should recognizers have ears?" – When embarking on my PhD, I was sure I would be able to answer this question one day (I believe this is the sentiment the ancient Greek call hubris). It was only much later that I realised I may have been looking at the problem the wrong way.

In 2015, I was in the audience of a roundtable discussion on Artificial Intelligence in Leuven. The question being discussed was whether creators of AI can benefit from studying nature, and specifically natural intelligence. One member of the audience brought up the topic of evolutionary algorithms as evidence that in fact AI already takes inspiration from nature. To my surprise, this idea was immediately rejected by a speaker who expressed his opinion that using biological terms – such as reproduction, mutation, and recombination – when talking about evolutionary algorithms is just a didactic tool to help us understand the process, but evolutionary algorithms use evolution as an analogy, not as an inspiration. While I still think that evolutionary/genetic algorithms would not be the same as those today without the likes of Darwin and Mendel to introduce the ideas of evolution and genetics to the World, this gave me a new perspective.

Evolutionary algorithms are used because they work. This is independent of what side we take on the issue of AI deriving inspiration from nature, or on the issue of the evolutionary algorithm's origins. The same should apply to spectro-temporal processing: the important question is not necessarily whether recognisers should have ears, or whether applying these methods is akin to giving ears to a recogniser, but whether spectro-temporal processing works. While physiology and psychoacoustics motivated the exploration of this topic, this in and of itself would not be an adequate reason to put these methods to use, if the results were not convincing. For this, here, spectro-temporal speech processing will be examined from a practical perspective. Do these methods lead to better recognition scores? Do word error rates decrease? Such questions will be the focal point of this study.

During my work towards this thesis, I was so fortunate to have the unique opportunity to work with some wonderful people and great researchers in two different countries. I would especially like to thank my supervisors, Dr László Tóth and Professor Dirk Van Compernolle for their help, guidance, and especially for their patience over the years.

I owe a debt of gratitude to many of my fellow PhD students as well (most of whom have since managed to add the letters d and r to their name). I would like to thank Deepak Baby for tirelessly answering my questions about Aurora-4, and even if my barrage of questions wore down his patience, he never showed it. I would also like to thank Tamás Grósz and Gábor Gosztolya for their helpful comments.

And while the work towards my goal was sometimes overwhelming, there was more to this journey than experiments, papers, and conferences. Over these years I shared many memorable moments, such as playing badminton with Joris D., Thom, and Xueru, playing poker at Kseniya's, having a dinner at Jort's, and driving his cat crazy with a laser pointer, participating in city games with the speech group, watching probably the most relatable movie ever made (The PhD movie), playing table tennis with Reza and Hasan, going to Emre's wedding, debating with Vincent over whether the Universe is infinite (a debate that ironically seemed to go on forever), visiting the Christmas fair with Thom and Xueru, sharing stimulating conversations with Alec, Bart, Joris P., Lyan, Meng, and Wang. All these, and many more experience like these (coffee breaks in the basement, lunches at the Alma) helped make my stay in Leuven more welcoming than I could have hoped for.

There were, of course, many who helped and supported me in Hungary as well. The SZESZT group, whose members I could always rely on for sharing their experiences and explaining how to survive in the academic world, or for getting me to try out new things. And I am also gratful for the support of my family, always there to tell me to rest more, worry less, and to listen to my complaints on deadlines, reviews, and papers.

Last but not least I would like to thank David P. Curley for examining this thesis from a linguistic perspective.

# Abstract

Speech technology today has a wide variety of existing and potential applications in so many areas of our life. From dictating systems to voice translation, from digital assistants like Siri, Google Now, and Cortana, to telephone dialogue systems. Many of these applications have to rely on an Automatic Speech Recognition (ASR) component. This component not only has to perform well, but it also has to perform well in adverse environments. After all, a dictating system which requires that we insulate our office, or a digital assistant that cannot work in traffic, or in a room full of chatting people is not so helpful. For this reason, noise robust ASR has been a topic of intensive research. Yet, human-equivalent performance has not been achieved. This motivated many to search for ways to improve the robustness of automatic speech recognition based on human speech perception. One popular method inspired by the examination of the receptive fields of auditory neurons is that of spectro-temporal processing.

In spectro-temporal processing, the aim is to capture the spectral and temporal modulations of the signal simultaneously. One simple way to do so is to extract the features to be used from spectro-temporal patches, and then use the resulting features in the same manner one would use traditional features like MFCCs. There is more than one way to bake a cake, however. And in this case this is true twice over. For one, there are various ways to extract our features from the patches. But there are other, more sophisticated ways to incorporate the concept of spectro-temporal processing into a speech recognition system. In this study we examine many such methods – some simpler, some more sophisticated, but all stemming from the same basic idea. By the end of this study we will demonstrate that these methods can indeed lead to more robust speech recognition. So much so, that they can provide results that are competitive with the state-of-the-art results.

# Abbreviations

| | |
|---|---|
| ANN | Artificial Neural Net |
| ARMA | AutoRegressive Moving Average |
| ASR | Automatic Speech Recognition |
| DCRN | Deep Convolutional Rectifier Neural Net |
| DCT | Discrete Cosine Transform |
| DNN | Deep Neural Net |
| DRN | Deep Rectifier Neural Net |
| FFNN | Feature Finding Neural Net |
| HAS-RGAI | Hungarian Academy of Sciences Research Group on Artificial Intelligence |
| HMM | Hidden Markov Model |
| HSR | Human Speech Recognition |
| MFCC | Mel-Frequency Cepstral Coefficient |
| MLP | MultiLayer Perceptron |
| PER | Phone Error Rate |
| PLP | Perceptual Linear Prediction |
| RASTA | RelAtive Spectral TrAnsform |
| SFFS | Sequential Forward Floating Selection |
| SNR | Signal to Noise Ratio |
| WER | Word Error Rate |
| WFST | Weighted Finite State Transducer |

# List of Symbols

| | |
|---|---|
| $S$ | Spectral representation of a speech segment |
| $S_W^{\delta}$ | the frame size (in milliseconds) applied in getting the S spectral representation |
| $S_W^{\nu}$ | the frame shift (in milliseconds) applied in creating the S spectral representation |
| $S_F^{\#}$ | The number of filters used in the filterbank |
| $P$ | Spectro-Temporal Patch extracted from the spectral representation |
| $P_t^{\delta_p}$ | The physical size of **P** patch in the time domain. It indicates how many milliseconds are covered by the patch. |
| $P_f^{\delta_p}$ | The physical size of **P** patch in the frequency domain. |
| $P_t^{\delta_t}$ | The "technical" size of **P** patch in the time domain. It indicates how many frames are covered by the patch. |
| $P_f^{\delta_t}$ | The "technical" size of **P** patch in the frequency domain. It indicates how many mel-channels are covered by the patch. |
| $P_f^{\#}$ | The number of patches used to cover the whole frequency domain. |
| $P_f^{\nu_p}$ | The proportion of patches that overlap in the frequency domain. |
| $\overline{P}$ | The originally 2-dimensional **P** patch written in vector form. |
| $\overline{P}^{\delta_t}$ | The length of $\overline{\mathbf{P}}$ vector. |

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Automatic Speech Recognition (ASR) in general is a task where the computer has to transcribe the sound detected by a microphone to a sequence of phones or words. There are many variations of the general task that can make it more or less difficult. With regard to the speaker for example, ASR may be speaker-dependent [52], where the computer only has to recognise the speech of a few or just one speaker, or it may be speaker-independent [93], where the computer has to recognise speech, regardless of the speaker's identity. The difficulty of the ASR task can also vary based on the size of the dictionary, the scope of which may extend from a few digits [135], to millions of words [61, 62]. There are many other variations in the ASR task, such as speech that is read or spontaneous speech, speech from non-native speakers, dysarthric speech, and speech from speakers with a heavy accent. For the purposes of this study however, the most important parameter is the presence or absence of noise.

In today's age when speech technology permeates our life, its noise robustness becomes more important. The ASR applications of today have moved out of our computers and into our cars, phones, and even watches. We can take them with us anywhere, but if we actually want to use them in all the different environments we inhabit (running cars, crowded bars, or busy streets), they have to be able to cope with a large variety of noise. In other words, we require speech recognition applications to be noise robust, similar to human speech perception. In all of these environments, when the conditions are not extreme, we can understand each other quite easily. Machines, however, despite the steady progress made, are only recently getting close to human performance [154, 200], and clearly fall behind when noise is introduced [140], regardless of whether it is a task of phone classification [153] or word recognition [151].

This gap in performance between Human Speech Recognition (HSR) and Automatic Speech Recognition (ASR) motivated many researchers to urge a closer collaboration between the two fields [21, 28, 78, 150, 202]. The rationale behind this proposal is that by studying the system that is working more effectively (i.e. human speech understanding), we should be able to improve the system that is less effective (i.e. ASR). Not by copying the former note for note, but by striving to understand which of its properties are relevant for decoding the speech signal and improving the performance of the latter [80]. Different approaches have been proposed that take inspiration from human speech understanding. For example, some try "to make computers learn speech, like a baby does" [46]. This approach has the advantage of not relying on preprogrammed linguistic knowledge (like phoneme sets, dictionaries and acoustic models), as this knowledge may be acquired through interactions with the user. It is also possible to base our efforts on knowledge about human auditory processing [111]. We have seen the success of the latter approach on several occasions. For instance, modelling human hearing's decreased sensitivity at higher frequencies (above 1000 Hz) with the mel-scale leads to improved speech recognition performance [104]. Perceptual Linear Prediction (PLP) [79] is another technique that works by approximating certain properties of hearing, and was shown to be advantageous in ASR [9, 103]. There are several other methods available that can improve ASR performance based on the analysis of auditory processing, one of which is called spectro-temporal processing.

In the early '80s Aertsen et al. examining the auditory system of grassfrogs coined the phrase, Spectro-Temporal Receptive Field (STRF) [5, 6] to describe the spectro-temporal sensitivity of auditory neurons. Later, experiments on other animals reinforced the notion that these cells in the auditory cortex are tuned to localised spectro-temporal modulations [36, 218]. It was also indicated that processing works with spectro-temporal patterns, instead of consecutive steps of spectral and temporal filtering [45]. These results lead to the idea of spectro-temporal processing [116], which seeks to capture these spectro-temporal modulations in speech using various methods. In this thesis we will focus on some of these methods, and propose new techniques inspired by them.

As mentioned before, we do not intend to examine whether the observed improvement in performance (when there is such) is due to the given methods' roots in auditory processing, nor do we intend to investigate whether applying these methods indeed makes ASR more similar to HSR. Although we will try to go beyond raw data by outlining our motivation, and drawing conclusions that reach further, our main focus will be on accuracy scores and error rates. The main question we ask will be whether the proposed methods improve the recognition performance. We will do so using three tasks; namely phone classification, phone recognition and word (continuous speech) recognition.

## 1.1 Overview of the Speech Recognition Process

Here, an overview of the experiments performed will be given. These experiments were mainly carried out in an HMM/ANN hybrid architecture, the main modules of which are depicted in Figure 1.1. As the task is to transcribe the speech signal picked up by the microphone, the first step is to convert the speech to a format suitable for digital processing. This step entails the analogue-digital conversation of the signal, by sampling and quantisation [104]. Once digitisation is performed, the next step is feature extraction. For the purposes of this thesis, this step will be discussed as two separate steps. In the first one, a spectral representation is extracted from the speech signal, and then this representation is used in the second step to extract the features used as an input in the HMM/ANN hybrid. Following Kleinschmidt [116], the two parts will be referred to as primary and secondary feature extraction respectively. The output of this step is a feature vector corresponding to given time-slices (or frames) of the speech. In the case of a phone classification task, the features corresponding to the same phone are aggregated and the resulting feature vector is used in the neural net for classification. In the case of a phone recognition or word recognition task, the individual frames are processed by the neural net, which assigns a set of probability values to these frames, each signifying the probability that the given frame comes from a certain phoneme model. These probability values are used by the HMM to transcribe our input as a series of phones in a phone recognition task; or in the case of a word recognition task, to provide input for the decoder (which is typically implemented by a Weighted Finite State Transducer - WFST), so that it can transcribe the input as a series of words [183]. Figure 1.1 shows that both for the recognition of phones and words in continuous speech, a language model is used. The technique used for this in the experiments will also be shortly described in the following sections, where I further elaborate on the different modules. I will do so, with emphasis on describing techniques that can be applied in the given step to increase the noise robustness, in order to put my work in context with earlier efforts in achieving noise robust ASR.

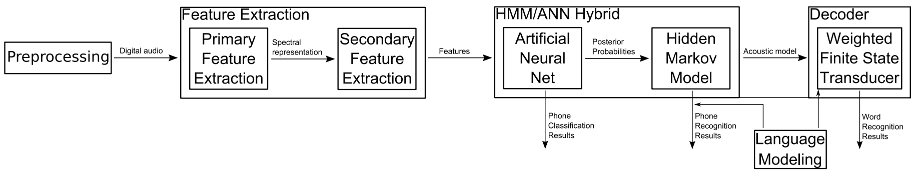

Figure 1.1: Overview of the speech recognition process.

## 1.2 Digital audio

The two steps of digitisation (the process of converting the analogue speech signal to a digital one), are called sampling and quantization. Sampling is performed by measuring the amplitude of the signal at a particular point in time. An important parameter of this step is the frequency of sampling, i.e. the number of measurements taken in a second. It has been known for a long time that the frequency of sampling (or sampling rate) should be at least twice as high as the highest frequency component of the signal [143]. This frequency is of course different for each signal. For human speech, a 20 kHz sampling rate would be needed for pefect quality, but for microphone speech (as we will later see) 16 kHz is often used [104]. As even a 16 kHz sampling rate means sixteen thousand measurements every second, the results of these measurements have to be stored efficiently. Quantization is applied for this reason, where real-valued numbers are represented as (16-bit) integers. As in the experiments carried out for the thesis digitisation was not performed, and instead previously digitised speech from various speech corpora was used, the bulk of this section will focus on describing these speech databases.

### 1.2.1 TIMIT

Upon creating a single language database, many design considerations have to be taken into account. Design considerations, such as the phonetic coverage of the language (the occurrence of phonemes and phoneme tuples), different speaking styles (isolated words, read speech, and spontaneous/conversational speech), and distribution of speakers (taking into consideration age, sex, and dialect). The same was true for the TIMIT speech corpus, which was created as a joint effort by Texas Instruments (TI), the Massachusetts Institute of Technology (MIT) and SRI International [129].

Regarding speaking style, TIMIT is a read-speech corpus, consisting of sentences read out by different speakers. These sentences were selected from three sets. The first set (made up of 2 sentences), namely the calibration sentences (provided by SRI) was created in such a way as to provide significant dialectical differences. The second set (450 sentences provided by MIT) was also manually created to provide an adequate phonetic coverage. The last set (1890 sentences provided by TI) was automatically created, as manual creation of sentences was not only a complex and time-consuming task, but also resulted in sentences lacking in stylistic variety [128]. In my experiments, the second and third sets of sentences were used.

To ensure speaker variability, creators of the corpus accounted for such variables as gender, age, height, race, education level, and dialect region of the speakers [59]. The dialects were grouped into 8 categories; namely, New England, Northern, North Midland, South Midland, Southern, New York City, Western, and "Army Brat" (for the speakers who had changed their residence many times). Sentences from the MIT set were each recorded from 7 speakers, while each sentence in the automatically genereated TI set was read by only one speaker. These recordings were digitised using a 16 kHz sampling rate.

Once the sentences were recorded, train/test partitioning had to be carried out. Here, the training set consists of 3696 sentences (with a combined duration of 3.14 hours), from 462 speakers. The test set has two versions. The full test set consists of 1344 sentences (with a combined duration of 0.81 hours) from 168 speakers, while the core test set has 192 sentences (with a combined duration of 0.16 hours) got from 24 speakers (from each dialect region, the sentences of 2 male and 1 female speaker were used) [139]. Despite its relatively small size, TIMIT still has an important role in testing new ideas, as improvements achieved on it can be observed to scale to larger speech corpora [199].

Another important question is the recognition task to undertake on a database. The tasks carried out here on the TIMIT speech database were frame recognition, and phone classification and recognition. Originally, transcription of the database was done using 61 phonetic labels (see Table 1.1). In some experiments

Table 1.1: The 61 phones originally used for transcribing the TIMIT database [139].

| Phone | Example | Phone | Example | Phone | Example | Phone | Example |
|---|---|---|---|---|---|---|---|
| iy | b*ee*t | ax | *a*bout | zh | a*z*ure | hh | *h*ay |
| ih | b*i*t | ix | deb*i*t | f | *f*in | hv | a*h*ead |
| eh | b*e*t | axr | butt*er* | th | *th*in | el | bott*le* |
| ey | b*ai*t | ax-h | s*u*spect | v | *v*an | bcl | b closure |
| ae | b*a*t | jh | *j*oke | dh | t*h*en | dcl | d closure |
| aa | b*o*b | ch | *ch*oke | m | *m*o*m* | gcl | g closure |
| aw | b*ou*t | b | *b*ee | n | *n*oo*n* | pcl | p closure |
| ay | b*i*te | d | *d*ay | ng | si*ng* | tcl | t closure |
| ah | b*u*t | g | *g*ay | em | botto*m* | kcl | k closure |
| ao | b*ough*t | p | *p*ea | nx | wi*nn*er | q | glotal |
| oy | b*oy* | t | *t*ea | en | butt*on* | | stop |
| ow | b*oa*t | k | *k*ey | eng | Wash*ing*ton | pau | pause |
| uh | b*oo*k | dx | mu*dd*y | l | *l*ay | epi | epenthetic |
| uw | b*oo*t | s | *s*ea | r | *r*ay | | silence |
| ux | t*oo*t | sh | *sh*e | w | *w*ay | h# | begin/end |
| er | b*ir*d | z | *z*one | y | *y*acht | | marker |

Table 1.2: The 39-element list reduced from the 61-element list of phones originally used for transcribing the TIMIT database [133].

| Label no. | Phone | Example | Folded | Label no. | Phone | Example | Folded |
|---|---|---|---|---|---|---|---|
| 1 | iy | b*ee*t | | 23 | dx | mu*dd*y | |
| 2 | ih | b*i*t | | 24 | s | *s*ea | |
| 3 | eh | b*e*t | | 25 | sh | *sh*e | |
| 4 | ey | b*ai*t | | 26 | z | *z*one | |
| 5 | ae | b*a*t | | 25 | zh | a*z*ure | |
| 6 | aa | b*o*b | | 27 | f | *f*in | |
| 7 | aw | b*ou*t | | 28 | th | *th*in | |
| 8 | ay | b*i*te | | 29 | v | *v*an | |
| 9 | ah | b*u*t | ax-h | 30 | dh | t*h*en | |
| 6 | ao | b*ou*ght | | 31 | m | *m*o*m* | em |
| 10 | oy | b*oy* | | 32 | n | *n*oo*n* | nx |
| 11 | ow | b*oa*t | | 33 | ng | si*ng* | eng |
| 12 | uh | b*oo*k | | 32 | en | butt*on* | |
| 13 | uw | b*oo*t | ux | 34 | l | *l*ay | |
| 14 | er | b*ir*d | axr | 35 | r | *r*ay | |
| 9 | ax | *a*bout | | 36 | w | *w*ay | |
| 2 | ix | deb*i*t | | 37 | y | *y*acht | |
| 15 | jh | *j*oke | | 38 | hh | *h*ay | hv |
| 16 | ch | *ch*oke | | 34 | el | bott*le* | |
| 17 | b | *b*ee | | 39 | cl (sil) | unvoiced | pcl,tcl |
| 18 | d | *d*ay | | | | closure | kcl,q |
| 19 | g | *g*ay | | 39 | vcl (sil) | voiced | bcl,dcl |
| 20 | p | *p*ea | | | | closure | gcl |
| 21 | t | *t*ea | | 39 | epi (sil) | epinthetic closure | |
| 22 | k | *k*ey | | 39 | sil | silence | h#,pau |

this phone set was used for training purposes, while in others 183 tri-state monophone phone models or 858 triphone tri-state phone models were used. But in all cases of evaluation (and some cases of training), a modified version of this list was used, where the 61 phonetic labels were reduced into a 39-item list, according to the suggestions of Lee and Hon [133], which has become the standard for evaluation [139]. For this first they identified 15 allophones, and folded them, creating a list of 48 possible labels. Then they identified 7 groups on the shortlist ({sil, cl, vcl, epi},{el, l},{en, n},{sh, zh},{ao, aa},{ih,ix},{ah, ax}), where within-group confusions were not counted. This process (see Table 1.2) lead to 39 labels. When otherwise not specified, these 39 labels were used for phone recognition, with a simple bi-gram language model.

The focus of this study is automatic speech recognition in adverse environments (i.e. with noise contaminated speech). The TIMIT speech corpus, however, is comprised of only clean speech data. Because of this, to examine the noise robustness of methods studied here, different types of noises had to be added to the original test sentences. These included both real-life and artificial noise types, namely

- Babble noise: real-life noise, simulating the effect of people talking in the background;
- Factory noise: real-life noise, simulating the effect of a nearby production line working;
- Volvo noise: real-life noise, recorded in a running car with its engine running
- Pink noise: artificial noise that has the highest energy at 0 Hz, and tails off at higher frequencies; and
- Band-limited noise: artificial noise created in our research group by filtering white noise with a bandpass filter active between 3000 Hz and 5000 Hz

These real-life noise types were taken from the freely available samples of the Noisex-92 database [230], and both the real-life noise types and the band-limited noise were added to the original sound file with various signal to noise ratios (SNR) using the FaNT tool [86]. The test set contaminated with pink noise was provided by Jake Bouvrie [24].

### 1.2.2 Aurora-4

Another English language database used during the experiments in this study was the Aurora-4 corpus [175]. This is a standard large vocabulary continuous speech database that, like the TIMIT speech corpus, was built upon read-speech. Aurora-4 was created based on the DARPA (Defense Advanced Research Projects Agency) Wall Street Journal corpus [177], which contains sentences taken from newspaper articles that had been published in the Wall Street Journal between 1987 and 1989. Similar to the TIMIT corpus, these recordings were created with a sampling rate of 16 kHz, but in order to test the degradation of speech recognition performance in different environments, it also contains a version that is downsampled to a 8 kHz sampling rate. In the experiments conducted here, however, only the former was utilised.

Aurora-4 is also partitioned into training and test sets. The training set contains recordings taken from 83 speakers, for a total of 15 hours, and it has two versions. One contains only clean speech (7138 utterances) recorded with a Sennheiser HMD-414 close-talking microphone. This allows one to test how well a speech recognition system performs when the train and test conditions are mismatched. In the other version of the training set (7137 utterances), several utterances are replaced with their counterparts that are noise contaminated (with car, babble, restaurant, street, airport, or train noise) and/or recorded with secondary microphones (secondary microphones that included common microphone types such as Crown PCC-160, Sony ECM-50PS, and a Nakamichi CM100) [175]. Training a system on this set allows one to test its performance when one knows beforehand the type of noise and microphone mismatch that will occur during testing.

The test set contains 14 subsets, each consisting of the same 330 sentences. These sentences were spoken by 10 speakers (33 sentence per speaker). The first seven subsets were recorded with the primary (Sennheiser) microphone, while the second seven subsets were recorded with different, secondary microphones. Out of the seven subsets for both groups, the first one consists of clean speech, while the other six consists of speech subsequently contaminated with one of the six different types of noises with an SNR between 5 dB and 15 dB. Results of the test set are commonly reported in 4 groups:

- Test set A: results on clean speech recorded with the Sennheiser microphone;
- Test set B: results on speech recorded with the Sennheiser microphone and contaminated with different types of noise;
- Test set C: results on clean speech recorded with the secondary microphones; and
- Test set D: results on speech recorded with the secondary microphones and contaminated with different types of noise.

The task on the Aurora-4 database is large vocabulary continuous speech recognition (LVCSR). An evaluation of the task was carried out on the above mentioned test sets A, B, C and D. As the sentences for the test set were selected in such a way that a 5000 word dictionary (distributed with the corpus) covers them all, there are no out-of-vocabulary (OOV) errros in the evaluation. Thus the richness of the language model does not affect the recognition accuracy, which means that we should get a clearer comparison of acoustic models.

### 1.2.3 “Szeged” Hungarian Broadcast news database

The third speech database used during the experiments performed here was a corpus of Hungarian Broadcast News, recorded and transcribed at the Research Group on Artificial Intelligence (HAS-RGAI), belonging to the Hungarian Academy of Sciences and the University of Szeged Informatics Department [226]. For the database, 70 news broadcasts were recorded from 8 different television channels. These recordings were cut into few sentence long blocks, and the resulting segments were placed into one of the following categories:

- Clean speech: speech in this category is well articulated, mostly planned, and has a minimal level of background noise. Most recordings in this category were originally filmed in a studio, and were spoken by professional newscasters. This is the category from which recordings were used in our experiments.
- Noisy speech: speech in this category is still mostly planned, but it has a higher level of background noise. Recordings in this category are typically taken from on-site reporters speaking in a noisy environment.
- Spontaneous speech: speech in this category is spontaneous, typically coming from an interviewee, whose speech is not necessarily as well articulated as that of an interviewer or professional newscaster.

Recordings of the 70 newscasts were partitioned into a train set, a development set, and a test set: 44 newscasts (altogether approximately 5.5 hours) were used for training, 9 newscasts (altogether approximately 1 hour) were used for development and validation, while the remaining 17 newcasts (altogether approximately 2 hours) were used for testing purposes. This partitioning of newscasts was carried out in such a way that each set contained recordings from all television stations. All the recordings were orthographically typed, and the corresponding transcripts were created with a simple phonetic transcriber. The phonetic labels of the database consist of 52 categories. Here, the task in the experiments conducted on the “Szeged” Hungarian Broadcast news database was to recognise these 52 phones in speech.

## 1.3 Feature Extraction

### 1.3.1 Primary feature extraction

In primary feature extraction our task is to generate the spectral representation $S$ of the digital audio. This should tell us how the different frequencies making up the signal change in time. Several such representations are used in the speech recognition literature, the most successful of which incorporate attributes of human hearing into the process [157]. Here, some of the most popular ones will be outlined.

#### Mel-scale spectrum

Used as a preliminary step in extracting MFCC features, but also as a primary feature extraction step for other methods, the production of the log mel-scale spectrum is carried out in four steps:

1. **Pre-emphasis**: To compensate for the high-frequency parts of the signal being suppressed during speech production [147], the speech signal is sent to a pre-emphasis filter. The most commonly used filter for this purpose is a Finite Impulse Response (FIR) filter that is characterised by the following formula [46]

$$H_{pre}(z) = 1 - a_{pre} z^{-1}. \tag{1.1}$$

2. **Framing and windowing**: As the first step, short segments of speech have to be selected, where we can assume the signal to be stationary [104]. We do so by applying a sliding window on the signal (the part of speech signal that is selected by the window is called a frame [46]). This windowing process is defined by several important parameters. One is the width of these windows ($S_W^\delta$) for which candidates are typically between 20 and 30 milliseconds [180]. Another important parameter is the timespan between the beginning of two consecutive windows (or the offset of the windows - $S_W^\nu$), determining the frame shift, which is usually between 10 and 20 milliseconds [180]. Another important parameter in this process is the shape of these sliding windows. The role of the properly chosen shape here is to emphasise the middle of the frame, and to prevent high-frequency artifacts that might result from the windowing process [180]. Several types of windows can be used (Dirichlet, Triangle, Hamming, Hanning, etc.), but the use of Hamming windows is the most common [76].

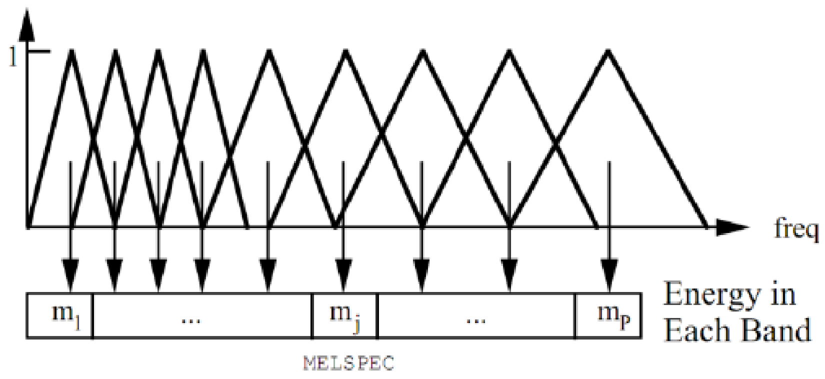

Figure 1.2: Mel-Scale Filter Bank [236].

3. **Spectral analysis**: To extract spectral information from our windowed signal, we perform a spectral analysis on it. This allows us to learn how much energy the signal contains at different frequency bands [104]. This is often carried out via the Fast Fourier Transform (FFT). We can visualise this information using a spectrogram that represents the energy of the signal at every frequency, at every point in time [46].

4. **Mel-filtering**: It has been shown that the sense of pitch (the perception related to frequency) in the human auditory system is proportional to the logarithm of the tone's frequency [179]. Various models exist of the pitch perception scales, such as the mel-scale, the Bark scale and the ERB scale [83]. Here, the mel-scale is used, defined by the following formula [236]
$$mel(f) = 2595 log_{10}(1 + \frac{f}{700}) \tag{1.2}$$
For the implementation of this filterbank in the HTK toolkit, the magnitude coefficients from the FFT are binned by correlating them with each triangular filter (see Figure 1.2). The triangular features normally span from 0 Hz to the Nyquist frequency, but it is also possible to implement lower- and upper-frequency cutoffs. After filtering, as a last step, the log of each mel spectrum value is calculated. The reasoning behind this is twofold. For one, the human response to signal level is approximately logarithmic. Furthermore, calculating the logarithm makes the results less sensitive to variations in the input [104].

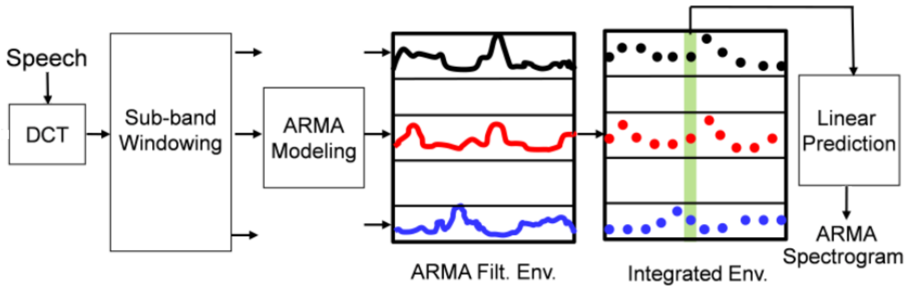

Figure 1.3: Block schematic of ARMA feature extraction.

### Power Normalised Spectrogram (PNS)

A primary feature extraction method utilised to improve the robustness of the resulting features against environmental differences is the extraction of the Power Normalised Spectrogram (PNS) [112]. The first steps in the extraction of PNS are very similar to what we saw in the case of the mel-scale spectrum. Both start with a pre-emphasis, followed by framing and windowing, and then by a spectral analysis. However in producing the PNS, instead of the triangular filters used in creating the mel-scale spectrograms, the Gammatone auditory filterbank is used [176]. This step is followed by a medium-time power bias subtraction. After which, similar to the log mel-scale spectrum, a nonlinearity is applied on the results. However in this case the logarithmic function is replaced by a power-function nonlinearity, which results in the output being close to zero when the input is very small, a property that matches the observations about human auditory processing [112]. The resulting representation can be exploited to create Power Normalised Cepstral Coefficients (PNCC) or used as an input for secondary feature extraction methods like the application of Gabor filters [31].

### AutoRegressive Moving Average (ARMA) spectrogram

The last primary feature extraction method to be discussed here is also the most recently introduced one. The AutoRegressive Moving Average (ARMA) feature extraction method was published by Ganapathy in 2015 [57]. Similar to many other techniques discussed in this study, this method was also developed to achieve noise robust results, even in the presence of a mismatch between the training and test environment. The block schematic of ARMA processing is shown in Figure 1.3.

The ARMA features are derived using AutoRegressive Moving Average spectrogram modelling. This process is a generalisation of the traditional AR modelling and it can estimate band-pass characteristics, while the AR modelling typically estimates low-pass characteristics [57]. The ARMA modelling approach is applied on the sub-band DCT components to estimate the temporal envelopes. The ARMA filtered envelopes are converted to a short-term spectral representation by energy integration. Afterwards, a linear prediction-based spectral smoothing is applied on this spectrogram.

Let $x[n]$ denote the input signal for $n = 0, \ldots, N-1$ ($N$ corresponds to 1000 ms here) and let $X[k]$ denote DCT components of the signal $x[n]$. In the proposed framework, the DCT components are used in an ARMA modelling framework to estimate the sub-band envelope. Specifically, the set of coefficients $a_l, l = 1, \ldots, p$ and $b_m, m = 1, \ldots, q$ are estimated such that

$$X[k] = \sum_{l=1}^{p} a_l X[k-l] + \sum_{m=0}^{q} b_m U[k-m], \tag{1.3}$$

where $p, q$ denote the model order of the AR and MA components and $U[k]$ denotes a zero-mean white noise signal. The AR model is a special case of the ARMA model with $b_m = 0, \quad for\ m\ >\ 0$. The ARMA envelope is given by

$$\hat{E}_x[n] = \frac{|\sum_{m=0}^{q} b_m e^{-i2\pi mn}|^2}{|\sum_{l=0}^{p} a_l e^{-i2\pi ln}|^2} \tag{1.4}$$

The ARMA envelope is the AR envelope (denominator) multiplied by a finite impulse response (FIR) filter provided by the MA modelling (numerator). The MA filter acts as a modulation filter over long temporal regions of signal. Hence, ARMA modelling combines AR estimation with a data-driven modulation filter. Here, gain-normalised ARMA envelopes ($a_0 = 1$ and $b_0 = 1$) are used. The estimation of ARMA model parameters is more cumbersome than that of AR modelling and an iterative gradient descent approach is employed [142]. For the ARMA spectrogram estimation, $p = 40, q = 6$ poles per second per sub-band is used. Also a compression factor of 0.2 is used on the MA part for envelope computation.

Lastly, the ARMA filtered temporal envelopes were processed with spectral smoothing using linear prediction. This is achieved by integrating the sub-band ARMA envelopes in windows of 25 ms duration with a shift of 10 ms. The resulting spectrally smoothed ARMA spectrogram coefficients serve as primary features. The ARMA spectrogram used in the experiments consisted of 39 spectral bands, which was supplemented by the same number of derivative-like features got as the difference between neighboring bands ($band(K+1) - band(K-1)$).

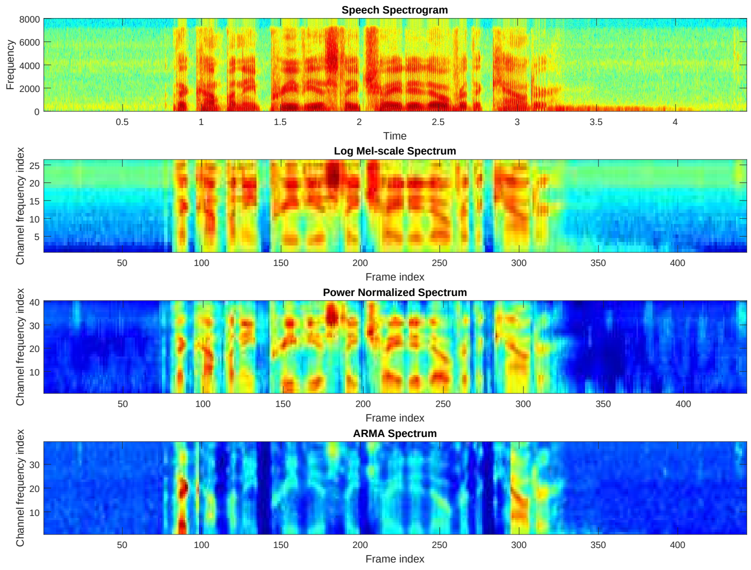

Figure 1.4: Comparison of spectral representations using the sentence "The company previously traded over the counter" from the clean test set of the Aurora-4 database.

Figure 1.4 shows the difference between the spectral representations described here (and the simple spectrogram), on a sentence taken from the clean test set of the Aurora-4 database. Here, for better visualisation, the ARMA spectrum does not include the derivative-like features.

## 1.3.2 Secondary feature extraction

Obtaining a spectral representation such as the mel spectral features (or their logarithmised version) is only the first step in the process of feature extraction. For dimensionality reduction reasons, and also to reduce the interdependency of the features [46], further processing steps are employed. Although Deep Neural Nets (DNNs) are shown to be capable of successfully using primary features (such as the log mel spectral features) as their input [68, 162, 188, 223], it has been shown that the use of robust secondary features can still be beneficial [145, 146] especially when there is a mismatch between the training and testing environment [30].

### Mel-Frequency Cepstral Coefficient (MFCC)

By applying the DCT on the log mel-spectral feature vectors derived from the primary feature extraction, we transform our data into cepstral space. This representation has the advantage of separating the glottal source (fundamental frequency, details of the glottal pulse, etc.) and the filter (vocal tract), which will be useful in distinguising different phones [104]. It also has the added benefit of its coefficients being less interdependent than log mel-spectral features [46].

When extracting features for phone recognition, it is common practice to just keep the first 12 cepstral values (representing information related to the vocal tract), along with the energy from the frame [104]. Then the $\Delta$ (velocity) and $\Delta\Delta$ (acceleration) coefficients are calculated from the resulting 13 features in the following way using a wide context:

$$\Delta_T = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{T+\theta} - c_{T-\theta})}{2 \cdot \sum_{\theta=1}^{\Theta} \theta^2}, \tag{1.5}$$

where $\Delta_T$ is the $\Delta$ coefficient at $T$ time calculated from the coefficients between $c_{T-\theta}$ and $c_{T+\theta}$ [236]. This results in a 39 dimensional feature vector. As the MFCC is a widely used feature set [58], in the experiments we often used the results obtained using MFCC as a baseline of comparison to the results achieved with features obtained using more sophisticated feature extraction methods.

### Perceptual Linear Prediction (PLP)

Another widely used feature extraction method is the Perceptual Linear Prediction (PLP) [87] method. Similar to spectro-temporal feature extraction it was proposed in order to incorporate auditory-like features into speech processing [157]. While originally it was designed to work on a Bark scale spectrogram [79] (where the Bark pitch perception scale is used instead of the mel-scale), “there seems to be no intuitive reason why the Bark filter-bank should be optimal for PLP” [87] (indeed Hönig et al. found a decrease in WER when the Bark scale was replaced with the mel-scale), hence some tools (such as the HTK toolkit) compute the PLP features based on the standard mel-scale spectrum [236]. Regardless of the perception scale used, after the transform to the pitch perception scale, two more psychophysically inspired transformations are used; these being equal-loudness preemphasis (to approximate the sensitivity of human hearing at different frequencies, as described by Robinson and Dadson [194]), and the application of the intensity-loudness power-law (to approximate the power-law of hearing [213]). Following these steps an Inverse Discrete Fourier Transform (IDFT) is applied, then the PLP coefficients are provided by Autogregressive Modelling.

### Power Normalised Cepstral Coefficients (PNCC)

In its original implementation the Power Normalised Cepstral Coefficients (PNCC) are derived from the PNS by applying a Discrete Cosine Transform on it, followed by mean normalisation [112]. The resulting feature set has been shown to lead to a better recognition accuracy than MFCC or RASTA-PLP processing in the presence of common additive noise types, and reverberation [113]. It was also heroded as the new feature set that is the most promising for robust ASR [108, 109]. However the noise robustness of the PNCC has a downside because it has a lower performance (compared to MFCC or PLP) under clean speech conditions [168]. Furthermore, the effectiveness of PNCC features strongly depends on the initialisation of the initial power parameter, especially in the case of short speech segments [53].

### Spectro-temporal feature extraction

There are various different approaches for capturing spectro-temporal modulations that are intended to increase ASR robustness. For the purposes of speech recognition, it is likely that auditory-based approaches would be useful [116]. As later Gabor filters and 2D DCT will be examined at length, these methods will not be discussed here (for more details, see Chapter 2). A discussion of spectro-temporal features here will be confined to a short description of two additional methods:

- Sigma-pi cells: originally proposed to be used as features for isolated word recognition, this method entails the multiplicative combination of two spectro-temporal windows, using multiplier cells to perform a logical AND function [67]. It was also shown that, applied in the right combination, this method can be applied for signal-to-noise ratio estimation [119]. A generalisation of the original method for multiple windows and variable window sizes was successfully used in noise robust isolated word recognition [115]. Sigma-pi cells have also been successfully employed for the task of automatic stress detection in speech [77].
- Fuzzy logic units: an extension of the sigma-pi cell feature extraction method, to account for its ambiguity, with other fuzzy logical operations (OR, NOR, NAND) [115].

## 1.4 Acoustic Modelling

Before discussing the Acoustic Model in detail, we should first take a step back and ask ourselves what our goal is. In Section 1.1, it was mentioned that continuous speech recognition and phone recognition were the tasks to be performed. But how can we formalise these tasks? The explanation of it here will be presented following the thought process of Jurafsky and Martin [104]:

Let us think of the speech segment of duration $t$ we hear (or in other words, observe) as a series of observations $O$ (where $O = o_1, o_2, ..., o_t$). We expect the computer to take this speech, and transcribe it as a W sequence (where $W = w_1, w_2, ..., w_n$) of words (or subword units – for example phones). That is to say, we expect the computer to select the best sentence (which in our case is the most probable sentence) from all the possible sentences, given observation $O$. The following equation formalises this notion:

$$\hat{W} = argmax_{W \in \mathcal{L}} P(W|O), \tag{1.6}$$

where $\mathcal{L}$ is the set of all possible sentences in the given language.

Applying the Bayes-rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.7}$$

to this, we get the following:

$$\hat{W} = argmax_{W \in \mathcal{L}} \frac{P(O|W)P(W)}{P(O)} \tag{1.8}$$

Since $P(O)$ is constant for observation $O$ (as its value does not depend on the present selection of sequence $W$), for the purpose of obtaining the most probable sentence, it should suffice to solve the following:

$$\hat{W} = argmax_{W \in \mathcal{L}} P(O|W)P(W) \tag{1.9}$$

This means that the most probable sentence ($\hat{W}$) can be determined from the conditional probability of observation $O$ given sentence $W$ ($P(O|W)$) and the probability of sentence $W$ (P(W)) by calculating their product for each sentence, and selecting the sentence for which this value is the highest. The latter probability could be got from the language model (see Section 1.5), while the former probability could be got from the acoustic model (which in our case will be a HMM/ANN hybrid – see sections 1.4.1 to 1.4.3).

### 1.4.1 Hidden Markov Model

Although in the HMM/ANN hybrid model [166] the application of ANNs precedes the application of HMMs, we can better understand the model if here this order is reversed. A HMM model is an automaton, defined by the following tuple: $M = (Q, A, B, \Sigma)$, where

- $Q$ is the finite, N-element set of states (while the number of hidden states can also be countably infinite [14], this simplication could be used in this short discussion of the topic, for the sake of brevity), where $q_o$ and $q_F$ $\in Q$ are special states, the starting and end state.
- $A$ is a matrix of transitional probabilities of a size N times N. Here $a_{ij} \in A$ denotes the probability of the model undergoing a transition from the i-th state to the j-th state (the main diagonal containing the probability of transition to the same state; in other words, the probability of remaining in the same state).
- $B$ is the set of emission probabilities. It describes for observation $o_t \in \Sigma$ and for each $q_i \in Q$ the probability of the observation emitted by $q_i$ state $(b_i(o_t))$
- $\Sigma$ is the set of emissional symbols. When this set is discrete, B can also be treated as a matrix, in which rows correspond to states and columns correspond to emissional symbols (or the other way around), and each value in the matrix describes the probability of the given state emitting the given symbol.

From this description it can be seen that a number of implicit assumptions have been made about the use of HMMs as acoustic models. One assumption is that we can transcribe speech as a discrete series of symbols taken from a finite (or countably infinite) set and that these symbols can be mapped to the states of the HMM. Another implicit assumption can be deduced from the use of the matrix of transitional probabilities: the way $A$ matrix is set up suggests that if at a $t$ time the HMM is in the state $q_t$, the probability of the HMM being in any given state at $t + 1$ time only depends on the current state ($q_t$), and not on the state the model was in the $t - 1$ or $t - 2$ or any previous step. Formally,

$$P(q_{t+1}|q_t, q_{t-1}, ..., q_0) \approx P(q_{t+1}|q_t) \tag{1.10}$$

This is called the Markov assumption.

### Decoding

When working with HMM models there are three important problems to consider. One is the problem of evaluation, or computing the probability of a specific observation sequence, given an HMM. That is, given the HMM model $M = (Q, A, B, \Sigma)$, and the sequence of observations $O$ ($O = o_1, o_2, o_t$), what is the probability of $M$ generating $O$ (P(O|M))? This can be used for example in isolated word recognition.

Another problem is the learning, or the parametrisation of the models, tuning its parameters using a given training set or task. That is, given the sequence of observations $O$ (which will serve as our training set), how should the parameters of the $M$ model be set so as to maximise the P(O|M) conditional probability?

The problem, however, that we should focus on here is finding the sequence of states our model goes through while emitting the sequence of observations $O$. In a regular automaton this would be a straightforward task, as for each step we are aware of the state the automaton is currently in. However in HMMs as the word "hidden" in the name suggests, this information is not available. We can only infer the current state of the automaton from its emissions. Thus the task is to find the $Q$ sequence of states from model $M$ that have most likely created the O observation seen. This formally means that our goal is to find the $Q$ sequence of states in $M$ model for which the conditional probability $P(O|Q, M)$ is the highest. This task is also known as decoding.

There are various algorithms in the literature that are used to handle the task of decoding. One such algorithm (also present in the popular speech recognition toolkit, called HTK [236]) that is based on dynamic programming is the **Viterbi** algorithm [233]. While running the algorithm a V matrix with $T$ columns (where $T$ is the number of observations) and $N$ rows (where $N$ is the number of states in the model $M$) is dynamically computed. A cell in this matrix reflects the probability of the most likely path leading up to the state and time corresponding to the cell. Formally,

$$viterbi[t, i] = max(q_0, q_1, q_2, .., q_t = i, o_i, o_2, ..., o_t|M) \tag{1.11}$$

As a simplification we will assume that if the optimal path for the full sequence crosses $q_i$ state at time $t$, then the optimal path of length $t$ ends in state $q_i$. Conversely, if at time $t$ the most likely path leading to $q_i$ state is more likely than the most likely path leading to any other state from the $Q$ set, then the most likely path at $t + 1$ time will be calculated using $v[t, i]$, in the following way:

$$viterbi[t + 1, j] = max_{q_i \in Q}(viterbi[t, i] \cdot a_{ij} \cdot b_i o_{t+1}). \tag{1.12}$$

The algorithm described above gives a result for a series of observations and an HMM. As we generally have more than one model (one for each word or subword unit), we need a transitional probability between models, such as N-gram models. However with this particular formulation, we would only be able to use bigram models.

Another shortcoming is that with this formulation, what we get is the most likely state sequence and not the most likely word sequence. Owing to this, several augmentations have been suggested to the basic Viterbi algorithm. For instance, Chow and Schwartz suggested an algorithm that provides the N-best hypotheses [204], while Murveit et al. created an algorithm that provides a lattice of words [169]. Another option for the task of decoding is the **A*** algorithm [97], which eliminates the weaknesses of the Viterbi algorithm [104], and which also spawned several modifications, such as the K-best Viterbi A* and the K-best Viterbi Iterative A* algorithms [94].

Another method for decoding HMMs that is of more interest to us here (as it is employed in the Kaldi toolkit [183] which was used for word recognition), is the **Weighted Finite State Transducer** (WFST) framework developed by Mohri et. al [165]. Weighted Finite State Transducers are Finite State Automata in which each transition in addition to having an input label, has an output label (potentially coming from a new alphabet), and a weight associated with it [163]. While WFSTs are applied for many tasks ranging from text to image processing, their principal application is in decoding HMMs [88]. Their success in this area is due to the approach offering "an elegant unified framework for representing the knowledge sources and producing a search network optimised up to the HMM state level" [11]. For this to work effectively, three important operators has to be implemented on WFSTs: composition, determinisation, and minimisation. The composition operator ($\circ$) permits us to combine transducers at different levels in such a way that the output of the first transducer becomes the input of the second [164]. This means that if we represent the context-dependent acoustic models (C), the context-independent lexicon (L), and the grammar model (G), creating the $C \circ L \circ G$ composition will result in a mapping of context-dependent phones to word sequences [11]. The result can be further optimised by determinisation (constructing an equivalent but deterministic WFST, meaning that each state of the new WFST has no more than one transition for any given input label, and there should be no empty input labels in the transducer [164]) and minimisation (minimising the number of states and transitions in the deterministic WFST) of the final network.

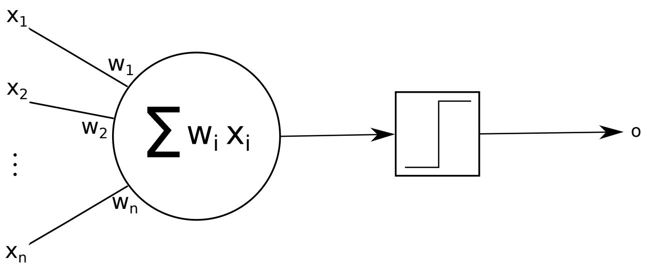

Figure 1.5: Diagram of the perceptron model.

## 1.4.2 Artificial Neural Networks

There are many different models that can be used in the construction of Artificial Neural Networks (ANNs) [221]. The simplest one is the perceptron model. Here, our neuron (see Figure 1.5) takes the weighted $(w_1, ..., w_n)$ sum of its inputs $(x_1, ..., x_n)$ and the output $(o)$ is determined by whether the given sum exceeds a predefined threshold value $(\theta)$ or not. This is similar to how neurons in nature (illustrated in Figure 1.6[1].) work: the neuron receives its input via the receptors spread over its dendrites [211], and whether the neuron is firing or not then depends on whether the potential of the neuron (changed by the summation of the stimuli on its dendrites) exceeds a given threshold [235]. In other words, the output of the natural neuron is regulated by the summation of its inputs.

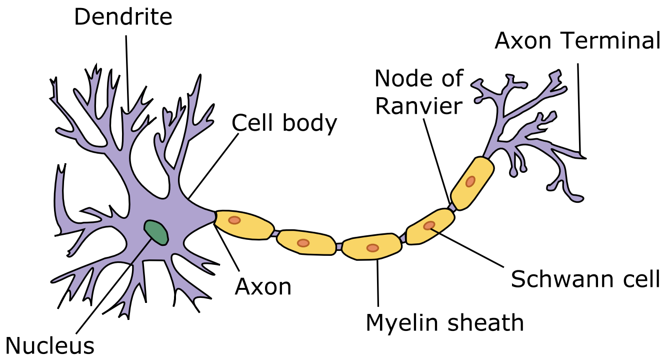

Figure 1.6: A simplified representation of a single neuron.

[1]Originally Neuron.jpg (public domain). Source: taken from "Anatomy and Physiology" by the U.S. National Cancer Institute's Surveillance, Epidemiology, and End Results (SEER) Program, redrawn by wikimedia commons user Dhp1080

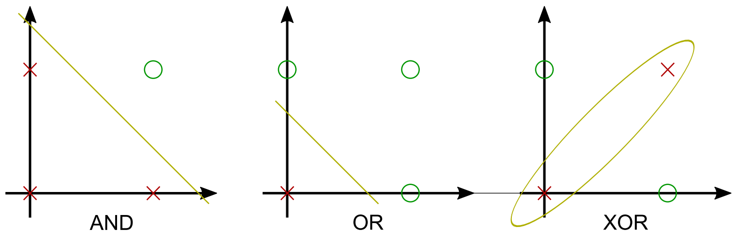

Figure 1.7: Linearly separable and linearly non-separable problems.

As in the perceptron model the information flows in one direction (from the input, through the neuron, to the output), it follows the feed forward model [207]. Using this model with the proper weights, it is possible to classify our input into two categories, given that these categories are linearly separable. Furthemore, it has also been shown that there not only exist proper settings for the weights that should define the hyperplane separating our classes, but there is a simple rule guaranteeing that we can find these settings [195].

The expectation of linear separability, however, is a strong constraint that only allows to solve a limited set of problems. For instance, while the logical function of AND and OR are linearly separable, this cannot be said about another basic function, XOR (see Figure 1.7). It has also been shown however, that by introducing a large enough layer of hidden peceptron-like neurons (here hidden means that these neurons are not directly connected to the environment, hence they are hidden from it – see Figure 1.8) creating an internal representation of the input, we can always find a mapping from the input to the expected output [158].

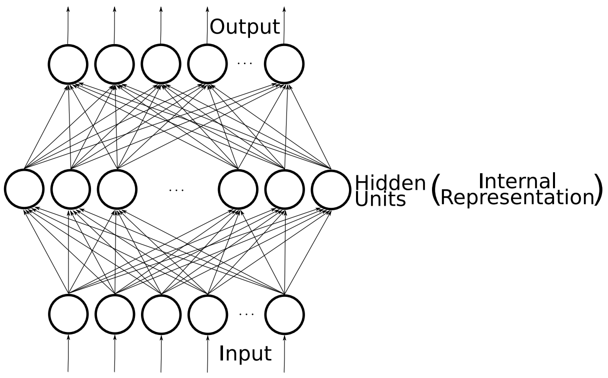

Figure 1.8: Artificial Neural Net with a hidden layer.

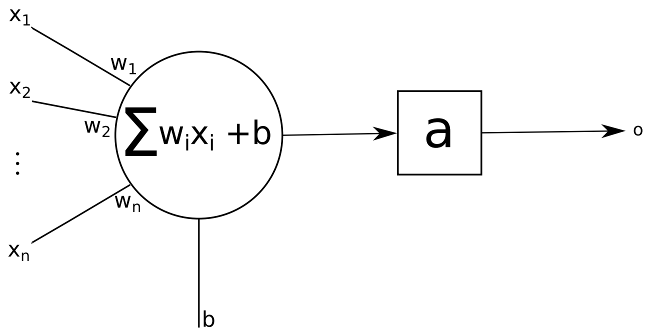

Figure 1.9: Representation of a hidden unit.

This particular architecture was used in the early experiments performed here. Here, the hidden unit (shown in Figure 1.9) is similar to the perceptron model we saw above, the only differences being the introduction of a bias $b$, and more importantly an activation function $a$. Now, the output $o$ of the neuron is found by using the formula:

$$o = a\left(\sum_{i=1}^{L} x_i \cdot w_i + b\right), \tag{1.13}$$

where $L$ is the number of weights. Here, various functions can be plugged in and used as the activation function $a(x)$ [63] in the neurons of the hidden layer

$$a(x) = \begin{cases} sig(x) = \frac{1}{1+e^{-x}} \\ tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \\ rectifier(x) = max(0,x) \end{cases} \tag{1.14}$$

Each has their respective advantages and disadvantages, which will be discussed later in Section 3.3. The output neurons however, use the softmax function, ensuring that the output values all lie between zero and one, and sum to one, providing us with the posterior-probability estimations we can use in the hybrid model [191]. There is still one issue, yet to be discussed, namely the training of our model. While the introduction of the hidden layer means an increased modelling power of the net, it also means that there is no simple, guaranteed learning rule we can use [196]. In our experiments for this purpose the backpropagation algorithm was used based on cross entropy minimisation [65].

### 1.4.3 The HMM/ANN Hybrid Model

So far we have discussed HMMs and ANNs, but not how the two fit together. For this let us go back to Section 1.4.1, where we described the $B$ set of emissional probabilities as a collection of probabilities that for each $o_t \in \Sigma$ observation and for each $q_i \in Q$ gives the probability of the observation being emitted by the $q_i$ state ($b_i(o_t)$). In other words, the probability of emitting the $o_t$ observation, given that we are in the $q_i$ state, is

$$b_i(o_t) = P(o_t|q_i) \tag{1.15}$$

But we did not discuss where these probabilities come from. It has been shown that ANNs can be used for posterior probability estimation [18], meaning that for an $x$ input and $c_i$ i-th class, the output of the right neuron in the output layer will provide an estimate for the probability of $x$ coming from the $c_i$ class

$$o(i) = P(c_i|x) \tag{1.16}$$

Thus if we train our neural net using $q_i$ states as class labels with $o_t$ observations as input,

$$o(i) = P(q_i|o_t) \tag{1.17}$$

It is quite apparent that there is a strong connection between equations 1.15 and 1.17. To exploit this, let us once again use the Bayes-rule, and reformulate Equation 1.15

$$b_i(o_t) = P(o_t|q_i) = \frac{P(o_t|q_i)P(q_i)}{P(o_t)} \tag{1.18}$$

After reordering this equation, we get

$$\frac{P(o_t|q_i)}{P(o_t)} = \frac{P(q_i|o_t)}{P(q_i)} \tag{1.19}$$

On the right hand side of this equation we see a scaled likelihood estimate [191], which we can calculate by dividing each output of the neural net by the prior probability of the corresponding state. Meaning, that even though we are not able to get the required likelihood on account of not having an estimate for the prior probability of the observation, we can use the neural net to calculate a scaled likelihood. It should be added here that as in experiments carried out by our group (HAS-RGAI), dividing by the prior probabilities had no demonstrable advantage in phone recognition tasks, we only used the prior probabilities in the word recognition experiments, while in the case of phone recognition, the HMM had probability estimates that directly came from the neural net.

## 1.5 Language Modelling

During the study, most efforts were focused on the feature extraction and the acoustic modelling phases. In language modelling, similar to the preprocessing of speech signal, preexisting resources were used on an as-is basis. However, it would be remiss on our part not to devote a few paragraphs on these methods as they were still instrumental in obtaining the results discussed in this study.

### 1.5.1 N-gram models

Although there are methods for language modelling that incorporate more linguistic knowledge [40, 98, 105], and methods with more advanced statistical modelling (like neural net-based language models [10, 156, 205]), due to their simplicity and good modelling performance, N-gram language models are the most widely used in state-of-the-art applications [10, 26].

Similar to what we saw in Section 1.4.1, the N-gram model is based on the assumption that the probability of the next element in a sequence only depends on a limited number of preceding elements. Formally,

$$P(w_{t+1}|w_1w_2...w_t) \approx P(w_{t+1}|w_tw_{t-1}...w_{t-N+1}) \tag{1.20}$$

Using this approximation, and the chain rule of probability, we should calculate the probability of word sequence $W$ ($W = w_i, w_2, ..., w_n$) as

$$P(W) \approx P(w_n|w_{n-1}w_{n-2}...w_{n-N+1})P(w_{n-1}|w_{n-2}w_{n-3}...w_{n-N})...P(w_1) \tag{1.21}$$

Now the question is how to estimate these N-gram probabilities. One possible way is via Maximum Likelihood Estimation (MLE) based on a training corpus [104]. Here, for the sake of simplicity, let us assume that $N = 2$, and that we are working with bigram models. In that case, estimating the probability of a given $P(w_{t+1}|w_t)$ bigram should be based on the relative frequency of sequences. That is,

$$P(w_{t+1}|w_t) = \frac{count(w_{t+1}w_t)}{count(w_t)} \tag{1.22}$$

This however would mean that for bigrams that do not occur in the corpus the estimated probability would be zero. In other words, using this model we would not be able to handle rare occurrences that were not part of our training corpus.

The importance of this problem is demonstrated by the findings of Allison et al. who, upon examining different training corpora, found that in a one-and-a-half-billion word corpus, 30% of legal trigrams never occurred [8]. However, there are various methods available that can handle this issue. The simplest one is adding one to the count of each N-gram model, also known as add-one or Laplace smoothing [138]. In a similar alternative, the N-gram count is increased by a fractional $k$ between zero and one (add-k smoothing) [100]. Although their simplicity makes these methods seem an attractive choice, generally, they perform poorly in language modelling [56, 55]. Another possibility is to estimate the probability of higher-order N-grams using their lower-order counterparts. Methods built on this intuition include Katz smoothing [107], Church-Gale smoothing [37], Jelinek-Mercer smoothing [99], Kneser-Ney smoothing [33, 120], and more recently Stupid backoff [25]. Another approach we can use to decrease the sparsity of our data is to decrease the number of N-grams by aggregating words into classes [27]. This aggregation can be carried out automatically by means of clustering [27, 144] or by using pre-existing linguistic information [172].

This latter approach can be especially useful in morphologically rich languages as Hungarian [216]. However, as in our experiments with the Hungarian language the task to be undertaken was phone recognition, we used a simple bigram language model. We utilised a similar language model in our phone recognition experiments on the TIMIT database. As for the experiments performed on the Aurora-4 corpus, we applied the standard 5k trigram language model, distributed with the corpus.

### 1.5.2 Language Model Scaling

If we recall the formula introduced earlier for the combination of the acoustic model and the language model,

$$\hat{W} = argmax_{W \in \mathcal{L}} P(O|W)P(W), \tag{1.9 revisited}$$

it might seem that the combination of the two seems trivial. However, as the acoustic model underestimates the probabilities [48], the two components are not on the same scale. Owing to this, a language model scaling factor ($LMSF$) is used to scale down its likelihoods (in Kaldi, the acoustic model is scaled for the same purpose), meaning that Equation 1.8 should have the following form [104]:

$$\hat{W} = argmax_{W \in \mathcal{L}} P(O|W)P(W)^{LMSF}. \tag{1.23}$$

This scaling has the side-effect of the decoder having the preference for more, shorter words [104], and to counterbalance this phenomenon, another factor, called word insertion penalty needs to be introduced.

## 1.6 Structure of the dissertation

**Chapter 2 - Spectro-Temporal Feature Extraction:** Here, we examine two feature extraction methods (2D DCT and Gabor filters) more closely. First, we experiment with 2D DCT feature extraction, proposed modifications to the process, and compared the resulting features with MFCC, demonstrating that the results of the former compare favourably with those of the latter, especially in the case of noise contaminated speech. We also examined the combination of the two feature sets, and demonstrated the benefits of combination. After this we discussed Gabor filters, and look at different methods for selecting a set of such features for the task of automatic speech recognition. We show that despite some issues in feature selection, competitive results can be attained. The chapter is based on the following publications:

- Kovács, G., and Tóth, L. Localised spectro-temporal features for noise-robust speech recognition. In *Proceedings of the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI)* (2010), pp. 481–485.
- Kovács, G., and Tóth, L. Phone recognition experiments with 2D-DCT spectro-temporal features. In *Proceedings of the IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)* (2011), pp. 143–146.
- Kovács, G., Tóth, L., and Van Compernolle, D. Selection and enhancement of Gabor filters for automatic speech recognition. *International Journal of Speech Technology 18*, 1 (2015), 1–16.

**Chapter 3 - The Joint Training of Spectro-Temporal Features and Neural Nets:** Here, we introduce a framework that combines spectro-temporal feature extraction and neural net training. First, we demonstrate the viability of the concept, comparing its result to those obtained without combining these methods. We also show that the results of the framework compares favourably to those attained using filter sets created via automatic methods both in terms of recognition scores and cross-database performance. After this, we introduce various improvements to the framework, highlighting its flexibility and its potential to include new advances in neural networks. The chapter is based on the following publications:

- Kovács, G., and Tóth, L. The joint optimization of spectro-temporal features and neural net classifiers. In *Proceedings of the 16th International Conference on Text, Speech, and Dialogue (TSD)* (2013), Vol. 8082 of *Lecture Notes in Computer Science*, Springer, pp. 552–559.
- Kovács, G., Tóth, L., and Van Compernolle, D. Selection and enhancement of Gabor filters for automatic speech recognition. *International Journal of Speech Technology 18*, 1 (2015), 1–16.

- Kovács, G., and Tóth, L. Joint optimization of spectro-temporal features and Deep Neural Nets for robust automatic speech recognition. *Acta Cybernetica 22*, 1 (2015), 117–134.

**Chapter 4 - The Multi-Band Processing of Speech using Spectro-Temporal Features:** Here, we examine another method for noise robust speech recognition inspired by human speech processing, connected with spectro-temporal feature extraction; namely, multi-band speech processing. Taking advantage of the compatibility of the two methods we show how the multi-band approach can further increase the performance when using spectro-temporal features. We also integrate this approach into the joint training framework and demonstrate that using the resulting method we can achieve state-of-the-art word recognition error rates. The chapter is based on the following publications:

- Kovács, G., Tóth, L., and Grósz, T. Robust multi-band ASR using Deep Neural Nets and spectro-temporal features. In *Proceedings of the International Conference on Speech and Computer (SPECOM)* (2014), Vol. 8773 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 386–393.
- Kovács, G., and Tóth, L. Multi-band noise robust speech recognition using Deep Neural Networks (in Hungarian). In *Proceedings of MSZNY* (2016), pp. 287–294.

**Chapter 5 - Band dropout:** Here, inspired by multi-band processing and dropout, we examine input-dropout in the joint training framework. For this, however we first redefine the settings of the framework, demonstrating that it is capable of providing competitive results as it is. We then examine our band dropout method that is related to multi-band processing, input dropout and data augmentation as well. The results tell us that applying this method leads to an improvement on our earlier results, and when combined with the ARMA technique, it can provide state-of-the art accuracy scores. The chapter is based on the following publications:

- Kovács, G., and Tóth, L. Optimisation of a spectro-temporal feature selection method integrated in Deep Neural Networks (in Hungarian). In *Proceedings of MSZNY* (2017), pp. 158–169.
- Kovács, G., Tóth, L., Van Compernolle, D., and Ganapathy, S. Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout. *Pattern Recognition Letters (2017), http://dx.doi.org/10.1016/j.patrec.2017.09.023*

# Chapter 2

# Spectro-Temporal Feature Extraction

*In speech recognition, there has been a trend to incorporate more and more knowledge about human hearing into the feature extraction step. One such approach is the application of neurophysiologically inspired localised spectro-temporal analysis in secondary feature extraction. In this chapter we examine two such techniques, namely the two-dimensional Discrete Cosine Transform (2D DCT), and the application of Gabor filters.*

*First, we will briefly discuss the 2D DCT method, and the results of our experiments conducted on the TIMIT database using this method. We will examine the phone classification and recognition performance of the 2D DCT feature extraction method performed on the conventional critical-band log-energy representation (log mel-spectrum). We will compare the results of this setup with the results obtained using Mel-Frequency Cepstral Coefficients (MFCC) on both clean, and noise corrupted speech. We will also attempt to combine the two feature sets to further improve the phone recognition accuracy scores obtained.*

*In the second part we will discuss spectro-temporal feature extraction (by means of processsing with Gabor filters). We will set up a simple filter set based on the 2D DCT method, investigate its performance on various speech databases, and compare these results with those obtained using MFCC features and with filter sets created by means of feature selection methods. Based on these results, we will also discuss the potential pitfalls of automatic feature selection, and suggest an alternative method in the following chapter.*

## 2.1 Introduction

We know more and more about human speech processing, but traditional feature extraction methods used in speech recognition take into account only the most fundamental properties of articulation and hearing. For example, the most commonly applied acoustic representation, the so-called Mel-Frequency Cepstral Coefficient (MFCC) [92], smooths out the fine details of the spectrum, as it is known that phonetic information is carried mainly by the spectral envelope. Also, it warps the linear frequency scale to the quasi-logarithmic mel-scale, which is known to fit human hearing better. But in other respects it is just a mathematical tool based on conventional signal processing algorithms. Although it is not strictly necessary that processing methods which seek inspiration from HSR should outperform the purely mathematical algorithms, in general it seems reasonable to expect a better behaviour from the methods that approximate the properties of human hearing more closely. One such property speech processing may benefit from approximating is the joint spectro-temporal sensitivity of the receptive fields of cortical cells [36]. Compared to what is known about the time-frequency tuning of these cells, the resolution of the conventional MFCC representation is much narrower in time and much wider in frequency.

As we saw in the last chapter, MFCC values are extracted from small, 20-30 millisecond pieces of the speech signal using conventional signal processing algorithms like the Fourier transform and the cosine transform [92]. This method of processing the speech signal in uniform 20-30 millisecond chunks has its roots in the speech coding tradition, and is retained mostly for technical convenience. Humans can barely recognise such short speech excerpts, which suggests that they are not an optimal choice for the basic unit of classification. Although the $\Delta$ and $\Delta\Delta$ coefficients (which are also part of the classic feature set of speech recognition in addition to the basic MFCC features) capture pieces of information from the neighbouring 4-4 frames (4 frames preceding and 4 frames following the current frame), both physiological and psychoacoustic experimental results indicate that the human brain extracts information from much longer time spans. Technically the simplest solution for this is to work with larger windows along the time axis: in neural-net based recognisers it is now standard practice to train the system on 9 or more neighbouring MFCC vectors [22, 110, 181]. There are still problems with these techniques, however. One is that although it is possible to capture more temporal information using this method, it is still different from having features tuned to specific temporal modulations [24]. Moreover, with the increasing number of neighbouring frames used, the number of features is growing significantly, and hence the application of dimension reduction methods such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) may be necessary.

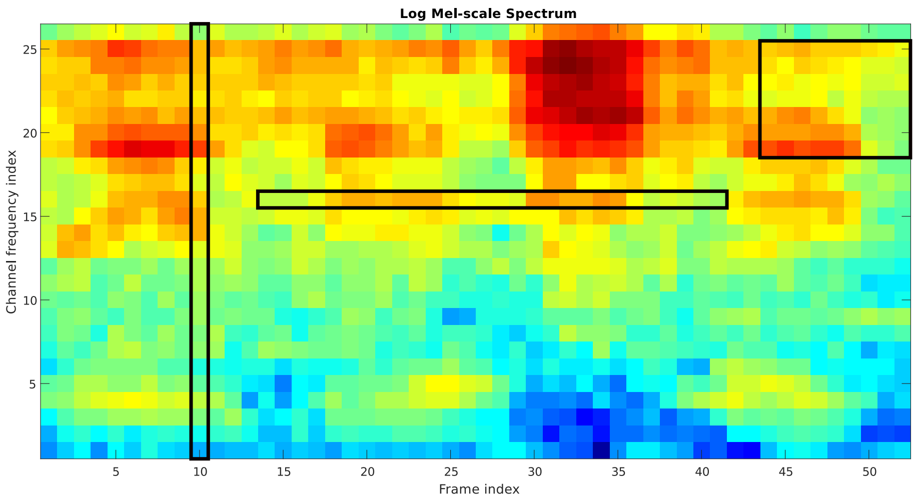

Figure 2.1: A short part of the log mel-spectral representation of the sentence, "The company previously traded over the counter" from the clean test set of the Aurora-4 database. The black boxes (left-to-right) show the shape of the feature extraction patches used by a) the classic MFCCs, b) the TRAP features, and c) localised spectro-temporal features.

Another problem with using multiple neighbouring MFCC vectors in training is that the resulting features are still global along the frequency axis. However, there is a purely practical argument against this: when the signal is corrupted with band-limited noise, a spectrally global analysis technique such as the MFCC will result in all the features being contaminated by the noise [24]. Beyond practical considerations, there are also physiological and psychoacoustic experiments which suggest that "human speech perception is based on relatively narrow frequency channels" [159]. This suggests that the windows should be localised in frequency as well. These studies motivated the introduction of the TRAP model by Hermansky et al. In this scheme each frequency band is processed separately, and the corresponding results are combined only at a later stage; the time-span of the processed trajectory patterns even goes up to 1 second in certain experiments [81]. This setup can be interpreted as a sort of "inverse" arrangement compared to the classic MFCC windows. We illustrate this approach along with the approach of MFCC in Figure 2.1 above. In the same figure there is also a visualisation of the approach followed in the experiments. This approach processes the spectro-temporal representation in patches that are localised in both time and frequency. The range of this approach is larger in the time domain, and smaller in the frequency domain than that of MFCC's.

Before proceeding with a discussion of specific methods for extracting localised spectro-temporal features, let us formalise the general process. This approach takes localised patches from the spectro-temporal representation of the speech signal, and creates features for ASR purposes by processing them using standard filtering methods. We can interpret this process as applying a spectro-temporal filter $F$ on a patch $P$, to obtain an output $o$ defined by the following formula:

$$o = \sum_{f=0}^{M-1} \sum_{t=0}^{N-1} P(f,t)F(f,t), \tag{2.1}$$

where $M$ and $N$ are the respective height and width of filter $F$ and patch $P$. Here, patch $P$ is extracted from a wider spectro-temporal representation in such a way that its size is the same as that of filter $F$. And $P(f,t)$ is referring to positions in this extracted patch (meaning that offseting will not be neccessary). One can get a set of features by using several filters with different coefficients, and/or applying the same filters at different positions in the time-frequency plane. A more difficult task than the calculation of spectro-temporal features is arriving at the correct family of filters that are suitable for the given task. In Section 1.3.2 we outlined two methods employed in spectro-temporal feature extraction. Below, we will examine and describe two additional methods.

## 2.2 2D DCT

An obvious generalisation of the MFCC feature extraction to localised spectro-temporal windows is to replace the DCT by its two-dimensional counterpart, namely the 2D DCT. The 2D DCT is obtained by applying a transform on a patch $P$ that is represented by an $M \times N$ matrix defined by the following formula

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} P_{mn} cos \frac{\pi(2m+1)p}{2M} cos \frac{pi(2n+1)q}{2N}, \tag{2.2}$$

where $0 \leq p \leq M-1$ and $0 \leq q \leq N-1$, the values of $\alpha_p$ and $\alpha_q$ are given by the following:

$$\alpha_p = \begin{cases} 1/\sqrt{M}, p = 0 \\ \sqrt{2/M}, 1 \leq p \leq M-1 \end{cases} \tag{2.3}$$

$$\alpha_q = \begin{cases} 1/\sqrt{N}, p = 0 \\ \sqrt{2/N}, 1 \leq p \leq N-1 \end{cases} \tag{2.4}$$

The resulting $B_{pq}$ values are the 2D DCT coefficients of patch $P$ [219]. These values correspond to the result of performing the filtering defined by Eq. (2.1) with the following filter coefficients:

$$F_{pq}(f,t) = \cos\frac{\pi \cdot (2f+1)\cdot p}{2M}\cos\frac{\pi \cdot (2t+1)\cdot q}{2N}, \tag{2.5}$$

where $M$ and $N$ are the respective height and width of the filters for $f$ and $t$, while $p$ and $q$ specify the modulation frequencies of the filter along the frequency and time axis. By definition, for a patch of size $M \times N$, a 2D DCT returns the same number of coefficients (the filters corresponding to these coefficients are shown in Figure 2.2 for the $N = M = 7$ case).

Several findings are available which indicate that these coefficients are not equally important for representing the underlying acoustic content. As regards the frequency axis, the good performance of MFCC clearly demonstrates that it is sufficient to keep the low-order coeffcents. Similar results regarding time-domain modulations can be found in the speech recognition literature [106]. The image processing field has also for a long time been making use of the notion that higher-order coefficients are less important than lower-order ones in image compression based on 2D DCT [35].

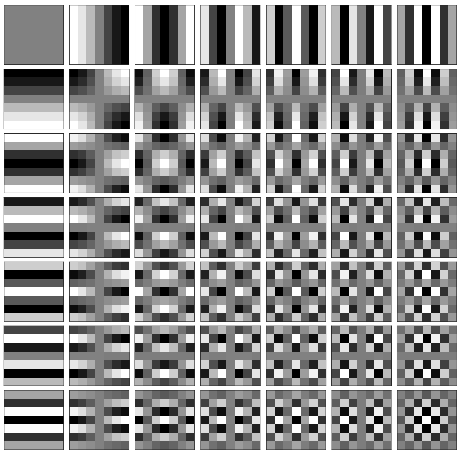

Figure 2.2: Basis functions of the 2D DCT for 7 by 7 matrices.

### 2.2.1 Phone classification experiments

While above we provided reasons about why we have chosen to keep the low-order coefficients of the 2D DCT as features, it is not obvious how many of these coefficients should be retained. The question concerning the size of the patches used for extracting these features also has to be resolved. Although we could rely on the studies of human perception in determining the patch size, in machine learning experiments different values may result in optimal recognition performance due to the various properties and peculiarities of the signal processing and machine learning algorithms applied. Thus the best we can do is to vary the sizes and look for the optimal parameters by empirical means. We did so by performing phone classification experiments on the TIMIT speech database [129], following the earlier study of Bouvrie et al. [24].

The phone classification experiments carried out during the evaluation consisted of the following steps:

1. Obtaining a spectral representation of the speech data
2. Extracting the time-frequency patches
3. Producing feature vectors of equal length corresponding to each phone
4. Classifying the feature vectors using ANNs

#### Obtaining spectral representation

Besides using log mel-spectrums with different number of channels, in the first set of experiments we also used conventional spectrograms. We did so, because one of our goals here was to reproduce the earlier results of Bouvrie et al. [24], who had been using conventional spectrograms as their spectral representation in their experiments on spectro-temporal speech processing. The spectrograms were obtained using the following parameters: in the framing and windowing stage, we applied Hamming windows with an offset of 32 samples, and each frame was Fourier-transformed using a 1024-point FFT. We tried two configurations for the frame size, namely 300 samples for the narrow- and 150 for the wide-band cases. Then the log-magnitude of the resulting spectrum was taken, and it was normalised so as to have unit variance and a zero mean for each utterance.

### Extraction of patches

The next step was the extraction of the time-frequency patches. This was carried out by computing a sliding localised two-dimensional DCT over the spectrogram. The window and step sizes were again chosen based on the suggestions in [24] (with a slight modification – substituting even window sizes with odd ones – for symmetry reasons). Namely, 51 by 21 bin windows were applied in the narrow band case, and 41 by 51 bin windows were applied in the wide-band case (always giving the height first). The step sizes were 25 bins for the frequency, and 2 bins for the time axis in both cases. Unlike the authors in [24], we tested the 51 by 21 bin window size not only with the narrow-band spectral representation, but also with the wide-band spectral representation.

By default, the DCT returns the same number of coefficients as the size of its input array. Similar to the 1D case (that is, the computation of MFCC), one can throw away the coefficients which correspond to higher modulation frequencies. With this step we smooth out the unnecessary fine details from the spectrum and reduce feature dimensionality at the same time. Bouvrie et al. proposed keeping just the 6 lowest-order 2D DCT coefficients corresponding to the upper left $3 \times 3$ triangle of the coefficient matrix [24]. Apart from this configuration, in certain cases we also experimented with retaining more (9 or 15) coefficients here.

### Producing feature vectors

Next, for the phone classification experiments we had to create a fixed-length feature vector from the variable number of 2D DCT coefficients extracted from the patches belonging to each phonetic segment. For this purpose we followed the technique proposed by Halberstadt [73], which was also found to work well by other authors [38, 122]. Each phonetic segment was divided into three parts along the time axis, and the coefficients belonging to the same patch index and DCT coefficient index were averaged over time within these segments. Two more segments were composed from the 30 milliseconds of the signal before, and from the 30 milliseconds of the signal after the segments, and were processed in a similar way. This technique yields a pooled feature vector that consists of the same number of components for each segment – five times the number of patches along the frequency axis and the number of DCT coefficients retained – independent of the segment duration. Afterwards, the segment duration values were also appended to this segmental feature vector.

Table 2.1: Phone classification error rates using a spectrogram.

| Spectral resolution | 2D DCT patch size | Patch step (vertical) | Patch step (horizontal) | No. DCT coefficients | No. features features | Error rate |
|---|---|---|---|---|---|---|
| Narrow-band | $51 \times 21$ | 25 | 2 | 6 | 511 | 20.53% |
| Wide-band | $51 \times 21$ | 25 | 2 | 6 | 511 | 20.14% |
| Wide-band | $41 \times 51$ | 25 | 2 | 6 | 511 | 20.66% |
| MFCC + $\Delta$ + $\Delta\Delta$ | | | | | 196 | 20.30% |

### Artificial Neural Net

In all the experiments, a multi-layer perceptron neural net [17] was applied as a classifier. It contained one hidden layer of 500 neurons, and the output layer applied the softmax nonlinearity, while the hidden neurons used the sigmoid function. The number of output neurons was set to the number of classes (39), while the number of inputs naturally varied, as we will describe later. The neural net was trained using standard backpropagation on 90% of the training data in semi-batch mode, and validation on the remaining 10% (selected randomly) was used in both the stopping criterion and in the learn-rate scheduler.

### Experiments on clean speech

The results obtained by extracting 2D DCT coefficients from various patch sizes applied on the conventional spectrogram are shown in Table 2.1. As a baseline result, the score got with the conventional MFCC features is also listed. MFCC features were extracted with the default parameter values given in Section 1.3: 13 mel-cepstral coefficients were calculated over 25 ms time frames every 10 ms, and the vectors were augmented with the $\Delta$ and $\Delta\Delta$ coefficients. On the resulting 39-dimensional MFCC feature vectors the same averaging method was applied to get a segmental feature vector of fixed size. Each phonetic segment was represented by 196 features (including the duration).

As the results show, the proposed features are quite insensitive to the exact parameter values (resolution and patch size). All three scores are similar to the MFCC result, and the best one even slightly outperforms it. Because of the big patch sizes and the larger number of features, however, the extraction of the 2D DCT features takes much longer than that of the conventional MFCCs. It should be mentioned here that the reported results are consistently better than those presented in [24], both with the conventional and the localised spectro-temporal features. We attribute this to the fact that though the processing of the patches was similar, a different type of classifier was applied on the resulting feature vectors.

Table 2.2: Phone classification error rates with the critical-band energy map.

| Spectral resolution | No. channels | 2D DCT patch size | Patch step (vertical) | Patch step (horizontal) | No. DCT coefficients | No. features features | Error rate |
|---|---|---|---|---|---|---|---|
| Wide-band | 105 | $17 \times 21$ | 6 | 2 | 6 | 511 | 19.55% |
| Wide-band | 104 | $15 \times 21$ | 6 | 2 | 6 | 511 | 19.90% |

The resolution of the spectrogram used by Bouvrie et al. is much higher than that usually applied in speech recognition, both in time and frequency. Lowering the resolution of our spectral representation not only yields a reduction in computational cost, but also makes our primary features more like those used in standard secondary features, which means that it is easier to make the case that the difference in results is due to the difference in secondary feature extraction. Moreover, nowadays it is widely accepted that the mel-warping of the frequency scale is useful for recognition, and so feature extraction methods that work on the linear frequency scale have mostly been abandoned. Motivated by these points, we decided to repeat the experiments on the same critical-band energy spectral representation that is used as the starting point of the MFCC computation (see Section 1.3.1). Fortunately, the HCopy module of HTK [236] can be parametrised so that it calculates just the critical-band spectrum and skips the final step, namely the DCT computation. This way it could be ensured that we worked on exactly the same spectral representation as the one from which the baseline MFCCs were obtained.

In the first pilot studies here, the window and step size of the spectral computation were adjusted so that they would agree with the wide-band resolution used in the spectrogram-based experiments. That is, the resolution of the spectrum was not decreased, but the mel-scale frequency axis warping was activated. More precisely, 104-105 frequency bands were extracted from the spectrogram (the number was varied slightly in order to support the full coverage of the spectral bands by the patches). This results in a smaller spectral map height than that of the spectrogram, so the patch size and the patch hop along the frequency axis were proportionally decreased to 15-17 and 6, respectively. The error rates got with two different patch sizes are shown in Table 2.2. As can be seen, both scores are better than the earlier results, indicating that we were justified in substituting the spectrogram with the mel-scale spectrum.

The next step was to reduce the spectral resolution so that it was equivalent to that used for the MFCC computation. Applying the default MFCC settings, 100 critical-band energy vectors were extracted per second from 25 ms frames. As the number of critical-band frequency channels was set to 26 during the MFCC extraction, in the experiments the number of channels was also set to 26 or approximations of its multiples/divisors (the number of channels used

Table 2.3: Phone classification error rates with the critical-band energy map.

| No. channels | 2D DCT patch size | Patch step (vertical) | Patch step (horizontal) | No. DCT coefficients | No. features features | Error rate |
|---|---|---|---|---|---|---|
| 105 | $17 \times 5$ | 6 | 1 | 6 | 511 | 20.71% |
| 105 | $17 \times 7$ | 6 | 1 | 6 | 511 | 20.65% |
| 105 | $17 \times 9$ | 6 | 1 | 6 | 511 | 20.31% |
| 105 | $17 \times 11$ | 6 | 1 | 6 | 511 | 21.01% |
| 50 | $9 \times 9$ | 3 | 1 | 6 | 481 | 20.60% |
| 50 | $15 \times 9$ | 3 | 1 | 6 | 451 | 21.01% |
| 52 | $7 \times 7$ | 3 | 1 | 6 | 511 | 20.53% |
| 52 | $7 \times 9$ | 3 | 1 | 6 | 511 | 20.77% |
| 53 | $9 \times 7$ | 3 | 1 | 6 | 511 | 20.22% |
| 53 | $9 \times 9$ | 3 | 1 | 6 | 511 | 20.32% |
| 26 | $5 \times 9$ | 2 | 1 | 6 | 361 | 20.43% |
| 26 | $5 \times 15$ | 2 | 1 | 6 | 361 | 22.45% |
| 26 | $7 \times 9$ | 2 | 1 | 6 | 361 | 20.27% |
| 26 | $7 \times 15$ | 2 | 1 | 6 | 361 | 22.39% |
| 26 | $5 \times 9$ | 2 | 1 | 9 | 541 | 20.04% |
| 26 | $7 \times 9$ | 2 | 1 | 9 | 541 | 19.88% |
| 14 | $7 \times 9$ | 2 | 1 | 9 | 271 | 20.66% |
| 14 | $7 \times 9$ | 2 | 1 | 15 | 451 | 19.73% |
| 13 | $5 \times 9$ | 2 | 1 | 6 | 181 | 21.87% |
| 13 | $5 \times 9$ | 2 | 1 | 9 | 271 | 20.37% |
| 13 | $5 \times 9$ | 2 | 1 | 15 | 451 | 19.79% |

were 104, 52, 26 and 13 – again 1-2 channels were sometimes added to make the patches fit the full range). As both the time and frequency resolution of the spectral representation became much smaller than that of the spectrogram used in our earlier experiments, the size of the time-frequency patches was also shrunk proportionally: for example, to 5 by 9 bins for the default 26 frequency channel case. The patch step size was 2 along the frequency axis and 1 along the time axis here, but of course, the proper step size again depends on the number of channels.

We experimented with various combinations of the number of channels, patch size and patch step; and the results got with the various parameter settings are presented in Table 2.3. As can be seen, the best scores were again slightly better than the results got using a spectrogram. Apart from some 15-long patches that performed significantly worse, the results are quite similar, independent of the actual settings. Hence a key finding of these experiments was that the resolution of the input spectrum and the size of the DCT patches can be reduced, hence the computational costs can be reduced without losing any recognition accuracy. Also, conventional tools such as the HCopy routine of HTK can be used for the critical-band log-energy extraction step.

Table 2.4: Phone classification error rates with the critical-band energy map.

| Feature set | Clean speech | Pink noise 20 dB | Pink noise 10 dB | Pink noise 0 dB | Babble noise 20 dB | Babble noise 10 dB | Babble noise 0 dB |
|---|---|---|---|---|---|---|---|
| 2D DCT on wide-band spectrogram 41 × 51 patches, 6 coeffs | 20.66% | 31.15% | 49.74% | 72.70% | 31.03% | 52.59% | 76.04% |
| 2D DCT on critical bands 53 chans, 9 × 7 patches, 6 coeffs | 20.22% | 37.83% | 60.35% | 80.30% | 30.93% | 51.69% | 76.84% |
| 2D DCT on critical bands 26 chans, 7 × 9 patches, 9 coeffs | 19.88% | 35.65% | 58.86% | 78.69% | 33.05% | 52.59% | 75.82% |
| 2D DCT on critical bands 13 chans, 7 × 9 patches, 15 coeffs | 19.79% | 34.83% | 57.75% | 77.17% | 35.06% | 55.43% | 77.25% |
| MFCC + $\Delta$ + $\Delta\Delta$ | 20.30% | 36.76% | 63.00% | 80.13% | 31.58% | 55.23% | 77.46% |

### Experiments on noise contaminated speech

To test the assumption that localised spectro-temporal features should be more robust to noise, we artificially contaminated the test dataset with pink and babble noise. The amplitude of the noise was tuned so as to get a signal-to-noise ratio of 20, 10 and 0 decibels in three different experimental settings. It should be mentioned here that again in all the noisy experiments, the training was performed on the clean data and the noisy data was only used for testing purposes.

For the noisy tests, three configurations used in the clean data experiments with 53, 26 and 13 critical bands were chosen. Naturally, the MFCC tests were also repeated so as to have a comparative baseline. The results got are shown in Table 2.4. As can be seen, in the case of speech contaminated with pink noise, the 2D DCT features yielded lower error rates than those of the MFCCs in almost every case, especially with higher levels of noise. It also turned out, however, that the feature set extracted from the spectrogram behaved much better than the critical bands-based set. The reason might simply be the unfortunate choice of the noise type: pink noise affects the lower frequencies more, while these frequencies are over-represented in the critical-band energy map compared to the linear frequency-scale spectrogram due to the mel-warped frequency scale of the former. To test this hypothesis we repeated the experiments with babble noise. This type of noise affects virtually the whole spectrum, so one might expect less gain from a localised representation. As the results show (see Table 2.4), the best 2D DCT scores are indeed slightly better than those got with MFCCs. One can also see, however, that in this case there was no significant difference between the performance of the spectrogram representation and the performance of two best configurations using the critical-band-based representation (53 channels and 26 channels).

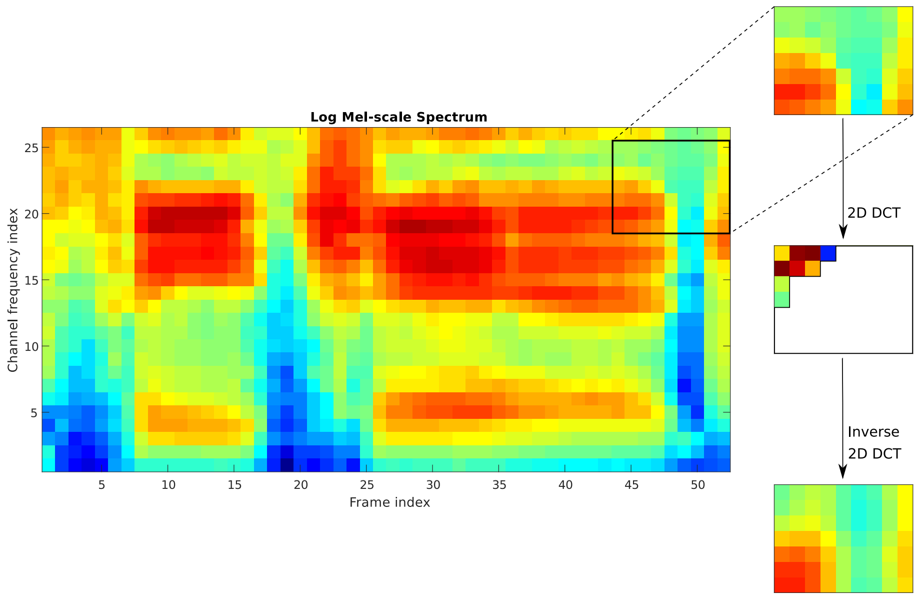

Figure 2.3: A short part of the log mel-spectral representation from the sentence, “Cut a small corner off each edge” from the train set of the TIMIT database. The boxes on the right show the degree to which a given patch can be reconstructed from the proposed coefficients using the inverse 2D DCT.

### 2.2.2 Phone recognition experiments

After examining the results of the phone classification experiments, in the other experiments we simply chose the settings where the spectral representation was created using 26 filterbanks (the same as in the MFCC), and where the window size was $7 \times 9$, and 9 2D DCT coefficients were retained as features (as shown in Figure 2.3).

Using these settings we carried out phone recognition experiments on the TIMIT database in the so-called HMM/ANN framework (see also Section 1.4.3). The neural net used here for producing frame-level likelihoods was the same as that used in the phone classification experiments, with the exception of the input layer (the size of this layer is always adjusted to the length of feature vectors), and the size of the hidden layer which was another parameter here. The algorithms necessary for building the HMM were provided by the HTK toolkit [236]. Once again, results will be reported on the full test set of TIMIT and its noise contaminated version.

Table 2.5: Frame classification error rates got on the clean test set.

| Feature set | No. of features | No. of hidden units | Frame error rate |
|---|---|---|---|
| MFCC + $\Delta$ + $\Delta\Delta$ | 39 | 500 | 34.77% |
| 9×(MFCC + $\Delta$ + $\Delta\Delta$) | 351 | 5000 | **27.87%** |
| 2D DCT | 108 | 1000 | 31.57% |
| 2D DCT + $\Delta$ + $\Delta\Delta$ | 324 | 5000 | **27.91%** |
| 5 ×(2D DCT + $\Delta$+$\Delta\Delta$) | 1620 | 5000 | 31.22% |

### Experiments on clean speech

As the neural net was trained to minimise the classification error of the frames, the frame-level error rate already gives a good indication of whether a feature set is good or not. Hence, first let us look at these results in Table 2.5. Similar to the previous experiments, the MFCC coefficients served as a baseline here as well. One baseline result was obtained using the 39 MFCC + $\Delta$ + $\Delta\Delta$ coefficients and with 5000 hidden neurons (first row). The result for the basic 2D DCT feature set is shown in the third row. Note that the 2D DCT feature vector is much bigger than the MFCC vector, and this is why we allowed more hidden neurons for this configuration. In general, as the number of features increased, we tried to increase the number of neurons in the same proportion, but this has the usual drawbacks, namely the training and evaluation times become longer, and the risk of overfitting increases.

With the MFCC feature set it is standard practice to use several neighbouring frames instead of just one. The result shown in the second row, obtained using 9 frames, clearly justifies this. However, the number of features dramatically increases, and the size of the net also had to be adjusted. As the MFCC set includes the dynamic features, we tried to similarly extend the 2D DCT features with the $\Delta$ and $\Delta\Delta$ coefficients. The resulting frame error rate indicates that it was worth doing so, even at the price of a threefold increase in the feature vector size. Note that the dimensionality of one vector of this feature set is almost the same in total as the dimensionality of nine neighbouring vectors of the MFCC set. Fortunately, however, the corresponding error rates are also comparable, which tells us that the 2D DCT features are more efficient in extracting long-term information. As a next step, similar to the practice with MFCCs [22], we attempted to use several neighbouring frames of this features set as well. However, because of the already high dimensionality of the input, only 5 frames were included. And in an attempt to limit the parameter count, the number of neurons was left as it was. This experiment did not bring about any further improvements, so in all the following tests the system briefly referred to as "MFCC-based" used the configuration shown in row 2, while the "2D DCT" system utilised the parameter values of row 4.

Table 2.6: Phone error rates (PER) got on the clean test set.

| Feature type | No language model | Phone bigram language model |
|---|---|---|
| MFCC-based | 29.11% | 27.13% |
| 2D DCT | 29.52% | 27.05% |

The frame-level phone probabilities were converted into phone strings using the HVite tool of HTK [236] with two settings. In the first case no language model was used at all, so that we could assess how the pure acoustic models built upon the two feature sets perform. In this configuration there was only one adjustable parameter, namely the phone insertion penalty, which was set to -3. In the second case the system was extended with a simple phone bigram language model with a weight of 1.0. The error rates listed in Table 2.6 were calculated from the accuracy scores of the recognition, which were got using the standard HTK evaluation tools and methodology. We can see that the system using the 2D DCT features performed slightly worse when only the acoustic model was employed, but when the language model was present – which is the normal situation in speech recognition – the results got with the new and the classic feature set were practically identical.

## Experiments on noise contaminated speech

In order to assess the noise robustness of our features, the phone recognition experiments were again repeated on noise contaminated versions of TIMIT's full test set. Here, we did not use a language model, and this allowed us to examine the performance of the acoustic features and models. The results are shown in Table 2.7 below. As can be seen, while the usage of MFCC features leads to a small improvement in phone recognition on the clean test set, on noise contaminated speech the 2D DCT features yield markedly lower error rates than those for the MFCCs. This is in accord with our earlier phone classification results.

Table 2.7: Phone error rates (PER) got on the clean and noise contaminated test sets.

| Feature | Clean | Pink noise | | | Babble noise | | |
|---|---|---|---|---|---|---|---|
| set | speech | 20 dB | 10 dB | 0 dB | 20 dB | 10 dB | 0 dB |
| MFCC-based | 29.11% | 56.78% | 74.78% | 85.57% | 48.04% | 73.56% | 86.29% |
| 2D DCT | 29.52% | 46.62% | 67.01% | 79.07% | 41.03% | 58.36% | 74.81% |

## Combining the feature sets

Apart from its superior performance in noisy condition, the new feature set might help the recognition process under clean conditions as well. Although tables 2.6 and 2.7 showed that under clean conditions 2D DCT features yielded a performance no better than the MFCCs when used alone, here we have the chance to combine the two feature streams. If the two feature sets perform poorly in different cases, there is a possibility that their combination might reduce the overall error rate. This is a popular research topic nowadays, and several methods have been proposed for combining feature streams (see, e.g. [137]). Here a simple technique is going to be used: the phone posteriors produced by the two neural nets trained on the two feature sets will be concatenated, and a third neural net will be trained on the resulting vector. If this second stage net is trained on several neighbouring frames — similar to the MFCC-based net — then it is able to correct some of the errors of the lower stage net(s) with the help of the long-term context. Hence, using such a second stage network can already be useful in itself, as was recently shown in [110] and [181]. Because of this, the earlier results will be compared to results of two 2-stage configurations: in one case the second stage network is trained only on the posteriors of the MFCC-based network, while in the other case in the second stage network we combine the MFCC-based and the 2D DCT based probabilities. This way, it should be possible to assess how much of the error rate reduction is due to the 2-stage approach, and how much of it is due to the combination of features. The neural net used in the second stage consisted of 1000 hidden neurons, and was trained using 9 neighbouring frames.

The recognition results obtained with a phone bigram language model are shown in Table 2.8. It can be seen here that the application of the 2-stage framework in itself leads to a relative error rate reduction of over 7%. This error rate reduction clearly shows the advantage of the 2-stage modelling technique. And when we compare the second and third rows of the table we can see that the simple combination strategy of the two feature sets can reduce the recognition error rate still further, yielding an overall relative error reduction of 10% compared to the original result.

Table 2.8: Phone error rates (PER) of the 2-stage system.

| Feature set | PER |
|---:|---:|
| 1-stage, MFCC-based | 27.13% |
| 2-stage, MFCC-based | 25.11% |
| 2-stage, MFCC & DCT Combination | 24.37% |

## 2.3 Gabor filters

It has been shown experimentally that Gabor filters, introduced in 1946 by Hungarian physicist Dénes Gábor [54], can be used to model the response profile of certain neurons [101, 187, 212]. This property made Gabor filters a popular feature extraction method for various audio and visual classification experiments [49, 74, 91, 115, 152, 201]. The popularity and success of the method inspired us to examine the use of Gabor filters in noise robust speech recognition. Gabor filters are commonly defined as the product of a complex sinusoid carrier and an envelope function. The exact definition, however, may vary slightly from author to author. Ezzat et al. [49] defines Gabor filters as a product of a two-dimensional Gaussian (2.6) and an oriented sinusoid (2.7). That is,

$$W(f,t) = \frac{1}{2\pi\sigma_f\sigma_t} e^{-\frac{1}{2}\left(\frac{(f-f_0)^2}{\sigma_f^2} + \frac{(t-t_0)^2}{\sigma_t^2}\right)} \tag{2.6}$$

$$S_{\Omega,\omega}(f,t) = e^{i2\pi\left(\frac{\Omega}{M}f + \frac{\omega}{N}t\right)}, \tag{2.7}$$

where $f$ and $t$ iterate over the frequency and time span of the window, and $\sigma_f^2$, $\sigma_t^2$ specify their respective bandwiths. $M$ and $N$ specify the transform size, while $\Omega$ and $\omega$ specify the slanting and the periodicity of the sinusoid. Figure 2.4 shows this effect by displaying the real part of three different Gabor filters of the same size created with three different $\Omega$ and $\omega$ parameter pairs.

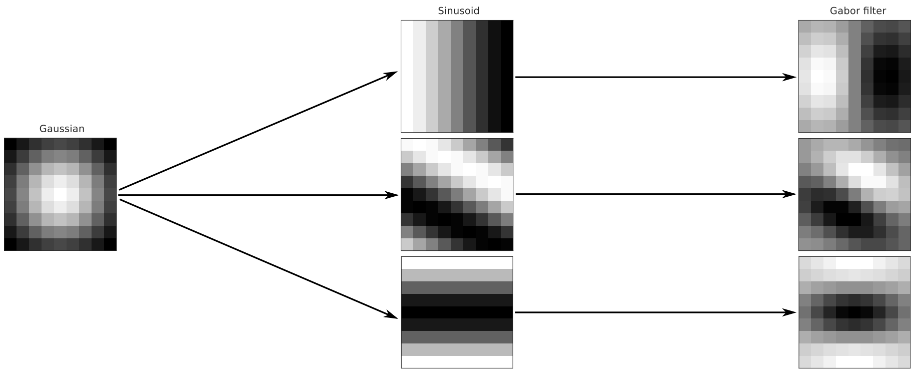

Figure 2.4: Examples of Gabor filters with a height and width of 9 units, with the remaining parameters being $\Omega = [0.0, 0.9, 0.9]$ and $\omega = [0.45, 0.45, 0.0]$, respectively (from top to bottom).

As we only use the real part of the Gabor filter, Eq. (2.7) can be rewritten in the following form:

$$S_{\Omega,\omega}(f,t) = cos\left(\frac{\pi \cdot f \cdot 2\Omega}{M} + \frac{\pi \cdot t \cdot 2\omega}{N}\right) \tag{2.8}$$

At this point we should note the similarity between equations (2.8) and (2.5) with the right assignment of their parameters. Later on this similarity will be exploited in the construction of a filter set. It should also be noted that while it is clear that with the proper adjustment of the $\Omega$ and $\omega$ parameters, we can control the shape of the Gabor filters, it is not so obvious how these parameters should be selected and how many filters should be used for optimal speech recognition performance.

This is a problem that has to be addressed with every feature extraction method: the selection of a reasonably small set of relevant features (i.e. filters) got from the huge variety permitted by the parametrisation process itself. To better understand the magnitude of the search space, let us consider that even if we fix parameters $M$ and $N$ of Eq. (2.8), as the $\Omega$ and $\omega$ parameters are continuous, they define an infinite space of possible filters. And while in the case of 2D DCT there are some assumptions about which features should be kept [24], with Gabor filters we have less a priori information. Thus the goal of selecting a reasonably small number (i.e. at most a couple of hundred) from among these filters in such a way that the selected filters provide useful features for speech recognition purposes is far from trivial. Here, we will discuss methods that seek to solve this problem by means of feature selection methods. That is, all the filters of the search space are systematically evaluated, resulting in a huge feature space of possible acoustic features. Then, the Gabor filters are selected based on the speech recognition performance of their corresponding features. The task then is to find a proper feature selection method, to which a major part of research in Gabor filters is devoted. Here, several points need to be considered. Namely,

1. Given the variety of spectral and temporal properties of human speech recognition, it is expected that a huge search space needs to be examined. One problem is that finding a global optimum in this vast search space can only be guaranteed by evaluating all the feature subsets, which is clearly impossible in this case. Also, as we intended to measure the quality of the feature set candidates by evaluating their usefulness on a speech processing task, the runtime of the feature selection algorithms was a key concern. This means that a very good heuristic must be applied during the search process.

2. The aim of filter selection here should be to create a simple, parametrised filter family that detects all relevant speech phenomena, and which is hopefully independent of the actual training database. That is, even when the filter set is optimised on a given speech corpus, it would be preferable to get a filter set that gives a nice recognition performance on different databases as well.

3. These filters do not constitute an orthogonal representation of the signal and lead to highly correlated features. The correlation depends on the similarity of the filter parameters, but it is hard to quantify. It can be expected that the correlation of filters will make the search task even more difficult than before.

Reviewing the literature of Gabor filters, we found that most authors apply automatic feature selection methods to find a proper subset of filters. These automatic methods are all built on machine learning principles [49, 117, 152, 215, 227]. Unfortunately, these feature selection algorithms are very slow and are based on a greedy search that may yield suboptimal solutions. And while there are tasks where the greedy strategy may give acceptable results, we will argue that this is not actually the case here. For this we will briefly present and evaluate one of the most popular algorithms for Gabor filter selection – the Feature Finding Neural Net (FNNN) [66] – along with another, general-purpose feature selection method called Sequential Forward Floating Selection (SFFS) [210]. Then we will introduce a filter set selected manually, where the selection was guided by some simple heuristic. We will show that this manually selected filter set almost always outperforms those found with automatic methods, not only by comparing its performance to the filter sets we created, but also to filter sets found in the literature. And to gain further insight into the performance of feature selection methods, the peformance of the resulting filter sets will also be compared to the performance of randomly selected filter sets as well as the performance of the MFCC features.

Before discussing the details of selecting the proper filters for secondary feature extraction, the primary feature extraction step should also be briefly discussed. Here, as outlined in Section 2.2.2 the log mel-spectrum was used as the primary feature set. And it was computed using 400 samples (25 ms) per frame at 160 sample (10 ms) hops, and based on the result the mel-filterbank was applied with 26 channels. In the case of third party filter sets, where the original study recommended a different number of mel-filters, both settings were used in the first experiments; and based on their results in the remaining experiments we used the better performing configuration.

## 2.3.1 Creating new filter sets

In equations (2.7) and (2.8) we presented the most important parameters regarding our filter sets, some of which we sought to optimise using feature selection methods. There are, however, other parameters that also strongly affect the results. Some of these are parameters that are not present in the formulas determining the filters, such as the number of filters applied on individual patches, and the overlap of the patches used for filtering. We selected the value of these parameters following the experiments with 2D DCT features: on each patch 9 filters were applied, and the overlap between consecutive patches along the frequency axis had to be between 50 and 60%.

There were other parameters, present in equations (2.6) and (2.7), that we fixed based on earlier experiments, and findings in the literature. Such parameters are the width and position of the Gaussian, and the size of the filters. First, concerning the parameters of the Gaussian, we followed Ezzat et al. [49], limiting the bandwidth of the Gaussian to one third of the patch size, and fixing its peak to the patch centre. Also, for computational reasons, only the real part of Gabor filters were retained. Secondly, the width of the patches was set to 9 columns found from earlier experiments on 2D DCT features, as well as studies of human speech understanding [220]. The width combined with the aim of creating square filters determined the height of the patches as well. Furthermore, to incorporate temporal information from a wider context (than the 120 ms that is covered by one patch), we also applied the standard $\Delta$ and $\Delta\Delta$ coefficients using HTK.

For the purposes of automatic feature selection, after setting the number of filters, filter size and filter overlap, the original formulation of the Gabor filters was modified to make the definition of the search space simpler. Motivated by the observation that although the sinusoid (2.7) of a Gabor filter is defined by four parameters $(\Omega, \omega, M, N)$ its ouput does not depend on the individual values of these, but on their ratios ($\frac{\Omega}{M}$, and $\frac{\omega}{N}$). We reformulated the original equation using two parameters

$$S_{\Omega,\omega}(f,t) = e^{i2\pi(\omega_f \cdot f + \omega_t \cdot t)}, \tag{2.9}$$

where $\omega_f = \frac{\Omega}{M}$, and $\omega_t = \frac{\omega}{N}$. The next step required by the feature selection algorithms was to define the interval and resolution of these new parameters. And in order to keep the periodicity of the sinusoids within a resonable interval, the minimum and maximum parameter values were set to -0.14 and 0.14 respectively. To exclude filters that are very similar, the resolution of the parameters was set to 0.004 (for symmetry reasons, $\omega_f$ was run from 0). But even with the above-mentioned restrictions, $\omega_f$ and $\omega_t$ still define a pool of 2556 filters.

### Filter sets created using Filter Finding Neural Net

The Feature Finding Neural Net [66] algorithm was popularised for the purpose of Gabor filter selection by Kleinschmidt and Gelbart [118]. During its operation this algorithm maintains a set of candidate filters, consisting of D+1 filters. In our case $D = 9$, the number of filters we would like to have in the final set, as in earlier experiments 9 filters proved to be sufficient. To obtain the candidate filter set, first we have to initialise the algorithm by creating an initial set. We can do this simply by randomly selecting the necessary number of filters. After the initilisation step, the algorithm repeats three further steps. Namely,

1. Evaluation step: Evaluate each D-element subset of the current candidate set. We perform this step by training a two-layered neural net on 90% of the training set of the TIMIT database, while using the remaining 10% as a stopping criterion, along with a validation set for evaluating the performance of the individual subsets.
2. Elimination step: Take the subset that yielded the best performance in the evaluation step (i.e. the subset that performed best on the validation set), and eliminate the filter from the candidate set that was not part of the best performing subset.
3. Replacement step: Choose a filter randomly, as a replacement for the eliminated filter, to keep the number of items in the candidate set constant. (To avoid selecting a filter that had already been evaluated alongside the current best-performing D-element filter set – which would lead to unnecessary computations – it may be useful to maintain a list of the eliminated filters. This could also be helpful in terminating the algorithm.)

By repeating these steps, we expect to gradually improve our filter set. The algorithm terminates when it attains a local minimum and cannot improve the feature set any further.

As the resulting set depends on the initial set, and the algorithm only guarantees finding a local optimum, we repeated the algorithm fifty times using fifty different initial random sets. We trained thirty three-layered neural nets on each resulting set, and then chose the best performing set by comparing the average accuracy values on the validation set. The set of filters got by using this approach will be referred to as the *FFNN set.*

## Filter sets created using the Sequential Forward Floating Selection

To find a good set of Gabor filters, we also experimented with another, general-purpose feature selection method called Sequential Forward Floating Selection (SFFS), introduced by Pudil et al. [184]. This algorithm is a member of the family of floating search methods, all of which have the added flexibility that previously selected features can later be discarded (unlike in forward selection), and already discarded features can later be reconsidered (unlike in backward selection) [217]. SFFS is a state-of-the-art method [170] that stands out even when compared with other floating search methods [242]. As a detailed analysis of this algorithm by Somol et al. is available in multiple sources [209, 210], and automatic feature selection algorithms are not the main topic of this thesis, here we shall confine ourselves to a short description. Essentially, the SFFS method consists of the following steps:

1. Initialisation: Start with an empty set of features
2. Inclusion step: Expand the current feature set by finding and adding the feature that improves the classification accuracy the most. Here this is done by adding one feature at a time, and using each new feature set to retrain the classifier using a randomly selected 90% of the training set of TIMIT. The feature set is expanded with the feature whose addition provided the best result on the remaining 10%.
3. (Conditional) Exclusion step: Find and discard the feature whose absence has the least detrimental effect on the classification performance (unless this feature was the last one added to the set. If this was the case, proceed immediately with the Inclusion step). If the resulting new set is better than the previous best (achieved using the same number of features in the feature set - noting that this requires that with a current set size of k, the best result should be stored for all $j < k$ feature set sizes), then finalise the exclusion of the feature, and repeat this step with the smaller feature set. If not, reinstate the excluded feature and go to the Inclusion step. This step was performed by removing one feature at a time, and retraining the classifier with the reduced set on the randomly selected 90% of the TIMIT train set. An evaluation of the models created using each set was again performed on the remaining 10% of the training set.

We repeated the feature selection experiment using this algorithm, and the resulting set of 9 Gabor filters will be referred to as the *SFFS set.*

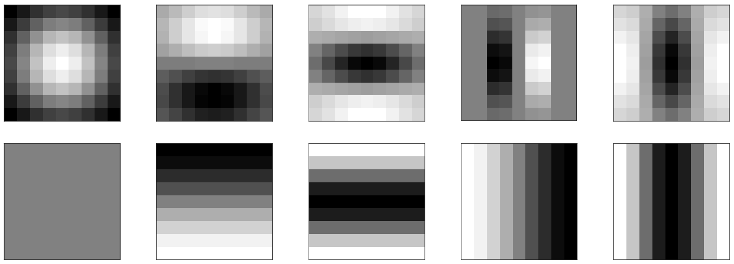

Figure 2.5: Part of the Manual set of Gabor filters (first row), and the filter set corresponding to the 2D DCT coefficients used (second row), with the corresponding filters vertically aligned to emphasise the similarities. The patch size is $9 \times 9$.

### Manually selected filter set

Apart from the automatic feature selection algorithms presented above, we also created a filter set by hand. For this manual selection process we used two sorts of a priori information. First, as in earlier studies it was found that processing the spectro-temporal patches via 2D DCT yields good results (see Section 2.2.), we conjectured that the filters defined by the 2D DCT coefficients would serve as a good starting point for our search. Second, a visual inspection of the shape of the most frequent transition types in spectrograms also served as a heuristic for which filters ought to be included in the final set. In the following, we will elaborate on how the proposed set was developed.

In the earlier experiments (see Section 2.2), the local patches were processed by applying the 2D DCT on them. Retaining nine coefficients proved to be an efficient representation during these experiments, which led us to exploit the similarity between 2D DCT and the sinusoid form in the Gabor filter, as outlined in Section 2.3. As the new filter set relies heavily on an earlier 2D DCT set, the cardinality of it was also chosen to be nine.

The selection of the first four filters was primarily motivated by the similarity between the formulas defining Gabor filters and 2D DCT coefficients. The first row of Figure 2.5 shows how the selected Gabor filters approximate the corresponding 2D DCT filters. As can be seen, a rotationally symmetric filter was included to detect the energy of the processed patch, while with the use of the four other filters we sought to capture the change in frequency and time, respectively. However, motivated by the similarity between the fourth filter in the second row of Figure 2.5 and the $\Delta$ vector, we modified the corresponding Gabor filter in such a way that it approximates the computation of the $\Delta$ vector.

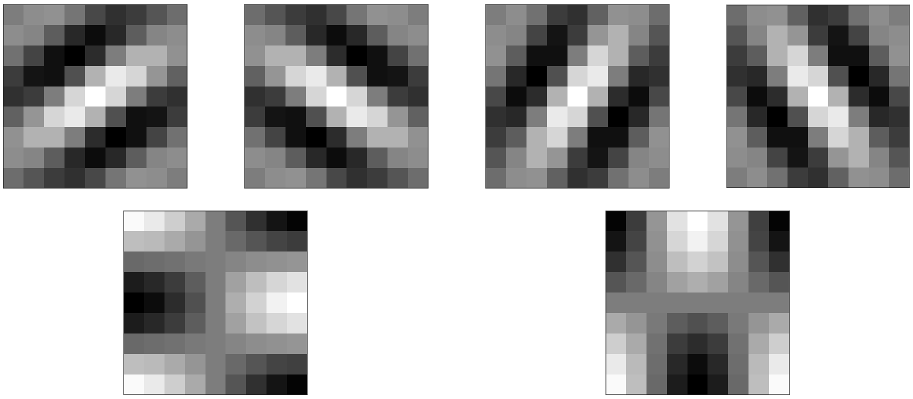

Figure 2.6: Part of the Manual set of Gabor filters (first row), and the filter set corresponding to the 2D DCT coefficients used (second row), with the corresponding filters vertically aligned to emphasise the similarities. The patch size is $9 \times 9$.

Conventional MFCC features extended with the $\Delta$ and $\Delta\Delta$ coefficients capture transitions in time and frequency independently. In contrast, with 2D DCT and Gabor filters, we can extract slanted energy transitions in the time-frequency domain in an explicit way. Presumably this property gives spectro-temporal features an edge over conventional features because it encodes spectro-temporal representations directly. This motivated the selection of the second group of filters in the filter set. The four slanted Gabor filters (shown in the first row of Figure 2.6) were selected by distributing the “slanting” of the filters uniformly. With these filters (which are also like the filters corresponding to the 2D DCT coefficients), we intend to detect spectro-temporal phenomena like formant transitions. Below, we will refer to the resulting set of 9 Gabor filters shown in the first rows of figures 2.5 and 2.6 as the *Manual set*.

## Randomly created filter set

Besides the automatic and manual filter selection methods, we also created ten additional filter sets by simply choosing filters randomly. These filter sets will serve as a baseline during the evaluation of the various filter selection algorithms. We will refer to the average performance of the ten random filter sets as *Random avg.* We will also list the performance score of the best performing random filter set that was chosen based on the results got on the validation set of TIMIT, which will be referred to in the tables as *Random best.*

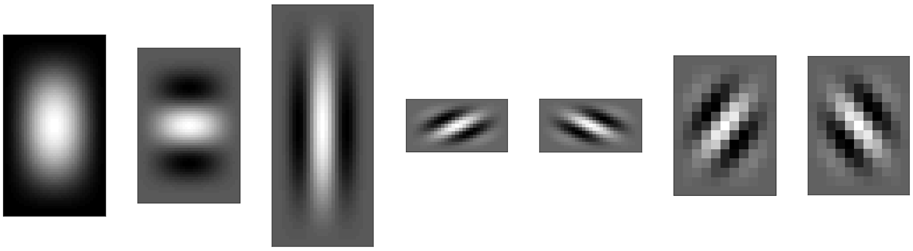

Figure 2.7: Examples from the SMK Gabor filter set (real part).

## 2.3.2 Filter sets in the literature

Kleinschmidt et al. not only described the FFNN algorithm in their paper [118], but also the three filter sets derived from their experiments were made publicly available. These filter sets are referred to as *G1, G2, G3*, and are available for download at the Berkeley website [60]. Each set consists of 60 filters, among them were real, imaginary and magnitude responses of Gabor filters, as well as purely spectral, purely temporal and spectro-temporal filters. The parameters of *G1* and *G2* were trained on TIMIT, while the parameters of *G3* were optimised on the zifkom corpus of German digits.

Schädler, Meyer and Kollmeier in their study [201] applied a different approach for selecting their Gabor filter set, and their method has some similarities with the manual selection method presented earlier (some filters are also similar in the two sets, as shown in Figure 2.7). They chose the parameters of spectral modulation frequencies based on MFCC and temporal modulation frequencies based on RASTA processing. Then they carried out a further optimisation of the parameters by performing some ASR experiments on the AURORA 2 task, creating a filter bank of 41 filters. Next, by subsampling the filter output for each filter, they selected the representative filter channels, which was again carried out on the AURORA 2 database. The result of this process was a filter set containing 311 elements in the case of the 23 channel log mel-scale spectrogram, and a filter set containing 356 elements in the case of the 26 channel log mel-scale spectrogram. The resulting filter sets and the code for feature extraction are available at the website of Oldenburg University [234] These filters will be referred to here as the *SMK set*, after the initials of the authors of the original paper.

### 2.3.3 Experimental settings

The frame-level feature vectors were created from the local patches as follows. First, each filter in the set was evaluated on each patch of the spectrogram. For the filter sets we created using automatic methods or manually, the patches had a length of 9 frames and a height of 9 channels, with a step size of 4 mels (4 channels) in frequency. For the pre-calculated filter sets of Kleinschmidt et al. [118], the sets were defined so that the filters covered the whole frequency range, hence no frequency-domain step size was required. Also, the filter set of Schädler et al. [201] did not have a unified step size either, because the filters were selected uniquely by evaluating each filter on each of the frequency band, without taking into account their overlap with the neighbouring filters. Having evaluated the filters, the feature values obtained were associated with the centre position of the patch, giving a set of feature values for each time position. This set of features was used to classify the given frame, as will be described next. As is usual with MFCC, in order to incorporate more temporal information the $\Delta$ and $\Delta\Delta$ ($\Delta$s) features were added to the feature sets, and 9 neighbouring frames of feature vectors were used to train the neural net classifiers. The only exception was the *SMK set.* Based on the results of some pilot experiments and because of the much higher dimensionality of the frame-level feature vectors produced by this set, only 4 neighbouring frames were used during neural net training, and the $\Delta$ and $\Delta\Delta$ features were left out.

In each experiment, the classifier applied was a multilayer perceptron neural network (MLP) with a hidden layer of 1000 neurons during filter set construction and 4000 neurons during the classification experiments. (In the FFNN algorithm only two layers were used as the number of nets to be trained would have produced a huge computational cost.) While the hidden neurons worked with the sigmoid activation function, in the output layer we applied the softmax nonlinearity. The number of neurons in the output layer was set according to the number of classes in the given task. Naturally, the number of inputs also depended on the number of features extracted by the currently applied filter set. This difference among filter sets should be compensated for by the relatively large size of the hidden layer. The neural net was trained with random initial weights using standard backpropagation on 90% of the training data in semi-batch mode, and the remaining randomly selected 10% of the data set was used as the validation data set. For each feature set ten independent neural nets were trained, and the average of the results got from applying these neural nets on the test set was reported here.

Table 2.9: Phone error rates (PER) got on the clean core test set of TIMIT (the average of 10 independently trained neural nets). The best score, and the scores not significantly different from it are shown in bold.

| Feature set | No. of mel channels | No. of features | PER |
|---|---|---|---|
| MFCC + Δs | 26 | 39 | 27.05% |
| SFFS set + Δs | 26 | 162 | **26.82%** |
| FFNN set + Δs | 26 | 162 | **26.85%** |
| G1 + Δs | 26 | 180 | 27.43% |
| G2 + Δs | 26 | 180 | 27.80% |
| G3 + Δs | 23 | 180 | 35.27% |
| SMK set | 23 | 311 | 28.49% |
| Manual set + Δs | 26 | 162 | **26.78%** |
| Random avg + Δs | 26 | 162 | 27.66% |
| Random best + Δs | 26 | 162 | **26.86%** |

### 2.3.4 Experiments and discussion

#### Experiments on clean speech using the TIMIT database

Table 2.9. lists the phone recognition error rates got from applying the different neural nets trained on the different feature sets on the original (clean) version of the TIMIT core test set. As a baseline, the recognition results got with the standard MFCC features are shown in the first line of the table. The following two rows show the results obtained using the filter sets we created by means of automatic feature selection methods. As can be seen, both of these filter sets yielded a slightly better performance than the MFCC, giving us the first indication that the automatic selection methods work as they were intended to.

The table also contains the results got with the preexisting *G1*, *G2*, and *G3* sets. These sets all yielded significantly worse results than either MFCC or the filter sets we created by using automatic feature selection methods. In particular, the performance of set *G3* is strikingly low, compared to the two other sets. This result seems particularly odd, as in the original paper [118] *G3* was supposed to have been the best performing set. A possible explanation for this phenomenon might be that while the parameters of *G1* and *G2* were optimised on TIMIT, *G3* was optimised on the 'zifkom' corpus of German digits. This suggests that filter sets were overtrained on the particular database, either overfitting on language specific information (German vs. English) or overfitting on the acoustic properties of the database.

Lastly, Table 2.9. shows the results obtained with the manually and randomly selected feature sets. As can be seen, the *Manual set* slightly outperforms every other feature set, and the difference is significant in the case of all the predefined Gabor filter sets as well as in the case of the baseline MFCCs. This strongly suggests that the automatic filter selection methods fail to find the optimal features, as the optimal set should be at least as good or better than a filter set created using very simple heuristics.

We observe even more surprising results when examining the performance of the randomly selected feature sets. At first it might seem counterintuitive, how close the average result obtained with random sets is to the performance of feature sets resulting from carefully designed selection methods. In fact, the average performance of the randomly selected sets is significantly better than that of two Gabor filter sets found in the literature (*G3* and *SMK*), while only one such set (*G1*) produces a significantly better result. Furthermore, the performance of the best random set is not significantly different from that of the best optimised feature set. We think this behaviour is rather unexpected and it indicates that the simple greedy feature selection strategy fails. These issues will later be discussed in more detail in Section 2.3.5.

### Experiments on noise contaminated speech using the TIMIT database

The models trained on clean speech were also evaluated on the noise-contaminated core test set of TIMIT. It should be added here that the filter selection and classifier training steps were NOT repeated. That is, the features and neural network parameters that were found to be optimal in the set of experiments *on clean data were also used in the experiments on noisy data.*

The results got with various types of noise added can be seen in Table 2.10. We see that with either babble or pink noise, the spectro-temporal filters proved much better than MFCCs in almost every case, regardless of the feature selection method applied. However under band-limited noise, only the best performing spectro-temporal feature set (*Manual*) gave better scores than the MFCCs. A thorough analysis revealed that this relative failure of Gabor filters for band-limited noise can be attributed to the simple normalisation technique used: due to the presence of a narrow but very loud noisy spectral channel, the remaining clean channels are also suppressed during normalisation. In the future, a better normalisation technique should be applied to alleviate this effect.

Comparing the two filter sets we created by automatic feature selection methods (*SFFS* and *FFNN* set), we see that while under clean conditions there was no significant difference between them regarding their performance, this is not so in the case of noise. The *SFFS set* significantly outperforms the *FFNN set* in all

Table 2.10: Phone error rates (PER) got on the core test set of TIMIT, contaminated with different types of noise. The best scores in each column (and those scores not significantly different from it) are shown in bold.

| Feature set | Babble | | Pink | | Band-limited | |
|---|---|---|---|---|---|---|
| | 20db | 10db | 20db | 10db | 20db | 10db |
| MFCC + Δs | 42.57% | 64.68% | 50.07% | 69.95% | 38.22% | **49.52%** |
| SFFS set + Δs | **36.90%** | **52.92%** | 44.23% | 65.82% | 39.41% | 52.60% |
| FFNN set + Δs | 37.25% | 53.98% | 44.76% | 66.82% | 41.50% | 53.03% |
| G1 + Δs | 40.23% | 60.41% | **43.80%** | **64.01%** | 45.68% | 55.68% |
| G2 + Δs | 39.25% | 59.48% | 44.14% | **64.08%** | 48.85% | 58.09% |
| G3 + Δs | 48.03% | 62.51% | 48.95% | **64.12%** | 60.03% | 69.15% |
| SMK set | 45.72% | 70.03% | 44.86% | 66.23% | 45.96% | 55.27% |
| Manual set + Δs | **36.79%** | **52.97%** | 44.26% | 65.76% | **36.41%** | **49.13%** |
| Random avg + Δs | 37.49% | 54.01% | 45.74% | 66.58% | 39.75% | 52.36% |
| Random best + Δs | 37.13% | 53.13% | 44.95% | 66.23% | 39.94% | 53.07% |

but one case. As regards the manually selected filter set, we observe that not only does it outperform the baseline MFCC in every case, it also significantly outperforms the *FFNN* set in all cases here, and it is never significantly worse than the *SFFS* set. Extending the comparison to the preexisting filter sets, we see that in the case of pink noise the *G1* and *G2* sets perform slightly better than the *Manual set*, but in every other case the performance score of the *Manual set* is significantly better than the performance score of any preexisting Gabor filter set. Overall, we can say that the *Manual set* performed better in noisy environments than any other feature set examined. This suggests that the simple heuristic rules applied in its construction work better in noisy conditions than the heuristics applied by the SFFS and FFNN feature selection methods. The fact that the performance of *G3* is again noticably worse than the performance of *G1* and *G2* reinforces our concerns about the cross-database performance problems of the automatic selection methods.

Now, when we turn our attention to the *random* sets, we notice that the one selected as "best" based on its performance on the clean validation set performs better than the average in most cases (with the exception of band-limited noise conditions). Compared to the predefined sets, the random filter sets outperform the former sets under babble and band-limited noise conditions, while the predefined sets work much better in the case of pink noise. What is really strange is that the performance gap between the random sets and the best set is always very small (just 1-2%). One would expect a strong feature selection method to yield in a feature set that performs consistently and convincingly better than a randomly selected one, but this was not apparent here.

Table 2.11: Phone error rates (PER) got on the clean test set of the "Szeged" Hungarian broadcast news corpus.

| Feature set | PER |
|---|---|
| MFCC + $\Delta$s | **25.03%** |
| SFFS set + $\Delta$s | 26.06% |
| FFNN set + $\Delta$s | 26.49% |
| G1 + $\Delta$s | 25.91% |
| G2 + $\Delta$s | 27.16% |
| G3 + $\Delta$s | 36.32% |
| SMK set | 26.95% |
| Manual set + $\Delta$s | 25.33% |
| Random avg + $\Delta$s | 27.21% |
| Random best + $\Delta$s | 26.37% |

### Experiments on cross-database performance using the "Szeged" corpus

Because we were concerned about the cross-database performance of feature selection methods, we decided to evaluate the filter sets on the Hungarian "Szeged" broadcast news corpus. As was the case with the noisy experiments on TIMIT, we did not repeat the selection of the filters, but used the filter sets optimised on the clean training data of TIMIT. These cross-language and cross-database tests were motivated by theoretical and practical aspects. Theoretically, one would expect the cortical receptive fields – and thus the Gabor filters that model them – to extract a set of invariant features that does not depend on a training database[1]. And a practical reason is that the automatic feature selection methods like *FFNN* and *SFFS* are extremely slow, so it would be advantageous if the feature selection process did not have to be repeated for each training corpus.

The phone recognition results got on the "Szeged" corpus are shown in Table 2.11. Evidently, the MFCC features yielded the best results, and from among the various Gabor filter sets only the *Manual set* managed to come close. All the feature selection algorithms and the predefined filters sets produced much higher error rates. This point clearly shows that the heuristics applied in the manual selection process are more general than the heuristics of the database-driven feature selection methods. By comparing the results got using various filter selection algorithms with each other, we see that the feature set created by SFFS significantly outperformed its FFNN created counterpart. As we observed a similar tendency with TIMIT, SFFS here seems to be a better

[1]One might argue that these features may be different for different languages, however. This issue should be examined to see if it is really the case.

selection algorithm than FFNN. However, the resulting filter set still performs significantly worse than MFCCs (and the *Manual set*). The same could be said about the *G1-3* filter sets created by Kleinschmidt and Gelbart, and the SMK filter set created by Schädler et al. So unfortunately the hope that the filter optimisation would not have to be repeated for each training database did not materialise. Also, we can see that the best randomly selected filter set gave scores that are almost as good as those of the *SFFS set*, and are better than the scores obtained with the *G2, G3* or the *SMK set*. This again suggests that the filter selection algorithms actually fail to achieve their goal.

Examining the available Gabor filter sets, we notice similar patterns as in the case of TIMIT: while the *SMK* set performed in the mid-range, *G3* gave much worse results than those got with the other feature sets.

### 2.3.5 The problem with automatic feature selection

During the experiments we repeatedly found that despite the time invested in the elaborate feature selection algorithms, the filter sets they produce were easily outperformed by a simple manually selected set, and even randomly selected sets could yield a very similar performance. This is surely not what one would expect. Hence below we try to provide an insight into the possible reasons for this.

Feature selection is a very difficult task when the features are strongly correlated, which is clearly the case with Gabor filters. It is hard to tell how the overlapping information content of the features assist each other, and it is also impossible to tell in advance how the correlations influence the performance of such a complex classifier as a multilayer neural network. Hence, the only way of finding a truly globally optimal feature subset would be to evaluate each possible subset. This is clearly impossible, as the number of subsets grows exponentially with the number of features. The need to reduce computational costs to a managable level encouraged researchers to use greedy search techniques such as the FFNN and the SFFS algorithms. Even when done this way, the given algorithms are very slow (they required several days to finish on TIMIT, which is now regarded as a very small corpus). Both of these algorithms rely on heuristics that assume the evaluation of each subset can be reduced to a search that adds and removes only one filter at a time. Even though the search strategy allows backtracking (which makes it more flexible), convergence would require the replacement of a feature to significantly change the actual performance of the hypothesis set. In this case, however, replacing one filter in the set with another one in most cases has only a negligible effect on the classification score, and this causes the algorithm to oscillate.

Table 2.12: Phone error rates (PER) got on the core test set of the TIMIT database using different versions of the SFFS set, modified in a random manner (the average of 10 independently trained neural nets), and p-values originating from the two-tailed student's t-test [214] with unequal variance, comparing the results of the original SFFS set with its derivatives.

| Feature set | | PER | p-value |
|---|---|---|---|
| SFFS original + $\Delta$s | | 26.82% | |
| SFFS modified | 1 + $\Delta$s | 26.70% | 0.1134 |
| | 2 + $\Delta$s | 26.98% | 0.1246 |
| | 3 + $\Delta$s | 26.87% | 0.7063 |
| | 4 + $\Delta$s | 26.87% | 0.5953 |
| | 5 + $\Delta$s | 26.87% | 0.5439 |
| | 6 + $\Delta$s | 26.97% | 0.1794 |
| | 7 + $\Delta$s | 26.73% | 0.3933 |
| | 8 + $\Delta$s | 26.85% | 0.7120 |
| | 9 + $\Delta$s | 27.03% | 0.0234 |
| | 10 + $\Delta$s | 26.73% | 0.4124 |
| Average of SFFS modified 1-10 | | 26.86% | |

A special experiment was designed so as to highlight the problem inherent in this selection procedure. The best filter set found by the SFFS method was taken, and in it the "first" filter (i.e. the filter that was chosen first by the SFFS method) was replaced by ten arbitrary Gabor filters, resulting in ten additional filter sets. As this filter was chosen first because it was the best "lone" filter for separating the classes, it is reasonable to expect that its replacement should have the biggest impact on the phone recognition scores. To evaluate these ten filter sets, we trained ten independent neural nets on each of them. The results got on the TIMIT core test set are shown in Table 2.12, with the score of the original *SFFS set* being in the *1st* row. We notice that there was no significant difference between the 26.82% phone recognition recognition error rate of the original *SFFS set* and the average phone recognition error rate of its derivatives (i.e the average performance of the sets created from the original *SFFS set* by replacing one filter with an arbitrary new filter), which is 26.86%. It is also true that the average recognition error rates of all the individual SFFS derivatives are very similar to the phone error rates of the original *SFFS set.* In fact, out of the 10 derivative filter sets, only one produced significantly different results (the p-value resulting from the t-test being smaller than 0.05) than the original. Furthermore, the score obtained using the original *SFFS set* is not even the best performing one, being slightly outperformed by 3 derivative sets (one achieving an average of 26.70% and two achieving an average of 26.73% PER).

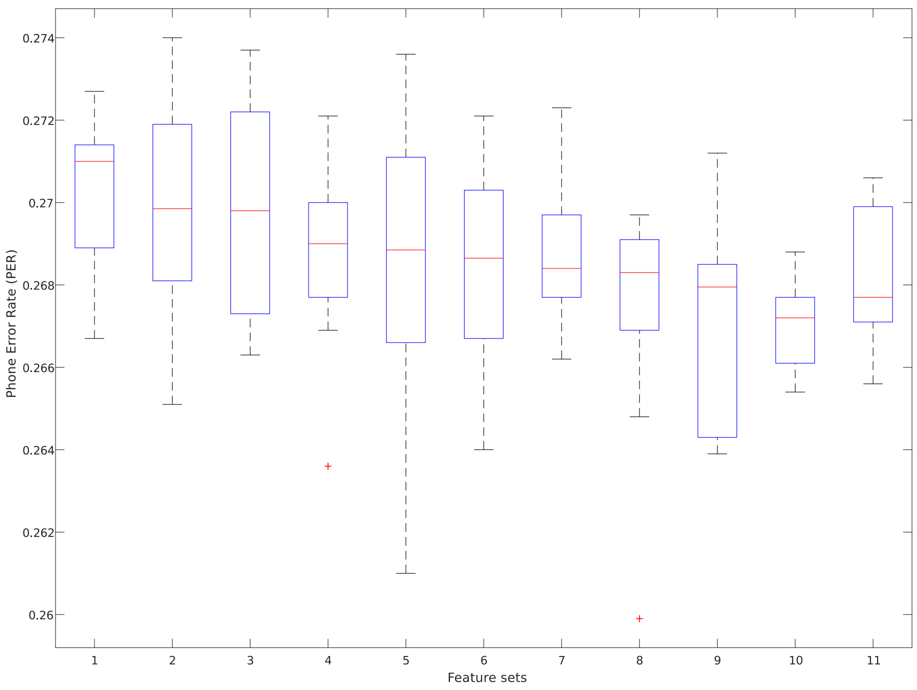

Figure 2.8: The phone recognition accuracy scores got on TIMIT core test set after performing a random filter replacement on the *SFFS set*. The quartiles shown correspond to the scatter of the scores obtained when we repeated the neural net training ten times.

As the neural network training starts from randomly initialised parameters, there is a small scatter in the resulting scores. This scatter can be seen in the box plot in Figure 2.8 represented by quartiles. Here, the results of the original SFFS set are in the last column, while in the previous columns the results of the derivative sets are contained, arranged in descending order of their median value. We not only see that it is not the original set that yields the best median score, but it is also clear that the scatter caused by the random factor of the training process is larger than the scatter caused by the random replacement of the first filter. Hence, the results of this experiment demonstrates that, in the general case, randomly replacing one filter in the set by another one brings about such a small improvement that it can be easily swamped by the "noise" of the ANN used to evaluate the given set. If there are no filters that clearly stand out from the rest, then the simple selection strategy used by our algorithms is doomed to oscillate without convergence.

The results of the last experiment suggest that the feature selection algorithms based on local decisions are not necesarily suitable for the given task. The fact that randomly selected filter sets yield such good results seems to support this conclusion. It suggests that there is a very small performance gap between a randomly selected set and the optimal one, and there are not only no clearly "much better" and "much worse" filters, but it is also difficult to differentiate between a "much better" and "much worse" filter set anyway. This makes the task of an algorithm that decides on each feature locally rather difficult. A possible improvement would be to modify the selection algorithms so that they restrict the addition of a filter to a minimum gain in classification accuracy, or to a certain level of dissimilarity from the previously selected filters.

The reader may find the good performance of randomly selected features rather surprising and counter-intuitive. However, multiple studies on various real-life machine learning tasks have found that creating a representation using a large set of overcomplete random basis functions is just as good as putting a lot of effort into carefully designing a small, orthogonal set of basis functions. The application of sparse, overcomplete bases was first proposed in image recognition, based on observations with vision [101], but quite recently similar studies in speech recognition have also been carried out. For example, Vinyals et al. [232] found that representing the speech signals by means of a randomly selected vector set gave phone recognition results that are just as good as those using an optimised, orthogonal basis vector set.

Further evidence on the applicability of random representations is given by the success of the Extreme Learning Machine (ELM) [89]. The extreme learning machine is virtually a neural network with two hidden layers. During training the lowest layer, as usual, is initialised in a random way, but then *it is not trained at all*, and only the weights of the upper layer are optimised. This strategy at first seems to make no sense, but in fact it can give very good results, and for many real-life tasks it proves no worse than the more tedious training of the full network. The extreme learning machine can be interpreted as if the first layer represented the input by means of a large set of random basis vectors, and then the second layer performed learning over this special representation. The success of this algorithm in practice also reinforces the suspicion that there is only a very small performance gap between a randomly chosen feature set and the optimal one, and hence *this optimisation is a very difficult task that cannot be solved by such naive strategies as the ones used by SFFS and FFNN.*

## 2.4 Conclusions

*In this chapter we examined two spectro-temporal feature extraction methods, namely the 2D DCT, and processing using the family of Gabor filters. We showed that a 2D DCT extraction like that described in the papers of Bouvrie et al. can be performed on the conventional critical-band log-energy representation as well, yielding similar recognition accuracies while requiring less computational effort. In accordance with the finding of the said authors, we found that these localised spectro-temporal features extracted by applying the 2D DCT can result similar, or better phone classification and phone recognition accuracies than the conventional MFCC coefficents. We also found that the advantage of the former is even more pronounced in noise contaminated speech. We also presented a simple yet effective strategy for combining the conventional and the new feature sets, and demonstrated that it can produce a better performance compared to that using just the standard features*

*In the second part of this chapter we showed that with a properly selected Gabor filter set, we can also achieve similar or better phone recognition error rates than that with the standard MFCC features. We also showed that in case of noise contaminated speech, we can get markedly better results with a properly designed Gabor filter set than those using MFCC features, especially in the case of the real-life babble noise. It is however not evident what constitutes a properly assembled filter set, and how they should be constructed. Most authors to date propose the use of automatic feature selection methods such as the FFNN method described earlier. The filter sets found by these algorithms give a reasonably good performance, comparable to that of standard MFCCs. These good results may have given the false impression that the feature selection methods are able to find optimal, or at least near-optimal parameter values. However, here we pointed out that the good performance may originate from the surprising fact that even a randomly selected filter set can yield nice results. This phenomenon is a relatively new observation in image processing, which is currently shifting its research efforts from finding a small but intensively optimised set of basis functions to using larger, overcomplete bases that are chosen almost randomly. Using the results of several experiments we also highlighted the problems plaguing the feature selection strategies used by the FFNN and SFFS algorithms during the selection of Gabor filters, and introduced a filter set created based on a simple heuristic that had better phone recognition performance and, more importantly, better cross-database and cross-language performance than the filter sets created by automatic means.*

# Chapter 3

# The Joint Training of Spectro-Temporal Features and Neural Nets

*In the traditional approach of recognition systems, the extraction of a fixed set of features, and the training of the adaptable classifier, are quite separate. In spectro-temporal speech processing this means that the extraction of features and the training of the acoustical model are traditionally done in two separate steps. While this separation is technically convenient, it might result in suboptimal features for the machine learning method applied.*

*In this chapter we introduce a neural network-based framework that combines the step of secondary feature extraction with the training of the acoustical model. This method enables the further training of initial features based on the speech data, meaning that we do not have to rely on a human expert to provide an optimal feature set for the given task. We demonstrate on various speech recognition tasks that in this framework the potential for adjusting the features leads to a better performance. We also show that this added capability of the neural net does not entirely eliminate the need for well-constructed feature sets, as providing the framework with a proper initial feature set yields further improved results in many cases. In the second part of this chapter, we demonstrate the compatibility of this technique with recent advances in neural nets. We do so by first replacing the hidden layer with multiple layers using the rectifier activation function, then by introducing convolution into the framework. We will see that both changes significantly reduce our error rates.*

## 3.1 Introduction

Traditionally, in pattern recognition systems, the task of feature extraction and model learning are delegated to separate modules. The first, fixed module is usually built on prior knowledge, while the second, trainable module is optimised by machine learning [130]. In this scenario the success of pattern recognition relies heavily on the suitability of the predefined feature set, be it manually or automatically constructed. Because of this, a major part of the research in pattern recognition has been devoted to the task of selecting a suitable set of features, and assessing the applicability of the resulting feature sets in the subsequent machine learning step.

It is easy to see the flaw in this approach when the feature set is entirely hand-crafted: the accuracy of recognition largely depends on the ability of our expert to design a useful feature set. In the previous chapter we also saw the complications associated with automatically assembled feature sets: the feature selection algorithms examined (SFFS and FFNN) not only proved to be slow, but they also failed to produce feature sets that would be suitable for other data sets as well. In fact, the manually selected filter set not only gave better results in the cross-database tests, but in most cases it surpassed the feature sets created by automatic means on our original task as well. This seems to suggest that the manually selected filter set is the one that should be used in future experiments. Unfortunately, the selection heuristics applied in the filter set design do not provide a guarantee either that the resulting set is optimal (in fact in the cross-database experiment of Section 2.3.4 we found that the standard MFCC features outperformed the suggested filter set by a small margin). Furthermore, the manual selection process does not offer the means to improve the filter set when more training data becomes available.

For the above reasons, our suggestion is to combine the feature selection step and the statistical modelling step into one. This has been suggested in the domain of image processing [130], and there are also studies that try to incorporate the feature extraction parameters into the optimisation of the acoustic models [16, 131, 134]. More recently, with the introduction of deep neural nets, several authors have suggested that these new types of networks are powerful enough to accept a less well prepared input than is usual with standard HMMs. For example, it now seems widely accepted that deep neural networks work more efficiently on a simple mel-spectral representation than with MFCCs [162]. Some authors even tried to train deep neural networks on raw acoustic data [95, 174]. Another appealing quality of the neural nets for the task is that their “architecture allows the incorporation and exploitation of existing knowledge about speech” [208].

In a recent study, the training of a convolutional neural net was extended to the optimisation of the feature extraction filter bank [197]. The solution proposed here is based on a similar concept, but it uses spectro-temporal filters instead of a conventional spectral filter bank. This method treats the feature extraction filters as the lowest layer of a neural net, subsequently enabling the training algorithm to fine-tune the filter coefficients as well.

To understand how it is possible to carry this out, let us recall from Section 2.1 the formula characterising the output of applying spectro-temporal filter $F$ on patch $P$:

$$o = \sum_{f=0}^{M-1} \sum_{t=0}^{N-1} P(f,t)F(f,t), \tag{2.1 revisited}$$

where $M$ and $N$ are the respective height and width of patch $P$ and filter $F$. If we represent $P$ and $F$ in vector form (as $\overline{P}$ and $\overline{F}$ which is just a notational change), we get the following formula:

$$o = \sum_{i=1}^{M \cdot N} \overline{F}_i \cdot \overline{P}_i. \tag{3.1}$$

Let us also recall from Section 1.4.2 the formula specifying the output $o$ of a perceptron:

$$o = a\left(\sum_{i=1}^{L} x_i \cdot w_i + b\right), \tag{1.13 revisited}$$

where $\mathbf{x}$ is the input of the neuron, $L$ is the length of the input, $\mathbf{w}$ is the weight vector, and $b$ is a bias corresponding to that neuron. For the activation function $a$ we usually apply the sigmoid function; but it is also possible to create a linear neuron by setting $a$ to the identity function. If we select the identity function as the activation function of our perceptron, and set the bias $b$ to be zero, we can write (1.13) in the following form:

$$o = \sum_{i=1}^{L} x_i \cdot w_i. \tag{3.2}$$

Now it is easy to demonstrate that (1.13) is just a special case of (2.1), by chosing the vectorised version of the patch $P$ as our $x$ input vector, and the vectorised version of $F$ filter as our $w$ weight. This means that the spectro-temporal filters can indeed be integrated into an ANN classifier system as special neurons, with the filter coefficients being the weights of the given neuron.

## 3.2 Experiments using Sigmoid Networks

Although in later stages more advanced neural net techniques will be applied, as a proof of concept, we first integrated this new filter extraction layer into a traditional sigmoid neural net. To test the viability of the proposed method, phone recognition experiments were carried out on the TIMIT speech database (using both the clean version of the core test set and its noise contaminated versions), as well as on the "Szeged" Hungarian broadcast news corpus. Here, based on the results of these experiments, we attempt to find the answer to three important questions regarding the strategy of the joint training of spectro-temporal features and neural nets:

1. Does the fine-tuning of the filter coefficients lead to better recognition results?
2. Does the strategy applied in the initialisation of the filter coefficients affect the recognition results, and if so, in what way?
3. How does the cross-language performance of the resulting feature set relate to that of the filter sets introduced in Section 2.3?

In these experiments the structure of the feature extraction layer was specified based on the parameters used in the experiments described in Section 2.3, as the parameters given there not only are results of extensive optimisation, but by using the same parameters a comparison of the respective results on the Hungarian database becomes more informative. In broad terms this means that the patch size was $9 \times 9$ (or in different terms, each layer had 81 inputs), the step size of patches in the frequency domain was fixed at 4 (leading to the use of six patches to cover the full frequency domain[1]), and on each patch 9 filters were applied (or in different terms, each of the 6 sub-layers had nine neurons). A detailed structure of the neural network, and the feature extraction layer depended on whether or not the neural net was allowed to use a broader context than one patch. The neural net structure was also dependent on the actual task carried out: neural nets used on the TIMIT phone recognition task had an output layer consisting of thirty-nine neurons, while the neural nets used on the "Szeged" Hungarian broadcast news database had an output layer of fifty-two neurons; in accordance with the corresponding number of monophone labels.

[1]Each such patch corresponds to a group of dedicated neurons in the feature extraction layer that is responsible for the processing of the given patch. For convenience, such groups of neurons will be referred to as "sub-layers", or when it does not lead to confusion, simply as layers.

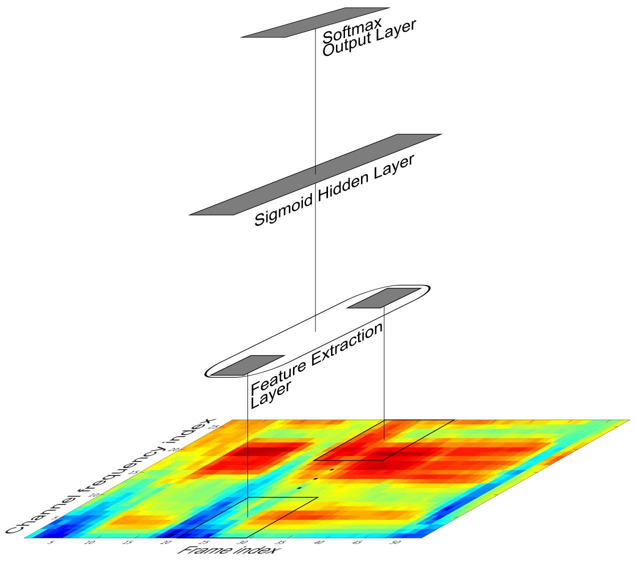

Figure 3.1: Structure of the ANN for joint feature extraction and classification.

### 3.2.1 Limited context

Figure 3.1 shows the structure of the neural net in the case where classification is based on only a context that is the same length as the patches used. When compared with a conventional MLP, the main difference is the introduction of the so-called feature extraction layer (for clarity, only two of the six sub-layers are displayed in the figure). Here, linear neurons perform the filtering step on their input, while their output is channeled into a sigmoid hidden layer. From this point on, the system works just like a conventional MLP. Hence, if the weights of the feature extraction layer were initialised with 2D DCT or Gabor filter coefficients, and were not modified during training, the model would be equivalent to a more traditional system; and incorporating the feature extraction step into the system would be just question of implementation.

The structure in Figure 3.1 allows the algorithm to evaluate the spectro-temporal features and the ANN in one step. However, the main purpose of this structure was to extend the scope of the backpropagation algorithm to the feature extraction layer, making it possible to train the weights associated with the spectro-temporal filters, and hence fine-tune the initial coefficients. Naturally, these coefficients could also be initalised randomly, but as backpropagation only guarantees a locally optimal solution, initialising the model with weights that already provide a good solution may be beneficial.

Table 3.1: Phone error rates (PER) got on the core test set of TIMIT (the average of 20 independently trained neural nets).

| Initial | filter weights | |
|---|---|---|
| filter weights | unaltered | trained |
| Random | 32.96% | 30.27% |
| 2D DCT | 31.19% | 30.21% |
| Gabor | 32.41% | 30.29% |

### Experiments on clean speech using the TIMIT database

The phone recognition results obtained on the TIMIT corpus with neural nets using a limited context, and a hidden layer of a thousand neurons, are listed in Table 3.1. The rows of the table correspond to the various filter initialisation schemes. The first column shows the results obtained when the filter coefficients were not trained, while the second column shows the results got by neural nets that modified the filter coefficients as well during backpropagation. The first thing we notice is that the joint training method always gives better scores than those obtained with fixed filter coefficients (with significance $p < 10^{-11}$). This indicates that the answer to our first question about joint training would be positive.

We can also see that with the filter-coefficients fine-tuned, the initialisation techniques gave practically the same results, so in this case, starting from the 2D DCT or Gabor filters did not help the optimisation process compared to the case with random initialisation. This suggests that the answer to our second question might be negative. However, we also see that in the first column, when there is no fine-tuning of filters involved, the 2D DCT and Gabor filter sets yield significantly lower error rates than the ones obtained using randomly initialised filter coefficients ($p < 10^{-5}$).

This sounds reasonable and, in fact, one might expect much worse results from random filters. Interestingly, there are studies which indicate that in many cases a large set of random base functions can give a representation that is just as good as a carefully selected function set. Recently, a similar study was published for the case of dictionary learning for speech feature extraction [232]. The Extreme Learning Machine of Huang et al. also exploits this suprising fact: their learning model is virtually a two-layer network, where both layers are initialised randomly, and the lowest layer is not trained at all [89].

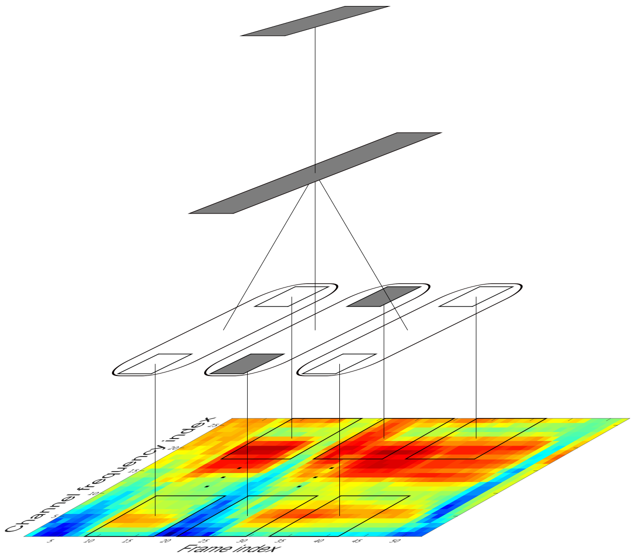

Figure 3.2: Structure of the ANN for joint feature extraction and classification with weight sharing on neighbouring patches in the time domain.

## 3.2.2 Expanded context

Figure 3.2 shows the structure of the ANN in the case where it uses multiple patches as its input in the time domain as well. However, the two domains are handled differently. For one, while the six patches in the frequency domain are positioned using a step size of 4, in the time domain we use 9 patches with a step size of 1 (for clarity, only three are displayed in the figure). Furthermore, while patches extracted from different parts of the frequency domain are channelled into different feature extraction sub-layers applying different weights, this is not the case for patches extracted from different parts of the time domain: these patches are processed in the same sub-layer, using the same weights. From the outside however, they work as we had as many sub-layers, as the number of patches used in the time domain, since the number of inputs required, and the number of outputs provided by them is the same it would be had we used multiple sub-layers. This duality is represented in Figure 3.2 by the addition of two sets of transparent sub-layers. It is interesting to note here that with this modification the resulting framework meets two criteria out of the three required in the standard definition of convolution [231], namely the use of local windows, and weight sharing. And while the third criterion (pooling) was not met in the original study that introduced the framework [124], this approach was also dubbed as convolution.

Table 3.2: Phone error rates (PER) got on the core test set of TIMIT (the average of 20 independently trained neural nets, with a hidden layer of 1000 neurons).

| Initial | filter weights | |
|---|---|---|
| filter weights | unaltered | trained |
| Random | **27.98%** | 27.39% |
| 2D DCT | 28.83% | 27.48% |
| Gabor | 28.26% | **27.17%** |

## Experiments on clean speech using the TIMIT database

Table 3.2 shows the phone recognition scores got on the TIMIT speech corpus using weight sharing on the current patch along with 4 preceeding and 4 following patches in the time domain. We see that the difference between the performance of the fine-tuned and the untrained filter sets is smaller than that for Table 3.1. As regards the initialisation methods of the trained filters, Gabor filters resulted in slightly lower phone recognition error rates in this case ($p < 10^{-2}$). However, this new network structure seems to work just as well with random filters as with 2D DCT coefficients. This is an interesting observation that needs to be examined further. But it is already quite clear that the new structure brings about a clear improvement to the network.

This improvement is also apparent when we examine the results of a similar ANN, with a hidden layer of 4000 neurons. Besides the increase in the number of neurons, here we also tuned the phone insertion penalties on the random 10% of the training database retained for evaluation purposes. As we can see in Table 3.3, while all the error rates were reduced due to the increased number of neurons, it still holds true that training the initial weights results in lower PER rates in both cases. Furthermore, regardless of whether the filter weights were kept the same or altered during the training, here initialising them based on the Gabor filters brings about a significant reduction in the error rates.

Table 3.3: Phone error rates (PER) got on the core test set of TIMIT (the average of 10 independently trained neural nets with a hidden layer of 4000 neurons).

| Initial | filter weights | |
|---|---|---|
| filter weights | unaltered | trained |
| Random | 27.01% | 26.23% |
| Gabor | **26.51%** | **25.69%** |

Table 3.4: Phone error rates (PER) got on the noise contaminated core test set of the TIMIT speech database (the average of 10 independently trained neural nets).

| Noise | Initial Filter weights | 20 dB unaltered | 20 dB trained | 10 dB unaltered | 10 dB trained |
|---|---|---|---|---|---|
| Babble | Random | 38.30% | 37.66% | 56.74% | 56.23% |
| | Gabor | 37.42% | **36.95%** | **54.36%** | **54.13%** |
| Pink | Random | 52.25% | 44.97% | 70.71% | 65.42% |
| | Gabor | 46.60% | **43.13%** | 67.84% | **64.03%** |
| Band limited | Random | 45.02% | **39.93%** | 55.54% | **53.65%** |
| | Gabor | **39.42%** | **39.87%** | 54.24% | **52.53%** |

### Experiments on noise contaminated speech using the TIMIT database

To examine their performance in noisy conditions, the neural nets containing a hidden layer of 4000 neurons that were trained in the previous experiment were also evaluated on the noise-contaminated version of the TIMIT core test dataset (without retraining on the noisy data). In order to make the comparison of phone error rates between pairs of the four versions of neural nets (applying no refinement, applying the refined initialisation technique based on Gabor filters, applying the refined training technique by training the feature coefficients as well, and applying both refinements) easier, in Table 3.4 the error rates are arranged in groups of four. Furthermore, in each group of four, the lowest phone error rates (along with those not significantly different from them) were highlighted in bold.

The error rates here display a pattern similar to that previously observed in Table 3.3. As we can see, in almost every case both the training of filter weights and the initialisation of those weights based on the Gabor filter set reduced the phone error rates in noisy environments as well. However, in most cases the best results were obtained by combining these techniques. In fact a combination of the two methods brought about a significant improvement in performance in the case of pink noise, with both signal-to-noise ratios, and with one signal-to-noise ratio in the case of babble noise. There was also an improvement (albeit not a significant one) in one of the cases of band limited noise, and in the other case of babble noise.

Table 3.5: Phone error rates (PER) got on the "Szeged" Hungarian speech database (the average of 10 independently trained neural nets).

| Initial | Filter weights | |
|---|---|---|
| Filter weights | Unaltered | Trained |
| Random | 26.94% | 25.06% |
| Gabor | 26.36% | 24.75% |
| Gabor (TIMIT) | **25.10%** | **24.64%** |

## Experiments on cross-database performance using the "Szeged" corpus

We also performed phone recognition experiments using the joint-training framework on the "Szeged" Hungarian broadcast news database. First, it allowed us to examine the performance of the framework on a database twice removed from TIMIT (as it is not simply a different database with the same language, but a different database recorded in a different language). It also made it possible to make a comparison between the method of joint-training and the feature selection methods we described in Chapter 2. If we consider the training phase of the neural net as a walk in the feature space, the weights corresponding to the filtering sub-layers can be viewed as the filters this process selects; and conversely, the output of these sub-layers can be viewed as the features selected by the joint-training process. This allows to compare the performance achieved using these features with the performance achieved using features obtained by the feature extraction methods discussed in Chapter 2.

In the initialision of the filter coefficients for the experiments with the "Szeged" corpus, three schemes were used: random initialisaton, initialisation based on the Gabor filter set described in Section 2.3.1, and initialisation based on the same filter set, after it had been trained on the TIMIT database. We can see the results of these experiments in the respective rows of Table 3.5. As is clear here, regardless of whether filter coefficients were further trained on the Hungarian database (column Trained) or not (column Unaltered), the lowest error rates were achieved when using the filter set that had already been enhanced by the joint optimisation method applied on the TIMIT database. This means that unlike in the case of filter selection methods, where the filter sets selected based on TIMIT all have performance degradation when migrated to the Hungarian corpus, the fine-tuning of the filter set on the TIMIT database did not degrade the performance on the Hungarian database, but even increased it. We can also see in this table that (similar to that in most earlier experiments) initialisation based on the original Gabor filters in itself leads to an improved recognition accuracy, and that training the filter coefficients brings about an improvement in each case.

## 3.3 Experiments using Deep Neural Networks

As described in Section 1.4.2, in standard ANN implementations there are ordinarily three layers, namely an input layer, an output layer (applying the softmax nonlinearity), and in between a hidden layer that uses a sigmoid activation function. Recently, it has been shown that a significantly better performance can be achieved by increasing the number of hidden layers [84]. Unfortunately, training these ‘Deep’ Neural Nets (DNNs) with three or more hidden layers has certain difficulties [63]. A solution to these problems was given by Hinton et al., leading to a renaissance of ANN-based technologies in speech processing [84]. An even simpler solution was later given with the introduction of rectifier neural networks [63, 223].

Rectifier neural nets differ from their conventional counterparts only in the activation function used by the neurons in the hidden layer(s). While standard methods use the sigmoid activation function, rectifier nets apply the rectifier function. The difference between these functions can be seen in Figure 3.3. The rectifier function ($rectifier(x) = max(0, x)$) has two interesting properties. The first is its linearity for positive input. Because of this, neurons using this activation function do not saturate as their activity increases. This means that even with multiple hidden layers, the problem of vanishing gradients can be reduced or avoided altogether [223]. Another interesting property is that for all negative values the function returns a constant value of zero. This means that the resulting neural net is usually more sparse, which has computational advantages. Also, sparsity is biologically more plausible [64].

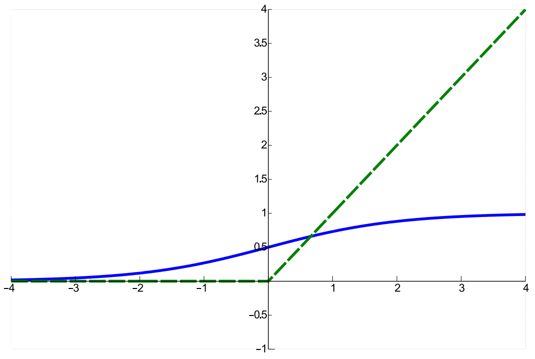

Figure 3.3: Plot of the sigmoid activation function (solid line), and the rectifier activation function (dashed line).

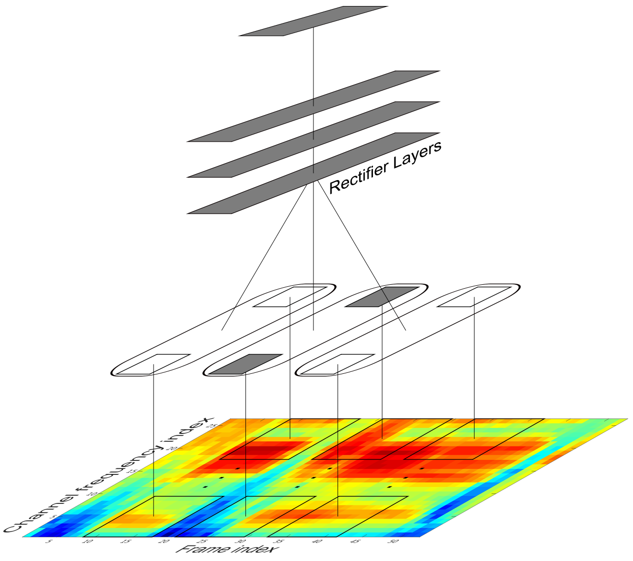

Figure 3.4: Structure of the Deep Rectifier Neural Net (DRN) for joint feature extraction and classification.

Due to the replacement of the sigmoid activation function with the rectifier activation function in the neurons, it was possible for us to use more hidden layers in the joint-traning framework without having to apply a pre-training algorithm [64, 223]. In the experiments presented here, the previously used hidden layer with 4000 sigmoid neurons was replaced with three hidden layers, each one consisting of one thousand rectifying neurons. There were two main reasons for setting the parameters of the new framework to these values. One was that with this choice, it allowed us to keep the number of trainable parameters in the new framework ($\sim$2.5 million) close to the number of trainable parameters in the original one ($\sim$2.1 million). The other reason was that while in our preliminary experiments the progression from two rectifier layers with 1000 neurons to three rectifier layers with the same size provided a significant improvement in the frame-level phone classification error rates on the validation set of TIMIT, the progression from three such layers to four sometimes even harmed the error rates, and when it led to an improvement, the difference was not always significant. The resulting structure is shown in Figure 3.4.

Table 3.6: Phone error rates (PER) got on the core test set of TIMIT (the average of 10 independently trained neural nets).

| Initial | Original framework | | DRN framework | |
|---|---|---|---|---|
| Filter weights | unaltered | trained | unaltered | trained |
| Gabor | 26.51% | **25.69%** | 24.42% | **23.37%** |
| 2D DCT | 27.31% | 26.12% | 25.00% | 24.15% |
| Random | 27.01% | 26.28% | 25.35% | 23.58% |

### 3.3.1 Experiments on clean speech using the TIMIT database

The results of the phone recognition experiments on the core test set of the TIMIT speech corpus are summarised in Table 3.6. For both the original framework (using one sigmoid layer), and the DRN framework, the lowest error rates are highlighted. The first thing to note here is that between these two highlighted results of the two settings, there is a 9% relative error rate reduction for the DRN. Not always to such an extent, but the advantage of deep learning is discernible in each case, with the relative error reduction ranging from 6.5% to 10%. The average was 7.5% for the unaltered settings, and almost 9% for the trained version.

Another observation we can make concerning the case where filter coefficients are modified during neural net training, and where filter coefficients are unaltered is the following: regardless of whether the experiments were carried out with the original framework or with the DRN framework, error rates are lower in each case where the error backpropagation was extended to the filtering layer. This further reinforces the importance of incorporating the feature extraction and training processes into one step. Lastly, we should note that for both the original and the DRN framework, the result highlighted above was attained when the filter coefficients were initialised based on the Gabor filter set. This again shows the value of initialising the filtering layer with the appropriate weights.

### 3.3.2 Experiments on noise contaminated speech using the TIMIT database

The experiments conducted on clean speech were repeated for noise contaminated speech. Here (as usual), the neural nets that had been trained using clean speech were evaluated in noisy conditions. The results in this case will be reported and examined (based on the same questions results with clean speech were examined) separately for artifical and real-life noise types.

Table 3.7: Phone error rates (PER) got on the core test set of TIMIT contaminated with band-limited and pink noise using different SNR values (reported error rates are the average of 10 independently trained neural nets).

| Noise type | SNR | Initial Filter weights | Original framework unaltered | | Original framework trained | DRN framework unaltered | | DRN framework trained |
|---|---|---|---|---|---|---|---|---|
| Band-limited | 10 dB | Gabor | 54.24% | > | 52.53% | 53.64% | > | 52.79% |
| | | 2D DCT | **52.26%** | | 52.43% | **51.48%** | < | 53.48% |
| | | Random | 55.54% | | 53.65% | 54.01% | | 53.86% |
| | 20 dB | Gabor | 39.42%* | | 39.87% | 41.97% | | 41.89% |
| | | 2D DCT | 39.99% | > | **39.29%*** | **41.08%** | < | 42.11% |
| | | Random | 45.11% | > | 40.68% | 41.42% | | 42.20% |
| Pink | 10 dB | Gabor | 67.84% | > | **64.03%** | 62.66% | > | **61.61%*** |
| | | 2D DCT | 67.77% | > | 64.76% | 64.27% | > | 62.81%* |
| | | Random | 70.71% | > | 65.40% | 64.30% | > | 62.96%* |
| | 20 dB | Gabor | 46.60% | > | **43.13%** | 40.68% | > | **39.95%*** |
| | | 2D DCT | 48.16% | > | 44.30% | 42.01% | > | 40.72%* |
| | | Random | 52.25% | > | 45.02% | 43.27% | > | 40.34%* |

### Experiments on Speech Contaminated with Artificial Noise

In the first set of experiments two artificial noise types were used, namely band-limited noise and pink noise. The numerical results of these experiments are listed in Table 3.7. As there is much more data here, a detailed analysis becomes more complex as well. Hence, let us examine the questions raised in the last section separately.

The first question was concerning the way the recognition performance changed when replacing the original framework (using the sigmoid activation function in its hidden layer) with the new DRN framework (using the rectifier activation function). In this case, to make the comparison easier and visually clearer, in each row of Table 3.7 where there is a significant difference between the best result got with the original framework, and the best result got with a DRN framework, the lowest error rate in each row was marked by an asterisk. Here, we can see a marked difference between the two artificial noise types. In the case of pink noise, the recognition scores got by the DRN framework are significantly better than those got by the original framework for each signal-to-noise ratio (SNR) and each weight initialisation scheme; but this is not so in the case of band-limited noise. In the case of speech contaminated with the latter noise type, for the most part there is no significant difference between the error rates produced by the two settings. Furthermore, in both cases where the difference between the pairs of results obtained by the different frameworks is significant, the original framework tends to produce better results.

The second question was about how the recognition performance varies in the case where the filter coefficients are kept constant compared with the case where the filter coefficients were also trained. To make the comparison visually clear, in each unaltered/trained pair of Table 3.7, when there was a significant difference, it was marked by an inequality symbol being placed between the two columns. As can be seen in the table, the two types of artificial noise behave quite differently in this case as well. With pink noise we see the same trend that we saw in the case of clean speech, namely that the trained version invariably yields significantly better recognition rates than the unaltered version. With band-limited noise, however, in most cases there is no significant difference between the trained and unaltered version. And when there is, usually the trained version is better for just the original framework.

Based on the results (see Table 3.7) concerning the first two questions to be examined, we can say that with pink noise the results are in accordance with our preliminary expectations based on the clean speech experiments: the DRN framework not only outperforms the original one, but it also consistently gives significantly lower error rates with trained filter coefficients. This is not the case with band-limited noise, however, where the DNN framework does not provide significantly better results than those got with the original one, and it also tends to produce better scores with fixed filter coefficients than with trained ones.

Lastly, to examine the third question, namely the impact of filter weight initialisation on phone recognition error rates, both in the case of the original and the DRN framework from each group of six error rates (corresponding to the same noise type and SNR), the lowest was highlighted in bold. Here, again there is a difference between the two noise types, but there is also a distinct similarity as well. The best performing initialisation scheme differs between the case of pink, and band-limited noise: for the first artificial noise type, the best phone recognition results were achieved when the filter weights were initialised based on the 2D DCT feature set, while for the second noise type, the lowest error rates were obtained when Gabor filters were applied for the same task. This, however, also means that in both cases the best performance is achieved when the filter coefficients are initialised based on a specific filter set, and not randomly. This indicates that regardless of the artificial noise type used, proper initialisation of the filter coefficients is essential in obtaining low phone recognition error rates.

Table 3.8: Phone error rates (PER) got on the core test set of TIMIT contaminated with Babble, Factory and Volvo noise, with different SNR values (reported error rates are the average of 10 independently trained neural nets).

| Noise type | SNR | Initial Filter weights | Original framework unaltered | | Original framework trained | DRN framework unaltered | | DRN framework trained |
|---|---|---|---|---|---|---|---|---|
| Babble | 10 dB | Gabor | 54.36% | | **54.16%** | 50.61%* | | 50.73% |
| | | 2D DCT | 55.90% | > | 55.04% | 51.33%* | < | 52.75% |
| | | Random | 56.74% | | 56.25% | **50.44%*** | < | 53.38% |
| | 20 dB | Gabor | 37.42% | > | **36.95%** | 33.41% | > | **33.09%*** |
| | | 2D DCT | 38.88% | > | 37.61% | 34.59% | > | 34.17%* |
| | | Random | 38.30% | > | 37.64% | 34.55% | > | 33.42%* |
| Factory | 10 dB | Gabor | 62.37% | > | **58.06%** | 56.07% | > | 54.73%* |
| | | 2D DCT | 61.73% | > | 58.68% | 56.66% | > | 55.58%* |
| | | Random | 58.85% | | 58.95% | 56.30% | > | **54.60%*** |
| | 20 dB | Gabor | 41.89% | > | **38.86%** | 35.93% | > | **35.34%*** |
| | | 2D DCT | 42.16% | > | 39.47% | 37.17% | > | 36.32%* |
| | | Random | 40.11% | | 39.94% | 37.43% | > | 35.52%* |
| Volvo | 10 dB | Gabor | 33.05% | > | 32.61% | 28.89% | > | **28.26%*** |
| | | 2D DCT | 32.97% | | 32.79% | 29.53% | > | 29.19%* |
| | | Random | **32.31%** | < | 33.86% | 29.37% | > | 28.59%* |
| | 20 dB | Gabor | 29.59% | > | **28.48%** | 26.32% | > | **25.53%*** |
| | | 2D DCT | 29.71% | > | 28.70% | 26.67% | > | 26.00%* |
| | | Random | 29.07% | | 29.22% | 26.85% | > | 25.54%* |

#### Experiments on Speech Contaminated with Real-life Noise

The results of experiments with noise types got from real-life environments are summarised in Table 3.8. First, comparing the error rates provided by the original and the DRN framework we see that the latter provided much lower error rates regardless of the noise type. The pattern is also apparent when it comes to the difference of error rates attained using unaltered and trained filter coefficients: for each noise and framework, training the coefficients led to better recognition scores in the majority of the cases. A comparison of initialisation schemes, however, is more complex. For here, in some cases the best result was achieved by randomly initialised weights. But in most cases the initialisation based on Gabor filters still performed better, meaning that, overall, Gabor filters proved to be the most useful for initialisation.

To summarise the findings presented in tables 3.6, 3.7 and 3.8, we can say that with the exception of one noise type, the DRN framework provides better results than the original. Furthermore, in most cases this framework also yields better recognition scores when the filter coefficients are trained. We also see that usually the best performance was achieved when employing Gabor filter initialisation.

## 3.4 Experiments using CNNs

Despite the modifications that we made, two important attributes remained the same. First, in both the original, and the DRN framework, the patches used were direct neighbours. Second, the filtering sub-layers processing the neighbouring patches were combined immediately, in the following layer. This, however, is not necessarily the optimal solution. Hence, below we will describe experiments where instead of simply using neighbouring patches, we sought to apply convolution in the time domain[2].

### 3.4.1 Adjusting the Structure of the Model

So far, the combination of results got from the feature extraction layer was carried out in the first hidden rectifier layer. It may be beneficial though to perform this combination later. This, however, would lead to a massive increase in the parameters of the model, since the number of outputs in the hidden layer would then be the same as the number of neurons in that layer times the number of neighbouring patches used, which in our case is 9000 (1000 times 9). This would increase the number of parameters by $\sim$ 7.5 million. To avoid this, three further modifications are proposed:

- First, instead of having all four neighbours on both sides, only two will be used. This modification will reduce the number of extra parameters to $\sim$ 3.5 million.
- Furthermore, to ensure that the time-span available for the neural network in its input is at least as long as before, the two neighbouring patches both from before and after the current patch are extracted in such a way that every second patch is skipped (or in different terms, we increase the step size/stride from one to two). This modification also means that the framework now also applies pooling on the spectro-temporal features Thus now the framework meets every criteria of convolution [203, 231], and will be referred to as a Deep Convolutional Rectifying Neural Net (DCRN).
- Lastly, the size of the convolutional rectifier layer is reduced from 1000 neurons to 200, meaning that the parameter count of the resulting DCRN framework is quite similar to the parameter count of the original framework ($\sim$ 2.1 million).

[2]One could argue that the structure applied here is a time-delay neural network [2] extended with sub-sampling [229]. Despite this, and the fact that convolution in image processing is conventionally carried out in 2D [203], here, following [3], [132], and [224] we will refer to this structure as a Convolutional Network.

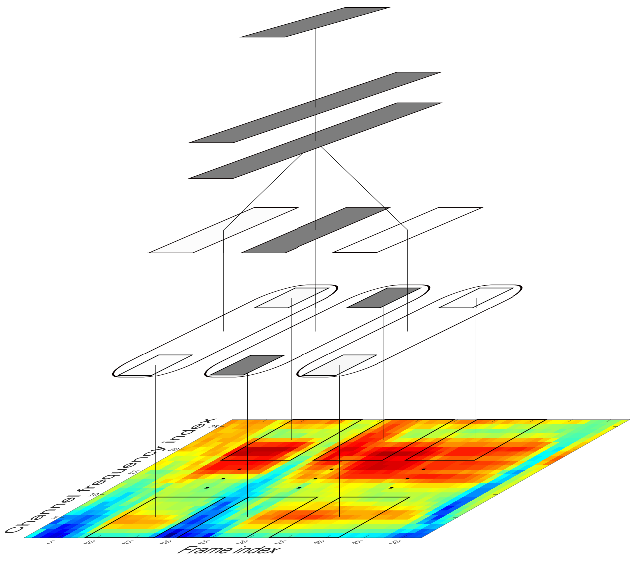

Figure 3.5: Structure of the Deep Convolutional Rectifier Neural Net (DCRN) for joint feature extraction and classification.

The structure of the new neural net framework with the proposed modifications can be seen in Figure 3.5. Again, transparent rectangles signify the virtual layers that occur due to the use of convolution. It should also be noted here that having a step size of two (thus skipping one patch between two patches used), despite performing better, is not necessarily an optimal solution either. Leaving out patches from being processed makes it possible for us to use a wider context in the input of the neural net, without the number of features (and thus the number of parameters to be trained) growing with the growing context. This raises the question of how the step size should be selected for optimal performance. And as our preliminary experiments with the step size being increased to two (skipping one patch between each two patches used) produced quite promising results, we decided to examine this question. We did so by first training 10 independent neural nets for each patch omission number from 1 to 10, initialising filter coefficients randomly, as well as based on Gabor filters and 2D DCT coefficients (meaning that altogether 300 neural nets were trained). The training of these nets was carried out on a randomly selected 90% of the training set of the TIMIT speech corpus. The resulting neural nets were evaluated based on the frame-level phone classification error rates they produced on the remaining 10% of the training set.

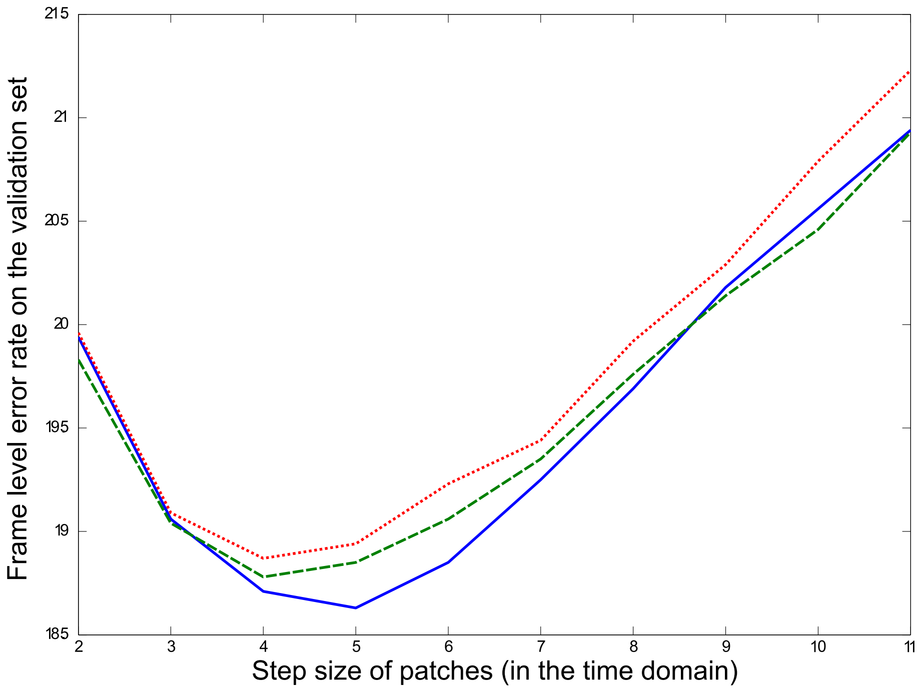

Figure 3.6: The frame-level error rates of the validation set as a function of the step size (in the time domain) in neural networks using different initialisation schemes – Gabor filters (solid line), 2D DCT coefficients (dashed line) and Random initialisation (dotted line).

The average frame-level error rates of these experiments are shown in Figure 3.6. As can be seen, with each initialisation scheme, after an initial decrease, the error rates steadily grow as step size increases. The best results with feature coefficients initialised with 2D DCT and Random filters were achieved (with a 18.78% and a 18.87% frame-level phone classification error rate, respectively) with a step size of four (skipping three patches between two patches used). Thus in later experiments, when working with 2D DCT or randomly initialised filter coefficients, we used a step size of four. With Gabor filters, however, some additional improvement could be achieved by increasing the step size to five (in this case the error rate dropped from 18.71% to 18.63%). As this difference was not significant, and since using a different meta-parameter with Gabor filters would have adversely affected comparability, a step size of four was applied when using Gabor filters as well.

Table 3.9: Phone error rates (PER) got on the clean core test set of TIMIT (reported error rates are the average of 10 independently trained neural nets).

| Initial Filter weights | Original framework | DRN framework | DCRN framework |
|---|---|---|---|
| Gabor | 26.24% | 23.37% | **22.98%** |
| 2D DCT | 26.54% | 24.15% | **23.22%** |
| Random | 26.53% | 23.58% | **23.25%** |

To evaluate the effect of the proposed changes on the phone recognition error rates of the framework, the experiments on the TIMIT speech database were repeated. As in sections 3.2.2 and 3.3, first the phone recognition results got on clean speech will be described, then we will discuss the results got on speech contaminated with different noise types (first, results with artificial noise will be presented, after which we will present the results got on speech signal contaminated with noise types arising in real-life applications). Because in the previous experiments the configuration where the filter coefficients were also trained by the backpropagation algorithm yielded a significantly better performance in most cases, here we only perform experiments using this setting. We will compare the results obtained in the DCRN framework with the best results got with the DRN and the original framework (regardless of whether the result was got using unaltered filter coefficients or trained ones).

### 3.4.2 Experiments on clean speech using the TIMIT database

The results obtained from applying the modified model on the uncontaminated version of the TIMIT core test set are summarised in Table 3.9. As can be seen, regardless of the initialisation scheme applied on the filter coefficients, the proposed modifications in the framework produced phone recognition error rates that are not just lower than those got using the original framework, but also lower than those got using the DRN framework. It should also be noted that the improvement in the recognition scores is significant in each case. Furthermore, we can see that in the DCRN framework, the lowest error rates were attained when we initialised the weights in the feature extraction layer based on the feature sets we experimented on in Chapter 2, the best performance being achieved when we used Gabor filters.

Table 3.10: Phone error rates (PER) got on the core test set of TIMIT contaminated with Band-limited and Pink noise, with different SNR values (reported error rates are the average of 10 independently trained neural nets).

| Noise type | SNR | Initial Filter weights | Original framework | DRN framework | DCRN framework |
|---|---|---|---|---|---|
| Band-limited | 10 dB | Gabor | 52.53% | 52.79% | 49.75%* |
| | | 2D DCT | 52.26% | 51.48% | **49.46%***|
| | | Random | 53.65% | 53.86% | 50.32%* |
| | 20 dB | Gabor | 39.42% | 41.89% | 38.10%* |
| | | 2D DCT | 39.29% | 41.08% | **37.48**%* |
| | | Random | 40.68% | 41.42% | 37.70%* |
| Pink | 10 dB | Gabor | 64.03% | 61.61%* | 61.45%* |
| | | 2D DCT | 64.76% | 62.81% | **61.33**%* |
| | | Random | 65.4% | 62.96% | 61.87%* |
| | 20 dB | Gabor | 43.13% | 39.95%* | **39.89**%* |
| | | 2D DCT | 44.30% | 40.72% | 40.09%* |
| | | Random | 45.02% | 40.34%* | 40.59%* |

### 3.4.3 Experiments on noise contaminated speech using the TIMIT database

#### Experiments on Speech Contaminated with Artificial Noise

Here, the test set was contaminated with artificial noise types. Then models trained on clean speech were evaluated on the resulting sets. The results of these experiments are summarised in Table 3.10. For better visualisation, in each row the lowest error rate (and those not significantly different from it) is highlighted with an asterisk. In the case of the DCRN framework, the best performing initialisation scheme for each noise type and SNR is in bold.

First, let us examine the question of how the phone recognition error rates change with the introduction of the DCRN framework. We can see that with band-limited noise the framework with the three new modifications significantly outperforms every other framework tested. The case of pink noise is not much different from this, as here in each case where there was one framework that performed significantly better than all the others, that framework was the DCRN. Looking at the difference between the results obtained via the various initialisation schemes, we can see that in each case the best result was obtained by the DCRN framework when using the 2D DCT coefficients or the Gabor filter set for this purpose.

Table 3.11: Phone error rates (PER) got on the core test set of TIMIT contaminated with Babble, Factory and Volvo noise, with different SNR values (reported error rates are the average of 10 independently trained neural nets).

| Noise type | SNR | Initial Filter weights | Original framework | DRN framework | DCRN framework |
|---|---|---|---|---|---|
| Babble | 10 dB | Gabor | 54.13% | 50.61% | **49.70%*** |
| | | 2D DCT | 55.04% | 51.33% | 50.60%* |
| | | Random | 56.25% | 50.44%* | 51.46% |
| | 20 dB | Gabor | 36.95% | 33.09% | **31.80%*** |
| | | 2D DCT | 37.61% | 34.17% | 32.17%* |
| | | Random | 37.64% | 33.42% | 32.62%* |
| Factory | 10 dB | Gabor | 58.06% | 54.73%* | 54.91%* |
| | | 2D DCT | 58.68% | 55.58% | **54.89%*** |
| | | Random | 58.85% | 54.60%* | 55.13% |
| | 20 dB | Gabor | 38.86% | 35.34% | **34.82%*** |
| | | 2D DCT | 39.47% | 36.32% | 35.32%* |
| | | Random | 39.94% | 35.52%* | 35.58%* |
| Volvo | 10 dB | Gabor | 32.61% | 28.26% | **27.52%*** |
| | | 2D DCT | 32.79% | 29.19% | 27.86%* |
| | | Random | 32.31% | 28.59% | 27.64%* |
| | 20 dB | Gabor | 28.48% | 25.53% | **24.75%*** |
| | | 2D DCT | 28.70% | 26.00% | 24.92%* |
| | | Random | 29.07% | 25.54% | 24.89%* |

### Experiments on Speech Contaminated with Real-Life Noise

Lastly, our experiments were repeated on speech contaminated with noise types arising from real-life applications. The results given in Table 3.11 tell us that with Volvo noise the results got with the DCRN framework surpass even the best results got with the original framework or the DRN framework regardless of the SNR or the initialisation scheme used. The picture is not so clear with babble and factory noise, but in most cases with these settings as well, the best results were got by applying the DCRN framework.

We can also see that the DCRN framework provides the lowest error rates when the weights in the filtering layer are initialised based on either 2D DCT coefficients or Gabor filters.

## 3.5 Conclusions

*In this chapter, we proposed an algorithm for the joint training of spectro-temporal features and neural networks. First, the algorithm was tested in phone recognition tasks on the TIMIT speech database, and the "Szeged" Hungarian broadcast news corpus. The results confirmed that joint optimisation does indeed result in a better recognition performance than that got by separate feature extraction and acoustic modelling. The results also indicate that this method leads to significantly better recognition results as well as to better cross-database and cross-language performance than those got via the automatic feature selection algorithms.*

*After the preliminary experiments, two further modifications were proposed in the joint training of spectro-temporal features and neural networks. One was the addition of further hidden layers with neurons using the rectifier activation function in the network, turning it into a Deep Rectifier Neural Net. The other was the introduction of convolution into this DRN structure. The effect of these further modifications was also tested in phone recognition tasks carried out on the clean and noise contaminated version of the TIMIT speech database. The results of these experiments show that while the first modification brought about an improvement in the error rates compared with the original framework, the best performance was achieved when both refinements were applied. As we saw, the error rates provided by the framework that applied both modifications were significantly better than those got using our original framework, regardless of the filter set applied or the noise used to contaminate the speech signal, and in the majority of cases (both in clean and noise contaminated speech) it also yielded significantly better results than those got using the framework that applied Deep Rectifier Neural Networks without convolution.*

*The latter set of experiments in sections 3.3 and 3.4 lead to two additional deductions. First, the experiments with the DRN framework confirmed that by extending the scope of the backpropagation algorithm to the feature extraction layer, and thus training the feature coefficients along with the other weights of the neural network, it is possible to attain better recognition rates. Second, the experiments on both the DRN and the DCRN framework confirmed that the proper initialisation of with the filter coefficients better recognition results can be achieved. That is, even in the case of the DRN framework it rarely occurred that the best results were got using randomly initialised coefficients, while in the case of the (better performing) DCRN framework this did not occur even once. It should also be remarked that out of the two feature sets used, Gabor filters overall achieved a slightly better performance than 2D DCT coefficients did.*

# Chapter 4

# The Multi-Band Processing of Speech using Spectro-Temporal Features

*Originally, spectro-temporal feature extraction and multi-band processing were both based on observations on human speech perception, and were both designed to increase the robustness of speech recognisers. And while these methods have been in use for a long time now, few attempts have been made to combine them, despite their evident compatibility. This is why in this chapter we will focus on integrating spectro-temporal processing into multi-band speech recognition.*

*First, we examine a simple method where spectro-temporal features derived from different regions of the frequency domain are processed separately by dedicated neural nets, then the output of these neural nets is combined in the input of an additional neural net. Here, the feature sets introduced in Chapter 2 will be used for spectro-temporal processing, while for classification, both conventional MLPs, and DNNs will be used. The assessment of the method will be carried out on a phone recognition experiment of the TIMIT corpus, using both clean and noise contaminated speech.*

*In the second part, we will introduce a similar method that combines multi-band processing with the joint optimisation of spectro-temporal features and neural net classifiers. We will assess the performance of this method on the word recognition task of the Aurora-4 speech corpus, using ARMA features as the spectral representation.*

## 4.1 Introduction

In multi-stream speech processing, information from several separate sources is combined [161]. A special case of this is multi-band processing (first described by Duchnowski [47]), where different frequency regions are treated as separate sources. In this paradigm the input is decomposed into spectral bands, then after independent processing of the bands (which usually entails a partial recognition being performed [29, 70, 82, 96]), information from the different bands is recombined to produce an overall recognition result. The use of this method was motivated by various considerations, such as signal processing [190], the opportunity to exploit the potential of parallel computing systems [160], and human speech perception [7, 51, 127, 193]. Robustness is another motivation [19, 82, 173], because in the case of a noise that is limited to certain bands, recognisers working with features originating from those bands that were unaffected by the noise would also be unaffected (in contrast with recognisers that rely on full-band features). The latter two might sound familiar, which is hardly surprising, as spectro-temporal speech processing was also inspired by human speech perception (see Section 2.1), and the reasoning supporting the expected noise robustness here is the same as that in the same section where we argued for the expected noise robustness of spectro-temporal features.

Moreover, as is indicated by their schematic representation in Figure 4.1, the similarity between the spectro-temporal and multi-band processing does not stop there. In fact, the approach we were following in Chapter 2 (where features extracted from different frequency bands of the full spectrum are later recombined in one feature vector), is sometimes referred to as a branch of multi-band processing, under the label “feature recombination” [34, 75, 173].

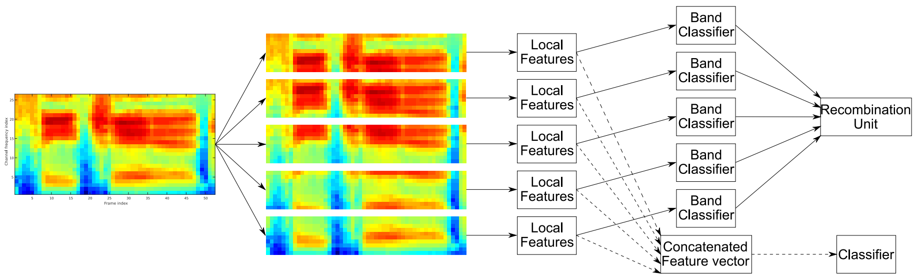

Figure 4.1: Schematic representation of a spectro-temporal processing schemes framework (see Chapter 2) and a typical multi-band processing framework. Where the two methods diverge, their process is represented by dashed/solid lines, respectively.

One would reasonably expect that methods especially designed for processing spectral bands (such as spectro-temporal processing) should produce a more optimal feature set. In most cases, however, when it comes to feature extraction, studies on multi-band processing rely on standard techniques as PLP [70, 82, 96], RASTA-PLP [71, 159] or MFCC [29, 173]. While rare papers attempt to combine spectro-temporal processing with the multi-stream approach, these mostly concentrate on separating the streams based on the properties of the filters, rather than one based on the frequency domain these filters were applied on for extracting the necessary features (i.e. multi-band approach) [148, 149, 238].

### 4.1.1 Processing and Recombination of the Bands

The reader might find the description of multi-band processing at the beginning of Section 4.1 too vague. That is because it is a blanket term describing a wide range of methods with quite different properties. The number of subbands used, for example, can range from 2 to 22 [43]. Another question is the overlap of the subbands. While in many cases the separation of subbands is carried out using rectangular windows with no overlap (apart from that originating from the filter-banks overlapping) [19, 29, 47, 82], overlapping rectangular or smooth windows can also be used [228]. And once the subbands have been selected, there are several decisions to be made on processing them and on combining the results. Subbands may for example be processed using GMMs [69], HMMs [29, 173], and HMM/ANN hybrids [82]. But typically this is performed via ANNs [20, 71, 159, 160] (here we also followed this practice of using ANNs). And after we have processed the subbands, we must make a decision about how much of the resulting information we should use. Do we use only the label of the winning hypothesis, the order of different hypotheses, or the probability predictions [29]? Let us now assume that the recombination is carried out based on the posterior estimates of ANNs (or the estimates of any other method we chose to process subbands with). Several methods still can be chosen for recombination, ranging from simple fixed linear combinations [29, 82] to sophisticated methods that try to dynamically assess the reliability of the bands [19, 69, 148, 167, 173], including ANNs [159, 160]. In the experiments we used neural nets for recombination as well, as our investigation of multi-band processing was partially inspired by the recent renaissance in ANN-based recognition (especially with DNNs [84]). In our opinion one of the main reasons to use ANNs in the recombination is that the recombination mechanism should be nonlinear to exploit in the best possible way the inherent complex correlations between the bands as argued in [19]. This is also confirmed by the experiments of Hermansky et al. in which ANN-based merging consistently outperformed linear combination counterparts in different experimental configurations [82, 148].

## 4.2 Separate feature extraction and neural net training

Let us first examine the effect of multi-band processing in the case where spectro-temporal feature extraction and neural net training is executed in two separate steps. In this approach, we first apply the 2D DCT/Gabor filters (see Chapter 2) on the spectral representation. Then we separate the resulting features into groups based on the frequency domain of their origin and independently train neural nets for each group. After training the first neural nets, their outputs are concatenated and fed into a recobminational neural net. Note the similarity between this method and the combination of MFCC and 2D DCT features earlier (see Section 2.2.2). The difference is in the features to be combined: while during the experiments described in Section 2.2.2 the combination merged different features originating from the same frequency domain (i.e. multi-stream processing), here, the same type of features originating from different frequency domains are combined (i.e. multi-band processing).

### 4.2.1 Experimental settings

As primary and secondary feature extraction was carried out according to that described in Chapter 2, the settings we will discuss here only involve the neural nets used. In the experiments described below, four different neural net architectures were applied. The only common features are the hidden layers using sigmoid neurons, and the output layer consisting of 39 units.

- FC: the neural nets employed for the “feature recombination” experiments. Their hidden layer consisted of 4000 neurons using sigmoid activation, while their output layer consisted of 39 neurons using softmax nonlinearity. These networks used 9 neighbouring frames for their input.
- MB small: neural nets utilised in the first version of multi-band experiments. In the case of multi-band processing, two types of neural nets were used; one for the processing of individual bands, and one for the recombination of the outputs obtained from the previous nets. Here, to ensure comparability with the feature recombination method (i.e. to ensure that the number of trainable parameters in the different nets combined is not significantly different from the number of trainable parameters in the FC net), the following settings were used: the neural nets trained on the individual frequency bands had one hidden layer of 1000 neurons, while the recombination net had one hidden layer of 4000 neurons, and both used 5 neighbouring frames in their input.

- MB big: in the next model both the nets processing the individual bands, and the recombination nets were replaced by larger neural nets that used 9 neighbouring frames and had hidden layers consisting of 4000 neurons (similar to the neural nets used in the “feature recombination” approach).
- MB deep: here both the neural nets processing the individual bands, and the neural nets responsible for recombination were replaced by deep neural nets. The first task (of processing individual bands) was carried out using DNNs that consist of three hidden layers of 1000 neurons. As for the recombination unit, a net with two hidden layers of 1000 neurons was applied. Again, both nets used 9 neighbouring frames. Note that unlike in Chapter 3, where rectifier neural networks were used for deep learning, networks here were trained using the pre-training algorithm of Hinton et al. [84]. However, as Glorot [64] and Tóth [223] have shown, the main difference between the two is their running time, while their results are similar.

### 4.2.2 Phone recognition experiments on the TIMIT database

Experiments reported here were carried out on the TIMIT phone recognition task, using both clean speech, speech contaminated with artificial noise types, and noise types resulting from real-life applications. Due to the various neural network architectures employed in these experiments, we should be able to examine different important questions using the resulting phone recognition error rates. Three key questions are:

1. How does the transition from the traditional spectro-temporal (or “feature recombination”) approach to the multi-band approach presented here affect the error rates we get?
2. How does the transition from smaller neural nets (where the seven neural nets altogether contain approximately 6 million trainable parameters) to larger neural nets (where the seven neural nets altogether contain approximately 16 million trainable parameters) in the multi-band approach influence the error rates we get?
3. How does the introduction of DNNs in the multi-band approach affect the error rates we get?

Table 4.1: Phone error rates (PER) got on the clean core test set of TIMIT (reported results are the average of 10 independently trained neural nets), and the number of free parameters (in millions) for the different settings. For both 2D DCT and Gabor filters, the best score is highlighted in bold.

| Settings | 2D DCT | Gabor | No. of ANN parameters |
|---|---|---|---|
| FC | 26.85% | 26.78% | ~ 6 |
| MB small | 26.07% | 25.45% | ~ 6 |
| MB big | 24.70% | 24.79% | ~16 |
| MB deep | **23.47%** | **22.81%** | ~17 |

## Experiments on clean speech

Although the biggest gains were expected in noisy conditions, we still found it useful to present results on clean speech as well. The reason for this was the contradictory nature of earlier results: Morris et al. [167] found that multi-band processing had a detrimental effect on speech recognition performance for clean speech (Besacier and Bonastre had similar results for the task of speaker recognition [15]), Bourlard and Dupont [20] concluded that the multi-band approach provides at least a comparable performance in clean speech, and Mirghafori and Morgan [160] found that the multi-band approach can be used to improve the speech recognition accuracy in clean speech as well.

Table 4.1 lists the phone error rates attained on the clean test set. It should be noted that for both the 2D DCT features and the Gabor filter set, the first multi-band setting (MB small) already significantly outperforms the feature recombination (FC) approach. This means that in our experiments the multi-band approach not only **not** harmed the speech recognition performance in clean speech, it even improved it significantly. Then each new setting gave a better recognition accuracy than the one before. And as anticipated, for both feature sets, the DNN approach provided the best recognition accuracy scores (giving an overall 12.62% and 14.83% relative error rate reduction in the case of 2D DCT and Gabor filters, respectively). It is also of interest here that with the FC setting the recognition accuracy scores for the 2D DCT and Gabor features do not differ much; but this was not the case with most multi-band settings, where the scores obtained with the Gabor filters were significantly better than those obtained with the 2D DCT features.

Table 4.2: Phone error rates (PER) got with 2D DCT features and Gabor filters on the core test set of TIMIT, artificially contaminated with band-limited noise and pink noise (reported error rates are the average of 10 independently trained neural nets).

| Feature set | Settings | Band-limited | | Pink | |
|---|---|---|---|---|---|
| | | 20 dB | 10 dB | 20 dB | 10 dB |
| 2D DCT | FC | 36.04% | 47.98% | 44.27% | 65.16% |
| | MB small | 34.45% | 41.21% | 41.97% | 62.50% |
| | MB big | 32.87% | 39.66% | **40.11%** | **61.61%** |
| | MB deep | **30.68%** | **37.21%** | **40.22%** | 63.55% |
| Gabor | FC | 36.41% | 49.13% | 44.26% | 65.76% |
| | MB small | 33.11% | 40.47% | 39.87% | 60.10% |
| | MB big | 31.75% | 38.98% | 39.31% | **59.54%** |
| | MB deep | **29.57%** | **35.93%** | **38.00%** | 61.46% |

### Experiments on speech contaminated with artificial noise

The models trained were also evaluated on the core test set contaminated with artificial noise types, such as pink noise and band-limited noise. The numerical results of these experiments are listed in Table 4.2. As can be seen, the simplest multi-band model (MB small) already outperforms the feature concatenation (FC) approach for each type of noise. Furthermore, in accordance with the suggested advantage of the multi-band approach in narrow-band noise [19, 160], multi-band processing indeed had the biggest gain in the case of band-limited noise, when the signal-to-noise ratio was small (10 dB). Here, average relative error rate reductions of 14.09% and 17.62% were reported with 2D DCT features, and Gabor filters, respectively. It is also true that the relative error rate reduction is bigger for each case here than it was for clean speech, corroborating the noise robustness of the multi-band approach. It can also be seen that in most cases the DNN approach yielded the best scores.

Moreover, just like in the case of clean speech, with the "feature recombination" approach we get comparable phone recognition scores for the 2D DCT and the Gabor features (with 2D DCT features being better in most cases); and for the multi-band settings the Gabor features consistently yielded lower phone recognition error rates.

Table 4.3: Phone error rates (PER) got with 2D DCT and Gabor features on the core test set of TIMIT, artificially contaminated with Babble, Car and Factory noise samples (reported error rates are the average of 10 independently trained neural nets).

| Feature set | Settings | Babble |  | Volvo |  | Factory |  |
|---|---|---|---|---|---|---|---|
|  |  | 20 dB | 10 dB | 20 dB | 10 dB | 20 dB | 10 dB |
| 2D DCT | FC | 37.11% | 53.46% | 29.14% | 32.21% | 40.27% | 58.82% |
|  | MB small | 35.63% | 50.74% | 28.26% | 31.58% | 38.55% | 56.84% |
|  | MB big | 34.01% | 49.13% | 26.57% | 29.83% | 37.24% | 54.98% |
|  | MB deep | **31.88%** | **48.38%** | **24.60%** | **26.83%** | **35.32%** | **54.50%** |
| Gabor | FC | 36.79% | 52.97% | 29.53% | 32.87% | 40.38% | 59.78% |
|  | MB small | 34.77% | 49.69% | 27.54% | 30.70% | 37.19% | 55.71% |
|  | MB big | 33.82% | 48.66% | 26.40% | 29.49% | 36.13% | 54.59% |
|  | MB deep | **31.07%** | **46.66%** | **23.94%** | **25.80%** | **33.82%** | **53.17%** |

### Experiments on speech contaminated with real-life noise

Models trained on clean speech were also evaluated on noise types coming from real-life applications, such as babble noise, Volvo noise and factory noise. The last two were added to the experiments, as Hagen et al. found that on such, more realistic noise types, multi-band processing is only capable of a small improvement [71]. The error rates got from these experiments are summarised in Table 4.3.

As can be seen in Table 4.3, similar to the previous cases, even the worst performing multi-band setting (MB small) significantly outperforms the feature recombination (FC) approach, with an average relative error rate reduction of more than 5%. Furthermore, similar to what we saw in clean speech, the error rates further decreased with each modification introduced. As a result, for real environmental noise, the DNNs provide the lowest error rates in each case. And again, while in the feature recombination (FC) approach Gabor filter-based recognition results in some cases are slightly poorer than the 2D DCT-based results, they take the lead in the multi-band case, where they consistently provide a better recognition performance than that obtained with the use of the 2D DCT features.

Overall we can say that for one, deep networks in most cases gave an improvement in phone recognition accuracy. What is more, contrary to some expectations [71, 96], in each case multi-band processing provided better phone recognition accuracy scores than those obtained with the traditional spectro-temporal feature extraction (i.e. “feature recombination”) method.

### Experiments on the representation of low frequency bins

Because of its highly technical nature, one matter of detail has not yet been discussed, concerning the feature extraction of spectro-temporal processing experiments described earlier. In each of these experiments up to this point, after the primary feature extraction phase but before secondary feature extraction, the filter banks corresponding to the lowest frequency bins were copied and mirrored, in order to avoid the artificial down-weighting of the low frequency bins. During this intermediary phase the number of filter banks to be mirrored depended on the height of the patches used for feature extraction: mirroring was executed in a fashion that allowed the positioning of the central row of the first patch on the lowest mel filter-bank (meaning that for a patch height of seven, three filter banks were mirrored, while for a patch height of nine this value was four – the latter case being shown in Figure 4.2). This technique was employed in our early experiments following the work of Bouvrie et. al [24], and was maintained in later experiments for compatibility and comparability reasons. It stands to reason, however, that the presumption on the necessity and usefulness of this technique should also be tested. Fortunately, the combination of the mirroring technique applied and the overlap of patches extracted meant that the mirroring only affected the patch extracted from the lowest frequency range, and all other patches were extracted just as if there had been no mirroring. This, combined with the fact that features extracted from different frequency bands were processed independently meant that the multi-band approach used in our experiments led us to examine whether mirroring was indeed effective and necessary before the secondary feature extraction phase, as out of the six filter bands only one was affected by mirroring, while the other five were unaltered.

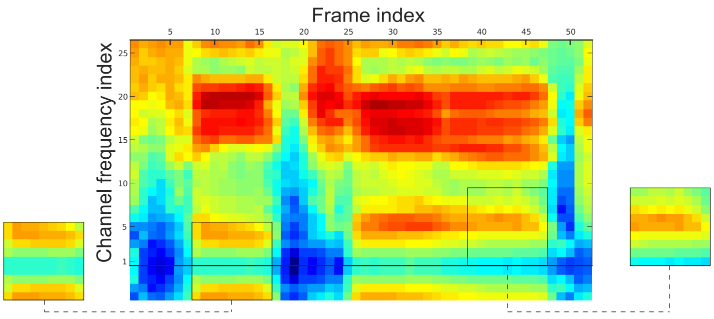

Figure 4.2: A representation of the lowest patch extracted in the presence of mirroring (rectangle on the left hand side), and in the absence of mirroring (rectangle on the right hand side).

Table 4.4: Phone error rates (PER) got on the clean core test set of TIMIT and its noise contaminated versions with different Signal to Noise ratios (SNR) using Gabor filters (reported error rates are the average of 10 independently trained neural nets). Where the difference between the two columns was significant (p < 0.05) the lower error rate is highlighted in bold.

| Speech type | Noise type | SNR | Mirroring | No Mirroring |
|---|---|---|---|---|
| Clean | – | – | **22.81%** | 23.01% |
| Noise contaminated | Babble | 10 dB | **46.66%** | 47.19% |
| | | 20 dB | 31.07% | 31.07% |
| | Volvo | 10 dB | 25.80% | 25.91% |
| | | 20 dB | **23.94%** | 24.15% |
| | Factory | 10 dB | 53.17% | **52.34%** |
| | | 20 dB | 33.82% | **33.55%** |
| Average | | | 33.90% | 33.89% |

For the sake of a comparison of results obtained from the mirrored spectrogram and its counterpart without mirroring, we repeated the training of the deep recombination nets from the MB deep settings with the only modification that here, the recombinatonal net only used the output of five filter bands (those that were not affected by the mirroring). The neural networks trained in this fashion on the TIMIT database were evaluated on the clean core test set of TIMIT, as well as the versions of the core test set contaminated by real-life noise types.

Because in the previous experiments features extracted by Gabor filters provided convincingly lower error rates than 2D DCT features, here only the former feature set was applied. The results of these experiments, along with the results obtained with the deep recombination net using the information from all six frequency bands are shown in Table 4.4.

We can see in the last line in Table 4.4 that as regards recognition accuracy there is virtually no difference between the two approaches. However in terms of the parameter number, the situation is different: the combined number of trainable parameters when using all six frequency bands in recognition is close to 17 million ($\sim$16.9), while in the case where only five frequency bands are used this number is closer to 14 million ($\sim$14.2), meaning that in the latter case we can attain practically the same recognition results by using about two and a half million fewer parameters. This suggests that mirroring the lowest frequency bins might not be necessary, hence in later experiments this method was not applied.

## 4.3 Joint training of spectro-temporal features and neural nets

The results of the experiments described in the previous section demonstrate that the multi-band approach already leads to a better recognition accuracy in spectro-temporal speech processing when the feature extraction and neural net training steps are carried out separately. However, in Chapter 3 we saw that it was possible to achieve better results when these two steps were combined. Thus it is also necessary to examine the effect of the multi-band approach within the framework of joint training. We did so by applying the neural net structure described in Section 3.4 to get the scores of individual frequency bands. Afterwards, the scores provided by these neural nets were concatenated and used as the input of a simple recombination net.

However, to highlight the generality of these techniques, the experiments to determine the performance of the multi-band approach in the joint training framework were not conducted on the TIMIT speech corpus, but on the larger Aurora-4 corpus. And as the lowest error rates on the Aurora-4 word recognition task (in the case of mismatched conditions in training and testing) were reported with the use of the ARMA primary features [57], we used the ARMA features as the spectral representation in the experiments.

### 4.3.1 Neural net structure

The shift from full-band processing to multi-band processing as well as the change in task and input representation naturally required modifications in the neural net structure described in Section 3.4. Before discussing the results, let us first examine these changes item by item.

The first difference is the difference in input. Here, following Ganapathy [57], not only the 39 frequency channels of the ARMA spectrogram were used, but a derivative of this spectrum (described in Section 1.3.1) of the same size was also applied. To accomodate the higher dimensionality of the input, instead of the six feature extraction sub-layers, 16 such sub-layers were used (8-8 on the ARMA spectrogram and its derivative, respectively). These 16 sub-layers provided the features for 4 independent frequency bands.

Another important difference was the number of targets used during training. Here, to ensure the competitiveness of our results, 1997 context-dependent tristate phone targets were used. This means that even if the number of independent bands were limited to four, the dimensionality of the recombination nets input would be a multiple of 7988 (depending on the number of neighbouring

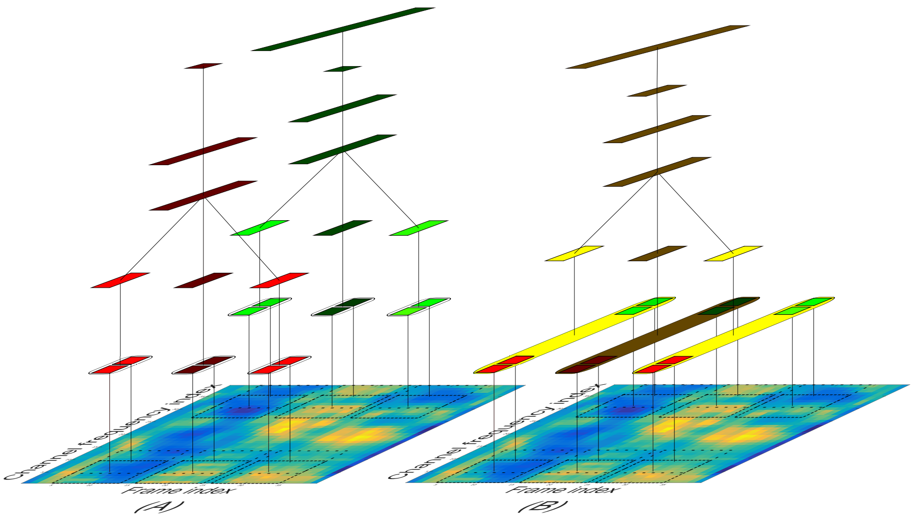

Figure 4.3: Schematic representation of multi-band processing (A) and traditional, full-band processing (B) for the joint training of neural networks and spectro-temporal filter coefficients.

frames used). To reduce this dimensionality, two techniques were examined: in one case the neural networks operating on the individual bands were trained on 42 context-independent monostate phone models, while in the other case a small "bottleneck" layer was placed in front of the output layer, and the output of this layer was used as the input of the recombinatonal net.

These structures are shown on the left hand side of Figure 4.3 (for the sake of clarity, the derivative spectrum is not displayed). Here, the structure of the "monostate" and the "bottleneck" method are shown on different frequency bands of the input. First, let us examine the details of the "monostate" case, which is shown on the lowest frequency band, with neural net layers shown in different shades of red (here, lighter shades denote the same effect that in Chapter 3 was denoted by transparency, i.e. virtual layers that share their weights with their darker coloured – or in Chapter 3 opaque – counterparts). As can be seen, after the small feature extraction sub-layers there is a bigger layer (with 200 rectifier neurons) that also applies convolution in the time domain, after which two rectifying layers of 1000 neurons are used (without convolution), and the structure ends with an output layer of 42 neurons (corresponding to the 42 monostate phone models). The "bottleneck" structure (shown in green) only differs in its output layer, which is much bigger (containing 1997 neurons), and in a small hidden layer (containing 50 neurons) right in front of it.

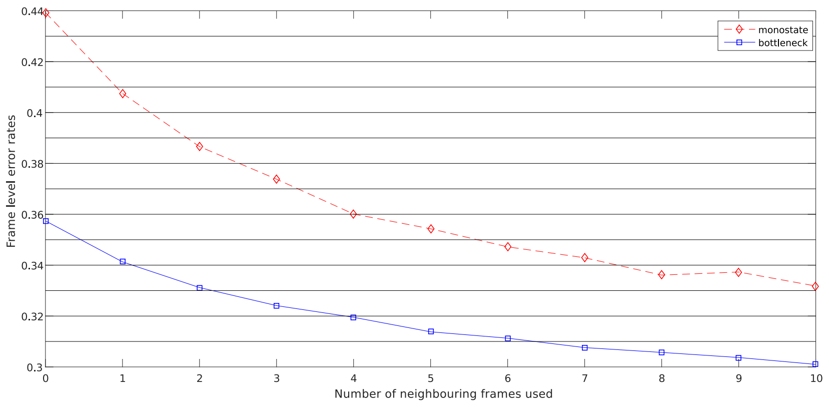

Figure 4.4: Frame-level classification error rates got using different multi-band structures on 10% of the clean Aurora-4 training set as a function of context size (reported scores are the average of 3 independently trained neural nets).

As a baseline, another structure was examined (shown on the right side of Figure 4.3), which only differs from the one described in Section 3.4 in the number of feature extraction sub-layers used, the size of the output layer (1997 neurons), and the addition of a smaller hidden layer (400 neurons, for comparability with the “bottleneck” multi-band structure) in front of the output layer. Otherwise this structure is the same as before. Outputs of the feature extraction sub-layers, regardless of their position in the frequency domain, are processed by the same layer in the next step, using convolution in the time domain. After this convolutional layer the information is processed by conventional, fully connected rectifier layers (containing 2000 neurons) up to the softmax output layer.

### 4.3.2 Recombination net

The output of all the structures displayed in Figure 4.3 was used as the input of a DRN with two hidden layers, each consisting of a thousand neurons, and a softmax output layer consisting of 1997 neurons. In the multi-band approach this net took on the role of the recombination classifier, while in the baseline, full-band case, this net had the same role as the second ANN in the 2-stage system described in Section 2.2.2. As we saw in Section 2.2.2 and Section 4.2, it is beneficial if this recombination net also has the opportunity to use a bigger context. It is not clear, however, how big this context should be. In order to examine this, 3 recombination nets were trained for each context size (between 0 and 10 neighbouring frames on both sides of the current frame) using the outputs of the two multi-band structures on the randomly selected 90% of the clean training set of Aurora-4. The different settings were evaluated based on the frame-level classification error rates on the remaining 10%. The results of these experiments are shown in Figure 4.4.

As can be seen, the “bottleneck” structure consistently outperforms the “monostate” structure, the former resulting in error rates that are 3 to 8 percentage points lower than those obtained by the latter. Although the frame-level results improve even with the biggest context we tested, after considering the flattening of the curve, and the observation that the expansion of the context can have a positive effect on the frame-level results even when its effect is detrimental on word level recognition [178], a context of 8 neighbournig frames (i.e. a context of 4 frames before and after the current frame) was selected for the later experiments (and preliminary experiments using the monostate structure justified the assumption that a bigger context is not beneficial in word recognition performance).

### 4.3.3 Word recognition experiments on the Aurora-4 database

The word error rates obtained using the various joint training structures on the different test sets of Aurora-4 are summarised in Table 4.5. For a better understanding of these results, and to put them in context, in the last column the table also contains the results Sriram Ganapathy got on the word recognition task of the Aurora-4 corpus, using the same input features [57]. To the best of our knowledge, the results of Ganapathy are among the lowest word error rates reported on the Aurora-4 corpus without the use of noise corrupted data in training.

Comparing the “bottleneck” and “monostate” columns, we see that the difference observed on the validation set manifests itself on the test set as well. The error rates got using the “bottleneck” structure are not only lower on average than those got using the “monostate” structure, but the former also significantly outperforms the latter on all but three test sets.

We can make additional interesting observations by comparing the feature recombination column with the “bottleneck” and “monostate” columns. There is a duality feature here. Namely, the relative performance of these approaches is quite different in the case of the test sets recorded with the Sennheiser microphone, and in the case of the test sets recorded with the various secondary microphones. On the test sets recorded with the Sennheiser close talking microphone, the “bottleneck” structure outperforms the baseline feature-recombination method by only a small margin, while the “monostate” structure performs even worse than that baseline. On the test sets recorded with the

Table 4.5: Word Error Rates (WER) on the test sets of the Aurora-4 corpus (reported scores are the average of 3 independently trained neural nets).

| Microphone | Noise | FC | Monostate | Bottleneck | Ganapathy[57] |
|---|---|---|---|---|---|
| Primary microphone | Clean | 3,9% | 3,8% | 3,7% | **3,0%** |
| | Car | 6,4% | 6,1% | 5,8% | **5,0%** |
| | Babble | 13,6% | 13,9% | **12,6%** | 13,0% |
| | Restaurant | 17,7% | 18,8% | **17,2%** | 17,3% |
| | Street | **14,0%** | 15,5% | **13,7%** | **13,6%** |
| | Airport | 14,0% | 14,5% | **13,5%** | **13,7%** |
| | Train | **14,6%** | 17,1% | **14,5%** | **14,5%** |
| | Average | 12,0% | 12,8% | **11,6%** | **11,4%** |
| Secondary microphone | Clean | 14,3% | **11,9%** | **11,6%** | **11,7%** |
| | Car | 21,7% | 18,6% | **17,7%** | 18,4% |
| | Babble | 30,1% | 29,8% | **27,5%** | 29,6% |
| | Restaurant | 30,6% | 31,9% | **29,6%** | 31,1% |
| | Street | 29,8% | 30,9% | **27,8%** | 28,3% |
| | Airport | 29,4% | 29,4% | **27,3%** | 29,5% |
| | Train | 30,3% | 31,0% | **27,3%** | 29,1% |
| | Average | 26,6% | 26,2% | **24,1%** | 25,4% |
| Average | | 19,3% | 19,5% | **17,8%** | 18,5% |

secondary microphones, however, both multi-band methods outperform the baseline, and the advantage of using the "bottleneck" approach is significant in each case, and the average error rate reduction provided by it increases from 3.7% to 9.3%. This suggests that while the performance of speech recognition increases in the presence of additive noise, we can benefit from it even more when the microphones used for training and test data are different.

We may arrive to the same conclusions after comparing the results of the "bottleneck" framework with those of Ganapathy. While the results obtained by our multi-band processing method are not substantially better than those of Ganapthy on the test sets recorded with the Sennheiser microphone (in some cases they are even worse), this is not the case for noise contaminated test sets recorded with the secondary microphones, where the use of the multi-band approach results in convincingly lower error rates. It should be added here that despite the less convincing performance on the first set of tests, the application of the "bottleneck" structure overall leads to a relative error rate reduction of close to 4% compared to that of Ganapathy.

## 4.4 Conclusions

*In this chapter, we examined the multi-band approach for spectro-temporal speech processing. First, we demonstrated that the results attainable with the use of 2D DCT coefficients and Gabor filters can be improved when speech processing is conducted in a multi-band framework. Despite expectations to the contrary, the multi-band approach proved to be advantageous in clean speech as well as speech contaminated with various artificial and real-life noise types. Furthermore, as expected, these results improved even further when deep learning was included in the process. As an extension to these experiments, we examined the utility of the practice of copying and mirroring the lowest frequency bands in the spectral representation. During these experiments we found that this practice had no obvious beneficial effect on the recognition rates. Because of this finding, we abandoned this method in later experiments of the study.*

*In the second part of this chapter we discussed the possibility of applying the multi-band approach for the joint training of spectro-temporal features and the DRN acoustic model. The performance of the resulting method combining various techniques for noise robust speech recognition was evaluated on the Aurora-4 continuous speech recognition task. The resulting word error rates indicated that if suitably included in the joint framework, the multi-band approach can measurably increase the robustness of speech processing, especially in a channel mismatched situation. Furthermore, it was also demonstrated that this method not only provides a relative improvement compared to the earlier version of the joint framework that applied the feature-recombination approach, but it also yields results comparable to the state-of-the art, on the Aurora-4 clean training task. In fact, to the best of our knowledge, at the time of their original publication these results were among the best reported word error rates on this task without speaker adaptation.*

*It has also become clear that the proper adaptation of these methods is crucial. And an additional lesson from these experiments was the realisation of how hard it can be to find a proper recombination for the values origination from the different frequency bands. For this, our goal for the next phase of this study was to find an approach that would allow an independent processing of frequency bands and would also bypass this problem.*

# Chapter 5

# Band dropout

*In this concluding part of the study, core ideas of the previous chapters are amalgamated and further developed to demonstrate that beyond providing relative improvements, these methods are also capable of yielding competitive results. To achieve this goal, the basic idea of spectro-temporal processing (see Chapter 2) was combined with the idea of handling sub-bands in some way independently (see Chapter 4) in a joint framework for which the basics were laid down in Chapter 3.*

*Before introducing the band dropout method in the joint framework however, it might be beneficial to revisit the basic parameters of this framework, in order to examine whether it is capable of providing competitive results in itself. This will be described first in this chapter, where we reexamine the performance of the joint framework using the TIMIT and Aurora-4 speech databases. We compare the recognition scores the new framework can now attain with those got earlier, as well as with scores found in the speech recognition literature.*

*In the second part this fine-tuned framework will be supplemented by a method partially inspired via multi-band processing. In the band-specific dropout (in short, band dropout) method during the training phase, the input from certain frequency bands in the neural net are the subject of input dropout. In Section 5.2 we will examine the effects of this method on the Aurora-4 task. We will compare the results obtained using this method with those obtained with no dropout as well as standard input dropout. We shall also compare our results to those found in the speech recognition literature, and demonstrate that the proposed method is capable of providing state-of-the-art speech recognition results.*

## 5.1 Revisiting the joint training framework

During the experiments which sought to optimise the filter sizes used for spectro-temporal feature extraction described in Section 2, only one aspect of the filter size was taken into consideration, namely the "technical" size of the filter; this is how many filter banks and frames are covered by its height and width, respectively. There is another aspect of filter sizes, however, namely the frequency range and time interval they cover (i.e. the physical size of filters). In earlier experiments it was taken for granted that a filter with a certain technical size would have a certain physical size. This assumption hinged on the supposition that the spectral representation in each case would be produced using the same parameter settings. But this is not necessarily the case. Upon revisiting the parameters of the joint training framework, we experimented with varying the physical size of filters just with the modification of parameters used in creating the spectral representation, and without modifying their technical size. To understand these experiments, however, certain parameters and notational conventions have to be introduced or reintroduced.

### 5.1.1 Parameters and notations

Important parameters to be discussed include the parameters applied in constructing the spectral representation, and the parameters used in extracting the patches for processing. Lastly, as patches are not utilised independently but in bigger groups in a neural net framework, some parameters corresponding to the neural nets should also be examined.

#### Spectral representation

Creating a mel-scale spectral representation, or Spectrogram ($\mathbf{S}$) from the digital speech signal requires (among others) the following important parameters:

- $\mathbf{S}^{\delta}_{\mathbf{W}}$: the frame size (in milliseconds) applied in getting the S spectral representation.
- $\mathbf{S}^{\nu}_{\mathbf{W}}$: the frame shift (in milliseconds) applied in creating the S spectral representation.
- $\mathbf{S}^{\#}_{\mathbf{F}}$: the number of filters used in the filterbank.

### Neural net and convolution

In earlier versions of the joint framework (see Chapter 3) two key modifications were included. First, as is common nowadays [64, 222], ReLUs were incorporated into the framework (see Section 3.3). This means that neurons in the hidden layers apply the rectifier activation function on their input, instead of the traditional sigmoid or hyperbolic tangent activation function. Another modification was that following Veselý et al. [231], the neural nets employed applied convolution (in the time domain) in some of their layers (see Section 3.4). In brief, convolutional layers can be most easily understood if we imagine them as separate layers that share their weights. This means that when a layer applies convolution, the number of weights does not change, but the number of inputs and outputs multiplies just as if we had several separate layers. Hence the most important parameters and notations describing a given layer $L$ are the following:

- $\mathbf{L^I}$: The Input of layer $\mathbf{L}$.
- $\mathbf{L^O}$: The Outpuf of layer $\mathbf{L}$.
- $\mathbf{L^{\#_n}}$: The number of neurons in layer $\mathbf{L}$ (this parameter will also be referred to as the size of layer $\mathbf{L}$).
- $\mathbf{L^{W_i}}$: The weight vector corresponding to the *ith* neuron of layer $\mathbf{L}$ $(0 < i \leq \mathbf{L^{\#_n}})$
- $\mathbf{L^{\#_p}}$: The number of free parameters in layer $\mathbf{L}$ (commonly determined using the size of the $\mathbf{L^I}$ parameter and the value of the $\mathbf{L^{\#_n}}$ parameter).
- $\mathbf{L_{C_n}}$: If layer $\mathbf{L}$ applies convolution in the time domain (or in other words the L layer is a convolutional layer), and gets its input from X distinct locations in time, layer $\mathbf{L}$ – as it was mentioned earlier – can be thought of as X number of separate layers. In this scenario, $\mathbf{L_{C_n}}$ denotes the n-th such layer (where $1 \leq n \leq X$), while $\mathbf{L^I_{C_n}}$ and $\mathbf{L^O_{C_n}}$ denote the input and output of this layer, respectively. As the weights are shared over these virtual layers, the weights of each of these X number of layers will still be denoted by $\mathbf{L^W}$ (since $\mathbf{L^W_{C_1}} = \mathbf{L^W_{C_2}} = ... = \mathbf{L^W_{C_X}}$). And as a consequence of the weights being shared, the number of neurons are also shared over these layers (meaning that the size of these layers is the same – $\mathbf{L^{\#_n}_{C_1}} = \mathbf{L^{\#_n}_{C_2}} = ... = \mathbf{L^{\#_n}_{C_X}}$), and the number of neurons in each of these X layers will be denoted by $\mathbf{L^{\#_n}}$.

## Patches and filters

The basic parameters related to the various patches ($\mathbf{P}$) used in the filtering phase are as follows:

- $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{P}}}$: The physical size of patch $\mathbf{P}$ in the time domain. It stands for the time span covered by the patch. This parameter will be measured in milliseconds.
- $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{P}}}$: The physical size of patch $\mathbf{P}$ in the frequency domain. It stands for the range of frequencies covered by the patch. It would be intuitive to measure this in Hertz, but different mel filter channels cover different ranges on the Hz scale, meaning that a patch of a certain size in the low frequency range would cover a broader frequency range (in Hz) than the same patch in the high frequency range. Because of this the physical size of patches in the frequency domain will be measured on the mel scale, and information about this will be given in mel units.
- $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{t}}}$: The "technical" size of patch $\mathbf{P}$ in the time domain. It denotes how many frames are covered by the patch. (Note that $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{t}}}$ can be determined from the $\mathbf{S}_{\mathbf{W}}^{\nu}$, $\mathbf{S}_{\mathbf{W}}^{\delta}$, and $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{P}}}$ parameters. Conversely, $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{P}}}$ can be found using the $\mathbf{S}_{\mathbf{W}}^{\nu}$, $\mathbf{S}_{\mathbf{W}}^{\delta}$, and $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{t}}}$ parameters.)
- $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$: The "technical" size of patch $\mathbf{P}$ in the frequency domain. It tells us how many mel-channels are covered by the patch. (Note that $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$ can be determined from the $\mathbf{S}_{\mathbf{F}}^{\#}$ and $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{P}}}$ parameters. Conversely, $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{P}}}$ can be determined using the $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$ and $\mathbf{S}_{\mathbf{F}}^{\#}$ parameters.)
- $\overline{\mathbf{P}}$: A simple notation for patch $\mathbf{P}$ (which is a two-dimensional matrix) when it is given in vector form.
- $\overline{\mathbf{P}}^{\delta_{\mathrm{t}}}$: The length of vector $\overline{\mathbf{P}}$. It can be calculated by applying the following formula: $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{t}}} \cdot \mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$

Since patches are limited in frequency, multiple patches are used in order to cover the whole frequency domain. This also means that other parameters need to be defined to explain this:

- $\mathbf{P}_{\mathbf{f}}^{\#}$: The number of patches used to cover the whole frequency domain.
- $\mathbf{P}_{\mathbf{f}}^{\nu_{\mathrm{P}}}$: the proportion of overlap between neighbouring patches in the frequency domain (note that assuming the overlap to be constant $\mathbf{P}_{\mathbf{f}}^{\nu_{\mathrm{P}}}$ is determined by the $\mathbf{S}_{\mathbf{F}}^{\#}$, $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$, and $\mathbf{P}_{\mathbf{f}}^{\#}$ parameters, and conversely $\mathbf{P}_{\mathbf{f}}^{\#}$ is determined by the $\mathbf{S}_{\mathbf{F}}^{\#}$, $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$, and the $\mathbf{P}_{\mathbf{f}}^{\nu_{\mathrm{P}}}$ parameters).

As convolution is applied in the time domain, at any given time multiple patches will be used by the neural net input not only in the frequency domain (as we saw above), but in the time domain as well. This means that another set of parameters will be necessary to describe this aspect. These parameters are as follows:

- $\delta_{\mathbf{T}}$: The time span (in milliseconds) collectively covered by the multiple patches used in the input of the neural net during the examination of any given frame.
- $\mathbf{P_t^{\#_i}}$: The number of patches with a physical size of $\mathbf{P_t^{\delta_P}}$ needed to cover the $\delta_{\mathbf{T}}$ time span, assuming that the consecutive patches used are immediate neighbours.
- $\mathbf{P_t^{\nu_P}}$: The proportion of overlap between consecutive patches used in the time domain.
- $\mathbf{P_t^{\#}}$: The number of patches with a physical size of $\mathbf{P_t^{\delta_P}}$ needed to cover $\delta_{\mathbf{T}}$ time span, assuming the consecutive patches used are not immediate neighbours, and the overlap in the time domain between consecutive patches is $\mathbf{P_t^{\nu_P}}$. (Note that $\mathbf{P_t^{\#}}$ can be calculated using $\mathbf{P_t^{\delta_P}}$, $\delta_{\mathbf{T}}$, and $\mathbf{P_t^{\nu_P}}$ parameters; and conversely any of these four parameters can be calculated using the other three.)
- $\mathbf{P_t^{\nu_t}}$: The step size used in order to maintain a $\mathbf{P_t^{\nu_P}}$ overlap between consecutive patches used in the time domain (as the notation suggests, this is the "technical" side of the overlap, which is expressed in frames, while the "physical side" - the percentage of overlap $\mathbf{P_t^{\nu_P}}$ - can be expressed in milliseconds as well).
- $\mathbf{P_{ij}}$: If we imagine all the patches the system uses at any given time as a matrix, where the patches coming from the same time-frame make up a column of the matrix, and patches coming from the same frequency-domain make up a row of the matrix, $\mathbf{P_{ij}}$ is the patch that is in the i-th row ($0 < i \leq \mathbf{P_f^{\#}}$) and j-th column ($0 < j \leq \mathbf{P_t^{\#_i}}$) of this matrix. Strictly following this notation would mean that the $\overline{\mathbf{P}}^{\delta_t}$ parameter introduced earlier should be denoted as $\overline{\mathbf{P}}_{\mathbf{ij}}^{\delta_t}$, but as in our experiments patches coming from different time-frames and frequency domains have the same size, the notation introduced earlier will be used.

## 5.1.2 Neural Networks

Earlier we discussed the structure of the joint framework in a way that was sufficient for our purposes in Chapter 3. To understand these investigative experiments better, however, a more technical discussion is necessary, which is facilitated by the formalism introduced above. For a digestible representation, here the description of the joint framework will be broken down into the description of neural net layers with different functionalities.

### Filtering layers

Each sub-layer responsible for filtering ($\mathbf{F_i}$) gets its input patches from a given frequency domain (or band, using a different nomenclature). This means that the number of filtering sub-layers will also be determined by the $\mathbf{P_f^{\#}}$ parameter which describes the number of patches applied in the frequency domain (and consequently the number of bands the process divides the full frequency domain into). Let us first recall how the output of a neuron in the filtering layer was described in Section 3.1. If we reformulate this formula with vectors in mind, using the formalism introduced above, the formula for the output on a non-specified neuron in one of the filtering sub-layers is:

$$o = \sum_{x=1}^{\overline{\mathbf{P}}^{\delta_\mathbf{t}}} \overline{\mathbf{P}}[x] \cdot \mathbf{F}^{\mathbf{W_k}}[x] + b. \tag{5.1}$$

This formula is a bit more complicated if we examine a specific $F_{iC_j}$ filtering layer ($0 < i \leq \mathbf{P_f^{\#}}$ and $0 < j \leq \mathbf{P_t^{\#_i}}$), and its neurons. For the kth neuron in the layer ($0 < k \leq \mathbf{F_i^{\#_n}}$[1]) the output can be calculated using the following formula:

$$o_{kj} = \sum_{x=1}^{\overline{\mathbf{P}}^{\delta_\mathbf{t}}} \overline{\mathbf{P_{ij}}}[x] \cdot \mathbf{F_i^{W_k}}[x] + b_k, \tag{5.2}$$

meaning that the input and output of the sub-layer is:

$$\mathbf{F^O_{iC_j}} = [o_{1j}, ..., o_{\mathbf{F_i^{\#_n}}j}], \tag{5.3}$$

$$\mathbf{F^I_{iC_j}} = \overline{\mathbf{P_{ij}}}. \tag{5.4}$$

[1]As in our experiments each filtering sub-layer contained the same number of neurons (i.e. $F_1^{\#n} = F_2^{\#n} = F_3^{\#n}...$), this value later will be denoted by the symbol $F^{\#n}$

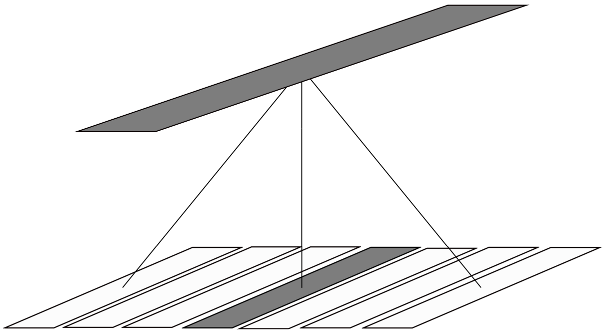

Figure 5.1: Illustration of the pooling described in Eq. 5.6. It shows the case where $\mathbf{P_t^{\#_i}} = 7$, and $\mathbf{P_t^{\nu_t}} = 2$.

### Convolutional layers

After applying the sub-layers responsible for filtering, additional convolutional layers are applied, the last of these layers being the Bottleneck ($\mathbf{B}$) layer. What is important here is the input of the first such layer, which is given by the following formula:

$$\mathbf{Conv^I_{C_j}} = [\mathbf{F^O_{1C_j}}, ..., \mathbf{F^O_{P_f^{\#}C_j}}], \; where \;\; 0 < j \leq \mathbf{P_t^{\#_i}} \tag{5.5}$$

### Fully connected rectifier layers

Following the Bottleneck layer, one or more simple rectifier layers are used (the middle layers, which are referred to as $\mathbf{M_1}$, $\mathbf{M_2}$, etc). What is important here is the input of the first such layer, which is responsible for pooling. In particular, without this layer we would not be able to call the network convolutional. Pooling here is accomplished by the way this layer utilises the output of the previous layer in its input:

$$\mathbf{M^I_1} = [\mathbf{B^O_{C_1}}, \mathbf{B^O_{C_{1+P_t^{\nu_t}+1}}}, ..., \mathbf{B^O_{C_j}}], \; where \;\; j = \mathbf{P_t^{\#_i}}. \tag{5.6}$$

As it can be seen from this formula (and also glancing at Figure 5.1), this layer not only takes care of pooling, but by controlling the number of potential inputs omitted, it also ensures that the required overlap between patches used in the output prevails.

From this point – including additional rectifier layers, and the output layer $\mathbf{O}$ – the neural net behaves just like traditional neural networks, hence we do not need to discuss it further.

## 5.1.3 Optimization of parameters

The purpose of these experiments was to examine the effect of modifying the physical size of patches and thus the filters applied on these patches (without changing their technical size) on the recognition results, and to determine the optimal physical size to be used in later experiments. At first sight this may not seem like a complicted task. However, we noted earlier in Section 5.1.1 that the physical size of patches may depend on various parameters, and that the different parameters are highly dependent on each other, so changing one may affect others as well.

Changing the physical size of patches for one means changing their size in the frequency domain (i.e. $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{p}}}$). We saw earlier that this parameter is determined by the technical size of patches ($\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{t}}}$), and the interval one mel-filter channel covers. The premise of these experiments (i.e. changing the physical size without modifying the technical size) means that here this latter component will be modified. This can be performed by changing the number of filter channels applied to cover the same interval (i.e. changing $\mathbf{S}_{\mathbf{F}}^{\#}$). There is a deeper motivation, however, behind experimentally setting the $\mathbf{S}_{\mathbf{F}}^{\#}$ parameter than the fact that the technical size of patches have already been examined experimentally. For this, let us recall the experiments described in Table 2.3 of Chapter 2. Here, similar results were attained using different numbers of mel filter channels. The selection of 26 channels was largely motivated by the fact that the use of 26 mel filter channels is the most common when creating MFCCs [39, 50, 114, 189, 240] (so much so that it is the default setting in the HTK toolkit [236, 239]), hence this selection ensured the comparability of results obtained with 2D DCT coefficients with the results obtained with MFCCs. In later experiments the number of mel channels used were retained for backwards compatibility, but as we saw earlier in Section 4.2.2 reexamining certain techniques applied might be beneficial. Furthermore, in recent years, when neural nets directly employed the filter bank representation in their input, better results were obtained with 40 channels [68, 162, 223, 237].

A change in physical size of patches may also occur in the time domain, by changing parameter $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{p}}}$. This parameter is determined by the $\mathbf{S}_{\mathbf{W}}^{\nu}$, $\mathbf{S}_{\mathbf{W}}^{\delta}$, and $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{t}}}$ parameters. While $\mathbf{S}_{\mathbf{W}}^{\nu}$ is traditionally between 10 ms and 20 ms [180], interesting experiments have been conducted with values lying outside this interval both above [72, 185] and below [241] it. This is why we opted for this in our experiments, and we modified the physical size of patches in the time domain by modifying the frame shift (the $\mathbf{S}_{\mathbf{W}}^{\nu}$ parameter).

## Parameter settings examined

We have already mentioned that certain parameters such as the technical size of patches and the width of windows used in creating the spectral representation ($\mathbf{P}_{\mathbf{f}}^{\delta_t}$, $\mathbf{P}_{\mathbf{t}}^{\delta_t}$, $\mathbf{S}_{\mathbf{W}}^{\delta}$ - fixed at 9, 9 and 25 ms respectively) should remain the same during these experiments. Naturally, one would not expect the frequency domain covered by the collection of patches to be drastically changed either, regardless of the number of filters used in the filter bank. Similar to the frequency domain, it is also reasonable to expect that the time span covered (i.e. the parameter $\delta_{\mathbf{T}}$, the value of which was approximately 265 ms in earlier experiments) should remain the same, in order to ensure that the potentially emerging difference in recognition scores cannot be attributed to the difference in the size of the context applied. Another parameter that we decided to fix throughout these experiments was the overlap of patches in both the time and frequency domain, which was 55% in both directions. This, paired with the technical size of patches ($9 \times 9$) meant that the step size of patches was to be 4 in both the time and the frequency domain.

As a number of parameters had fixed values throughout these experiments, some parameters of course changed as a result of the modification of the physical size of patches. In the frequency domain, for example, using a higher number of filter channels means that in order to keep the technical size of the patches and their overlap (proportional to their size) constant, the number of patches used ($\mathbf{P}_{\mathbf{f}}^{\#}$) had to be increased. Based on a similar rational, we see that with a new, different frame rate, the number of patches used in the time domain ($\mathbf{P}_{\mathbf{t}}^{\#}$) also had to be changed.

Another aspect to consider was the way parameters that are either explicitly fixed or implicitly restricted influence the values the parameters can realistically take. In the time domain, for example, it is required in the joint framework that the central patch should be centered on the currently examined frame, which means that the number of patches to be used have to be odd. This, combined with what we have discussed so far here means that when we increase or decreasing the frame rate, not every value is possible, just those values can be considered which make it possible to cover a context of approximately $\delta_{\mathbf{T}}$ ms length using an odd number of patches with an overlap of $\mathbf{P}_{\mathbf{t}}^{\nu_t}$. And a similar line of reasoning can be used concerning the physical size of patches in the frequency domain. Thus in reality while the change in the physical size of patches can be better explained by a change in frame rate and the cardinality of the mel filter bank, what was actually driving the change in the parameters was the number of patches used in both the time domain and the frequency domain ($\mathbf{P}_{\mathbf{t}}^{\#}$ and $\mathbf{P}_{\mathbf{f}}^{\#}$).

Table 5.1: Parameter settings used in the time domain.

| parameters | T1 | T2 | T3 |
|---|---|---|---|
| $\mathbf{P}_{\mathbf{t}}^{\#}$ | 5 | 7 | 11 |
| $\mathbf{S}_{\mathbf{W}}^{\nu}$ | 10 ms | 8 ms | 6 ms |
| $\mathbf{P}_{\mathbf{t}}^{\delta_{\mathrm{P}}}$ | ~105 ms | ~89 ms | ~73 ms |

With all this in mind, let us discuss the parameter settings used in our experiments. Three settings were devised to examine the effect of changing the physical size of patches in the time domain. These settings can be seen in Table 5.1. To examine the effect of changing the physical size of patches in the frequency domain, another 5 settings were created. As these parameter settings for the frequency domain were independent from those for the time domain (hence any setting from the former group could be paired with one setting from the latter group), 15 settings were examined altogether. The parameter values for the frequency domain can be seen in Table 5.2.

At this point we should mention the parameters of the neural nets used in these experiments. Some neural net parameter values were dependent on the parameter values shown in tables 5.1 and 5.2, while other parameter values depended on the task. The following were however true for all experiments: each filtering sub-layer had 9 neurons ($\mathbf{L}^{\#_{\mathbf{n}}} = 9$), whose output was connected to a convolutional bottleneck layer, consisting of 200 neurons ($\mathbf{B}^{\#_{\mathbf{n}}} = 200$)

Table 5.2: Parameter settings used in the frequency domain.

| parameters | F1 | F2 | F3 | F4 | F5 |
|---|---|---|---|---|---|
| $\mathbf{P}_{\mathbf{f}}^{\#}$ | 3 | 5 | 7 | 9 | 11 |
| $\mathbf{S}_{\mathbf{F}}^{\#}$ | 18 | 26 | 34 | 42 | 50 |
| $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{P}}}$[2] | ~1420 mel | ~980 mel | ~750 mel | ~610 mel | ~510 mel |

[2]Calculations here are predicated on the lower cutoff of the first filter in the filter-bank being at 0Hz, and the higher cutoff of the last filter being at the Nyquist frequency (which in the case of TIMIT, a database recorded with a sampling rate of 16kHz, is 8000Hz), and also on the assumption that the conversion between the Hertz scale and Mel scale is computed by the following formula: $2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)$, where $f$ is the Hz value we wish to convert. Given these assumptions, we can see that if 18 mel-channels cover a range of 8000 Hz (or 2840 mel), 1 mel-channel covers a range of approximately 160 mel, and 9 mel-channels (the technical size of the patch) covers a range of approximately 1420 mel. In every other case $\mathbf{P}_{\mathbf{f}}^{\delta_{\mathrm{P}}}$ is calculated the same as described here.

Table 5.3: Phone error rates (PER) on the validation set using monostate and tristate phone models. Reported error rates are the average of 10 independently trained neural nets. (The best error rate and those not significantly different from it are highlighted in the monostate case and the tristate case as well).

| | 61 monostate models | | | 61 tristate models | | |
|---|---|---|---|---|---|---|
| | T1 | T2 | T3 | T1 | T2 | T3 |
| F1 | 21.36% | 20.78% | 20.68% | 21.29% | 20.39% | 20.04% |
| F2 | *20.65%* | 20.20% | 20.12% | *20.26%* | 19.64% | 19.28% |
| F3 | 20.18% | 19.81% | 20.00% | 20.05% | 19.32% | 19.03% |
| F4 | 19.89% | **19.65%** | 19.79% | 19.61% | 19.15% | **18.88%** |
| F5 | 19.85% | **19.52%** | 19.84% | 19.64% | 19.08% | **18.86%** |

### Results

We examined the phone recognition results attained by using these parameters on the randomly selected 10% of the training set of TIMIT as a validation set. Experiments were carried out with 61 (monostate) and 183 (tristate) phone models.

Results of these experiments are summarized in Table 5.3. In earlier experiments we utilised the settings corresponding to the F2/T1 parameter pair (results obtained with these settings are shown in Italics in the table). As can be seen in Table 5.3, the best results in the monostate and tristate cases are got with different settings. The minimal error rate that can be got in the two cases is also quite different. The lowest error rates in the experiments were obtained using tristate models with the F4/T3 and F5/T3 settings. This means that the best results were attained when the frame shift of the primary feature extraction phase was decreased from 10 ms to 6 millisecond, while the number of mel channels used was increased from 26 to 42 or 50. However, while the increase in mel-channels from 34 to 42 resulted in significantly lower error rates, the same was not true for the increase from 42 mel-channels to 50. Hence, we decided not to further increase the number of mel-channels used, and fixed the number of mel-channels to be used during processing to 42. This means that when examining whether the suggested parameter changes were indeed beneficial, we will compare those attained using the original F2/T1 parameters with the results attained using the new F4/T3 parameters.

### 5.1.4 Delta and Acceleration coefficients

In our earlier publications on the topic of joint optimisation of spectro-temporal features and acoustic models, the idea of incorporating delta ($\Delta$) and acceleration ($\Delta\Delta$) coefficients was raised multiple times [125, 126]. This idea was based on the results of earlier experiments where the addition of $\Delta$ and $\Delta\Delta$ coefficients to spectro-temporal features increased the accuracy of the recognition process [123]. Here, we examined two techniques for implementing this modification. Delta and acceleration coefficients could be added by incorporating dedicated neurons to the framework between the feature extraction layer and the following layer, which emulate the computation of these coefficients with the proper setting of their weights and the size of the context used in their input. There is however no obvious solution for this, as it would necessitate finding a way for the backpropagation to bypass these neurons. Because of this we opted for a simpler solution to incorporate Delta-like coefficients to the joint framework. In the HTK toolkit $\Delta$ coefficients are calculated by the following formula:

$$d_T = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{T+\theta} - c_{T-\theta})}{2 \cdot \sum_{\theta=1}^{\Theta} \theta^2}, \tag{5.7}$$

where $d_T$ is the $\Delta$ coefficient in time $T$, calculated using the coefficients between $c_{T-\theta}$ and $c_{T+\theta}$ [236]. If we apply this formula to the features obtained by applying filter $F$ on patch $P$,

$$c_T = \sum_{t=1}^{N} \sum_{f=1}^{M} P_T(f,t) \cdot F(f,t), \tag{5.8}$$

where patch $P$ is extracted from the spectral representation $S$ like so:

$$P_T(f,t) = S(f, T+t), \tag{5.9}$$

then (5.7) can be reformulated in the following manner:

$$d_T = \frac{\sum_{\theta=1}^{\Theta} \theta \Bigg( \sum_{t=1}^{N} \sum_{f=1}^{M} F(f,t) \cdot \Big( S(f, T+\theta+t) - S(f, T-\theta+t) \Big) \Bigg)}{2 \cdot \sum_{\theta=1}^{\Theta} \theta^2}, \tag{5.10}$$

As the value of the denominator depends only on the $\Theta$ parameter, for a given $\Theta$ the denominator has a constant value. This means that we can replace the denominator by a constant and make the process of deduction easier to follow. This constant will be called $\vartheta$, and its value will be given by the formula

$$\vartheta = 2 \cdot \sum_{\theta=1}^{\Theta} \theta^2. \tag{5.11}$$

If we now reorder (5.10) and use the constant introduced in (5.11), we get the following formula for the calculation of the $\Delta$ coefficient from a feature obtained as the output of a filter applied on spectral representation $S$:

$$d_T = \sum_{t=1}^{N} \sum_{f=1}^{M} F(f,t) \cdot \frac{\sum_{\theta=1}^{\Theta} \theta \Big( S(f, T+\theta+t) - S(f, T-\theta+t) \Big)}{\vartheta}, \tag{5.12}$$

However, if we first apply (5.7) on the spectral representation $S$, we get a $\Delta$ version of the spectral representation, where the element corresponding to frequency $f$ and time $t$ is given by the following formula:

$$S_{\Delta}(f,t) = \frac{\sum_{\theta=1}^{\Theta} \theta \Big( S(f, t+\theta) - S(f, t-\theta) \Big)}{\vartheta}. \tag{5.13}$$

In this case applying Filter $F$ on Patch $P$ extracted from the spectral representation $S$ at time $T$ would mean computing the following formula:

$$c_T = \sum_{f=1}^{N} \sum_{t=1}^{M} F(f,t) \cdot S_{\Delta}(f, T+t). \tag{5.14}$$

After resolving the second member of the product in (5.14) based on (5.13) we see that (5.12) is equal to (5.14), meaning that $c_T = d_T$. Hence, if our goal is to calculate the $\Delta$ coefficient of a feature extracted from the spectral representation, we can also achieve this by first calculating the $\Delta$ version of the spectral representation ($S_{\Delta}$), and then applying the process of feature extraction on $S_{\Delta}$. Similar reasoning can be applied to the calculation of $\Delta\Delta$ coefficients.

The above results tell us that it is possible to use the $\Delta$ and $\Delta\Delta$ coefficients of the features extracted in the joint framework. For this to work, however, it must be ensured that the same Filter $F$ is applied on $S$, $S_{\Delta}$ and $S_{\Delta\Delta}$. With the proper initialisation of weights, this could be resolved in the current framework before the training of the network containing the layer implementing feature extraction. However, currently there is no mechanism that would guarantee that the weights in the filtering sub-layers working on the spectral representation as input, and the filtering sub-layers working on the derivatives of the spectral representation will be the same at the end of the training process. This is why the term “Delta-like coefficients” was used instead of Delta coefficients. It should be noted here that this is not necessarily a disadvantage, as the increased degree of freedom in the extraction of features from the derivatives of the spectral representation can also be beneficial. The examination of this possibility, however, is beyond the scope of this study. Here, we will only examine the effect of the addition of these so-called “Delta-like” coefficients on the speech recognition performance of the joint framework.

Table 5.4: Phone error rates (PER) got on the core test set of TIMIT (Reported error rates are the average of 10 independently trained neural nets). Here, the best score (and those not significantly different from it) is highlighted in bold.

| Initialisation | Parameters | | $\Delta$ | PER |
|---|---|---|---|---|
| | Frequency | Time | $\Delta\Delta$ | |
| Random | F2 | T1 | | 24.4% |
| Gábor | F2 | T1 | | 24.2% |
| Random | F4 | T3 | | 18.8% |
| Random | F4 | T3 | ✓ | **18.5%** |

## 5.1.5 Experiments

Unlike that in the experiments conducted to determine the optimal parameter settings, not only was the relative performance attainable in different parameter settings important, but also the absolute performance. For this, in the following experiments we used bigger neural nets. Preceeding the bottleneck layer three further hidden layers (each consisting of 1000 rectifier units) were inserted, and the resulting network was trained in two steps (similar to the 2-step training applied in [222]). In the first step the network was trained without convolution in such a way that the output layer was placed directly after the bottleneck layer. In the second step this output layer was removed, and two additional hidden layers (each consisting of 1000 rectifier units) were added alongside with a new output layer before the neural net was further trained, this time with the use of convolution. Another difference was that when the framework was trained based on the F2/T1 parameter settings, the bottleneck layer previously consisting of 200 neurons had 220 neurons, while the number of neurons in the hidden layers previously containing 1000 neurons was increased to 1100 neurons. The reason for these changes was to ensure that the frameworks based on different parameter settings had a similar number of trainable parameters.

### Experiments on clean speech using the TIMIT corpus

To facilitate a comparison of our results with those of other researchers working on the TIMIT corpus, following the procedure described in [222], training was carried out using 858 context-dependent states. As outlined in Section 5.1.3, before the evaluation phase these 858 labels were reduced to 39.

The results of these experiments are summarized in Table 5.4. First, let us examine the top two rows of the table. Here, we can see the results obtained when the weights in the filtering layer were initialised randomly and based on

Table 5.5: Phone error rates (PER) presented in the literature, on the core test set of TIMIT. The best score is highlighted in bold. The score got by us is highlighted in Italics.

| Method | PER |
|---|---|
| Baby et al. [13] | 19.6% |
| Plahl et al. [182] | 19.1% |
| Tóth [222] | 18.7% |
| Current work | *18.5%* |
| Graves et al. [68] | 17.7% |
| Tóth [224] | **16.7%** |

the Gabor filter set introduced in Chapter 2. Although initialisation based on Gabor filters leads to a significant improvement ($p < 0.05$), the relative error rate reduction is below 1%. Because of the low rate of improvement, in later experiments conducted in this study,, only random initialisation was used, for the sake of simplicity.

Looking at the lower lines of Table 5.4 we can examine the effect the proposed modifications had on the results when the filtering layer was initialised randomly. We can see that both proposed modifications (the fine tuning of parameters, and the addition of Delta-like coefficients) resulted in a significant decrease in PER scores. The combined effect of these modifications lead to a relative error rate reduction of about 24% overall.

By comparing the results presented here with various results found in the speech recognition literature (see Table 5.5), we see that with the proposed modifications the joint framework attains competitive recognition scores. Although the recognition scores are worse than those reported by Graves et al. [68], the authors in the given study used recurrent networks, which means that a direct comparison of results may not neccessarily be apt. The phone recognition error rates presented here are also significantly higher than those reported by Tóth [224], but a direct comparison of the two framework is not necessarily justifiable, as in the case of Tóth's neural network convolution was applied *both* in the time and frequency domains, and the dropout method was also applied. A more apt comparison would be the network applied in an earlier study by the same author [222], the results achieved by which are slightly worse than those presented here, even though the size of our network was only a quarter of the size of the network used in the other study.

Table 5.6: Average word error rates (WER) on the Aurora-4 speech corpus (the results reported are the average of 5 independently trained neural networks). Here, the best score (and those significantly not different from it) is highlighted in bold.

| Initialisation | Parameters | | $\Delta$ | WER |
|---|---|---|---|---|
| | Frequency | Time | $\Delta\Delta$ | |
| Random | F2 | T1 | | 12.4% |
| Random | F4 | T3 | | 11.9% |
| Random | F4 | T3 | ✓ | **11.6%** |

### Experiments on noise contaminated speech using the Aurora-4 database

To test the performance of the neural net framework in the presence of various additive noise types, as well as under mismatched transfer characteristics, the experiments were repeated on the Aurora-4 continuous speech recognition task. In this case the multi-conditional set was used for the training of the models, then the resulting models were evaluated on the standard test sets for the task (the same test sets that were described in Section 4.3.3). Similar to what we saw in Section 4.3.3, for the sake of simplicity and to save time, we only experimented with random initialisation of filter weights. The same will be true for the rest of the experiments in this chapter. The reasoning behind this is twofold. First, a comparison of results attainable using random filter initialisation and using Gabor filter sets was already described in Chapter 2. Furthermore, it could be argued that with the modified time- and frequency parameters in creating the spectral representation, a new set of initial Gabor filters should be selected for optimal performance. While this would be an interesting task, the focus here is the utility of the joint framework, and if it can be demonstrated even with the random initialisation of filters, the task of further improving these results by finding a more suitable set of initial filters may as well be left for future study.

The results of the experiments on the modified framework using the Aurora-4 database are summarized in Table 5.6. We notice that with each subsequent modification introduced, the error rates achieved by the framework are lower and lower. First, the change in the parameters used for creating the spectral representation leads to a relative error rate reduction of about 5%, then the inclusion of Delta-like features leads to a further relative error rate reduction of 2%. Altogether the two modifications lead to a combined relative error rate reduction of about 7%. What is more is that with both proposed modifications, the improvement in recognition scores is significant (at $p < 0.00005$).

Table 5.7: Word error rates (WER) got on the Aurora-4 test set in the multi-condition training scenario. The best score is highlighted in bold, while our result is highlighted in Italics.

| Method | WER |
|---|---|
| Chang and Morgan [32] | 16.6% |
| Seltzer et al. [206] | 12.4% |
| Martinez et al. [145] | 12.3% |
| Baby et al. [12] | 11.9% |
| Current work | *11.6%* |
| Narayanan and Wang [171] | 11.1% |
| Rennie et al. [192] | **10.3%** |

Let us again compare these results with those published in recent speech recognition literature. Even without the modifications introduced so far by us, the word error rates attained by the proposed framework are much lower than those reported by Chang and Morgan [32] in their 2014 study with CNNs, although they used a noise robust representation (Power Normalized Spectrogram) as well as Gábor filters. Due to a change in parameters for time and frequency in the spectral representation, the proposed framework achieves lower error rates than those reported by Seltzer et al., who also applied Noise Aware Training and Dropout [206]; as well as those reported by Martinez et al., who applied their DNNs on AMFB (Amplitude Modulation Filter-Bank) features [145]. In the latter comparison, it is also noteworthy that while Martinez et al. reported a training time of 4 days for their DNNs on an NVIDIA Tesla K20C GPU, our neural net training times were below 12 hours on a GPU of equal performance (GeForce GTX 770) [1]. Lastly, the inclusion of Delta-like coefficients meant that the proposed framework could also attain lower WERs than those reported by Baby et al. using a sophisticated speech enhancement technique [12].

The results reported here however were somewhat worse than those reported by Narayanan and Wang, who applied a noise adaptive training method, and also combined multiple systems by averaging their posterior (noting that their best performing single system got a very similar WER score than that reported here, namely 11.5%) [171]. The error rate reported here is also higher than those reported by Rennie et al. [192]. However, they use several advanced techniques in their network that so far have not been introduced in ours. These are maxout DNNs and dropout (specifically, annealed dropout).

## Experiments with maxout

These latest experiments on the Aurora-4 database were repeated using maxout deep convolutional neural nets. Similar to the rectifier function, maxout is a novel activation function applied in hidden neurons, illustrated in Figure 5.2. The same figure also shows the structure of the neural network. Owing to the maxout decreasing the number of connection between layers, and the omission of one hidden layer below the bottleneck, we were able to increase the number of neurons in the various layers and still decrease the number of trainable parameters in the network. At the same time, the performance of the neural net on the Aurora-4 corpus using the multi-conditional training scenario remained practically the same (11.6%) despite the slightly smaller size.

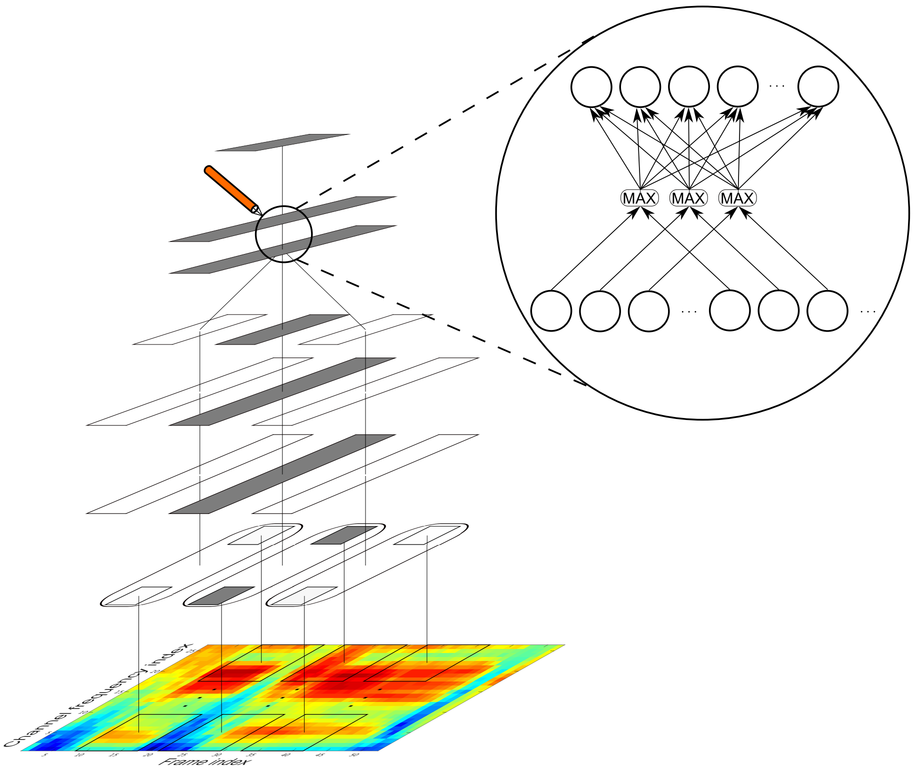

Figure 5.2: Structure of the Deep Convolutional Neural Net used, with the maxout magnified for the sake of clarity.

## 5.2 Band Dropout

The dropout method was shown to make deep neural networks generalise better by randomly omitting neurons during training [85], and now it is widely applied in automatic speech recognition [42, 155, 222]. The original study by Hinton et al. reported additional improvements when applying dropout to the input features as well [85]. However, in the framework of ASR, Deng et al. found that “applying dropout to input filterbank features has not been effective” [44]. Meanwhile, Miao and Metze reported that “an input dropout factor greater than 0 definitely degrades the recognition results” [155]. Hence the usefulness of input dropout in ASR is questionable, and requires further investigation.

Here, apart from examining input dropout, we also propose a modified version. To understand the motivation behind our modification, let us first look at the original approach. In the original input dropout scheme each feature is discarded or retained independently, meaning that random “pixels” are being dropped from the time-frequency map (see Figure 5.3 (A)). Behind this there is the implicit assumption that the order of the features does not carry any information. But the input, which here is a spectrogram-like time-frequency representation, has a well-defined structure, and the dropout scheme could exploit this fact. Furthermore, it is known that humans can recognise bandpass-filtered speech surprisingly well [7]. Therefore, here we propose a version of input dropout that discards whole frequency bands (see Figure 5.3 (B)). We hypothesise that this band dropout strategy will force the network to rely less on the whole spectrum, making it more robust to channel mismatches. Furthermore, as real-life background noise is more likely to affect the spectrogram in patches, rather than spreading all over the time-frequency plane in a uniform manner, we also expect this method to be more robust to background noise.

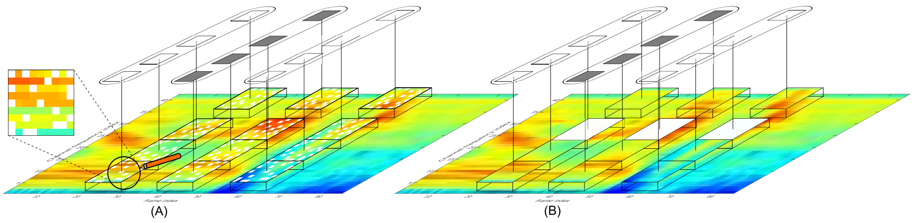

Figure 5.3: Illustration of the CNN structure applied here, with
(A) input dropout, discarding randomly selected features (marked by blank dots in the input of feature-extraction sub-layers), and
(B) band dropout, discarding the same amount of features, but with an entirely different distribution.

The frequency bands being dropped here consist of several neighbouring mel-filter channels arranged in such a way that these channels coincide with those used in the input of filtering sub-layers in the convolutional neural network structure. This means that dropping one frequency band can be implemeneted by setting the input of the corresponding convolutional filters to zero, while leaving the operation of the other filtering sub-layers untouched. Consequently, the layer that merges the output of the filtering sub-layers is forced to learn that it should not rely on all of the channels. Figures 5.3 (A) and 5.3 (B) show the difference between standard input dropout and band dropout (for the case of 4 processing bands). Clearly, with the proper selection of the dropout rate parameters, the two methods discard the same amount of data points, but with a quite different distribution. Another difference between the two approaches is that while the original method generates a new dropout mask for each data instance, here we use the same dropout mask within a given mini-batch. The advantage of this method is that it allows a faster form of matrix multiplication (Graham et al., 2015). Moreover, it can be easily and efficiently combined with a CNN, because we can implement the dropout of a convolutional channel for the whole mini-batch by simply skipping the evaluation of the given filters and replacing their output by zeros. Although Graham et al. thoroughly evaluated their batch-wise dropout method and concluded that it gives basically equivalent results with the standard frame-wise solution, we will also examine the batch-wise version of the original input dropout method (where for each mini-batch the same "pixels" will be dropped), and compare its results to those got with the input dropout where the dropped "pixels" vary within the mini-batch.

A comparison of batch-wise input dropout with input dropout's original implementation, as well as a comparison of traditional input dropout (both with and without batch-wise dropout) with band dropout will be performed using the continuous speech recognition task on the Aurora-4 corpus. As outlined in Section 1.2.2, this corpus was specially designed to evaluate the noise-robustness of speech recognisers. Although we expect dropout to be more beneficial in mismatched train and testing scenarios (i.e. when we have no samples of the noise present in the test set during training), both the clean and the multi-conditional training scenarios were used. Before discussing the experiments and their results, however, we should outline the roots of the band dropout method in earlier speech recognition research.

### 5.2.1 Relation to Prior Work

Although speech recognition using Convolutional Neural Networks and the band dropout method are relatively new ideas, our proposal of dropping spectral bands during training is clearly related to some well-established technologies.

For one, this method is similar to the so-called multi-band scheme introduced over two decades ago [47]. Multi-band processing (described earlier in Chapter 4) was also based on the concept of training acoustic models on partial spectral information [19]. The training of deep structures, however, was an open question, so separate classifiers were trained on the spectral bands (or subsets of bands). Then the band-based estimates produced by these classifiers were combined using a sophisticated decision logic which, in the simplest case, consisted of a 'merger' neural network [19]. In a way, band dropout can be viewed as a modern version of band-based training adjusted to deep convolutional networks.

As band-based training is a special case of the multi-stream approach, the band dropout method is connected to the latter as well. Our solution proposed here is clearly related to multi-stream combination method introduced by Mallidi et al. [141], which drops streams during the training of the merger network. Despite the similarities, however, there are notable differences between the two methods, both in the motivation and the implementational details. As is usual with the multi-stream scheme, Mallidi et al. train a large set of DNNs instead of just one deep structure, and apply a performance monitor at test time to decide which streams should be kept. The unselected streams are replaced by zeros, and their motivation for training the merger network with dropped streams is to prepare the merger net for the possibility of channels being zeroed out by the stream selection process. Here, we train just one deep convolutional network without any explicit channel selection process, and our motivation for dropout is the same as that for standard input dropout [85]. Hence, while the old multi-band systems were based only on a crude perceptual motivation, our approach is also supported by the machine learning theory of dropout. Furthermore, while the solution of Mallidi et al. is more complicated to implement, and also slower at run time (as the performance monitor needs to be invoked), the results we present here are significantly better than those cited in their study [141].

Another related technology is that of data-augmentation, which is an approach for increasing the robustness of our models by artificially generating additional training vectors from the existing ones [41, 121]. Recently, in a study by Bouthillier et al. it was pointed out that dropout can indeed be interpreted as a kind of data augmentation [23]. Hence, one may also regard band dropout as a special case of data augmentation where, by randomly deleting spectral bands, we artificially extend our set of training data with additional distorted versions of the training vectors on the fly. According to this interpretation, while Ko et al. modified the speech signals in the time domain [121], band dropout belongs to the family of data augmentation techniques that manipulate the data in the spectral domain [41].

### 5.2.2 Experiments

Convolutional neural networks seek to exploit the local spectro-temporal correlations of the input, and thus require a spectrogram-like input representation. Hence, these types of networks are usually trained on the log-energy outputs of a mel-scaled filterbank [4, 198, 225]. Because of this, in our first set of experiments we evaluated band dropout using the standard mel filterbank features. We worked with a mel-filterbank of 42 channels, and the energy values were extended with the corresponding $\Delta$ and $\Delta\Delta$ features.

However, this simple representation still has plenty of room for feature engineering (some examples of this feature engineering are given in Section 1.3.1). For example, Chang and Morgan experimented with power-normalised spectrum (PNS) features in combination with CNNs [32]. Recently, Ganapathy proposed an ARMA spectrogram modelling technique that outperformed the PNS features when evaluated with a DNN acoustic model on the Aurora-4 task [57]. This ARMA spectrogram representation preserves the local topology of the spectro-temporal feature space, and hence it is also a suitable input for CNNs. Because of this, in our second set of experiments, we evaluated the ARMA spectrogram features in combination with CNN acoustic models and band dropout. Here, the 39 ARMA features were extended by adding 39 derivative-like features obtained as the difference between neighbouring bands as folllows:

$$band_{\Delta}(K) = band(K+1) - band(K-1). \tag{5.15}$$

The proposed band dropout method was parametrised by using two variables. With parameter $P$ we can tune the probability of the band dropout being applied to the current batch of data. The other parameter $N$ sets the maximum number of bands that can be discarded via band dropout (as here the spectral representation was processed in 9 – overlapping – frequency bands, the parameter values available for parameter $N$ lay between 1 and 9 inclusive). If the current batch were to be selected for dropout (based on parameter $P$), then a random number $q$ is generated in the $[1, N]$ range. Then the input data was discarded in $q$ randomly chosen convolutional bands. Even though dropout slows down the convergence of the training process, so one can perform more sweeps through the training data when using dropout [42], here we did not modify either the training epochs or the stopping criterion. This way, our dropout results might be slightly suboptimal, but the training times stayed roughly the same as those without dropout.

Table 5.8: The effect of band dropout on the frame error rates for the train and development sets, and on the word error rates for the test set (using the mel-spectral features and the multi-conditional training set). The baseline (no dropout) score is given in the $P = 0$ column.

| | Frame error rate (%) – Train | | | | | Frame error rate (%) – Dev | | | | | Word error rate (%) – Test | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N \ P | *0* | *0.2* | *0.4* | *0.6* | *0.8* | *0* | *0.2* | *0.4* | *0.6* | *0.8* | *0* | *0.2* | *0.4* | *0.6* | *0.8* |
| *5* | | 32.9 | 34.0 | 34.9 | 37.4 | | 36.9 | 36.7 | 36.8 | 37.7 | | 11.3 | 11.2 | 10.9 | 11.3 |
| *6* | 30.2 | 33.1 | 35.2 | 36.3 | 39.1 | 36.8 | 36.8 | 36.9 | 36.9 | 37.9 | 11.7 | 11.2 | 11.1 | **10.8** | 11.2 |
| *7* | | 34.2 | 37.1 | 38.3 | 41.0 | | 37.0 | 37.3 | 37.1 | 37.9 | | 11.2 | 11.2 | 10.9 | 11.1 |

## Results with mel-spectral features

We first evaluated the usefulness of band dropout using mel-spectral features in the multi-conditional training scenario. As our training process optimised the frame-level cross-entropy, let us first examine the frame-level error rates achieved with different parameter settings (see Table 5.8). As one would expect, band dropout makes the learning process more difficult, so when we increase either $P$ or $N$, the frame-level error rate quickly increases on the train set. This is not the case, however, on the development set. As narrowing the gap between the training and development error rates decreases the chance of overfitting the training data, a reasonable heuristic for meta-parameter tuning is to chose the largest $P$ and $N$ values for which the error on the development set is not significantly larger than that for the baseline model. Applying this strategy here suggests that we should chose values of 0.6 and 6 for P and N, respectively.

Let us now examine the WER scores on the test set (the rightmost part of Table 5.8). These are quite consistent with the frame-level errors of the development set, and are actually lower than that for the baseline with all parameter values. And while the WER is the lowest (10.8%) for the parameter values suggested by our heuristic ($P = 0.6$, $N = 6$), slightly different $P$ and $N$ values result in similarly low scores. The relative error rate reduction compared to not using band dropout is 7.3%, which was found to be statistically significant in a paired t-test ($p = 0.00008$). The score of 10.8% also compares favourably with results reported by other authors. For instance, the Kaldi recipe gives 13.6% with a DNN, while Huang et al. got 13.4% using FBANK features and a CNN [90]. While Rennie et al. attained a lower error rate (10.3%) with a similar setup when using annealed dropout [192], with a fixed dropout rate their WER score was no lower than the 10.8% reported here. Other authors have also reported better results (e.g. Qian et al. [186]), but these methods rely on explicit noise and/or speaker adaptation, which is not required in our approach. Compared to these methods, the implementation of band dropout requires only minor modifications, and has a negligible computational cost.

Table 5.9: Word error rates (WER) obtained with various versions of input dropout on the Aurora-4 task, using the mel-spectral features.

| Method | Implementation | Parameter values | Training scenario | |
|---|---|---|---|---|
| | | | Multi-conditional | Clean |
| band dropout | batch-wise | $P = 0.6, N = 6$ | 10.8% | 26.8% |
| standard input dropout | frame-wise | $P = 0.1$ | 11.3% | 31.4% |
| standard input dropout | frame-wise | $P = 0.2$ | 11.0% | 31.4% |
| standard input dropout | frame-wise | $P = 0.3$ | 11.4% | 33.3% |
| standard input dropout | batch-wise | $P = 0.2$ | 10.8% | 30.5% |
| no dropout | – | – | 11.6% | 33.7% |

The second column of Table 5.9 compares the performance of channel dropout with two versions of standard input dropout. First, we looked for the optimal dropout probability value for standard (frame-wise) input dropout. The parameter values $N = 6$ and $P = 0.6$ mean that on average 3.5 channels (out of 9) are dropped with a probability of 0.6, which corresponds to a $(3.5/9) \cdot 0.6 = 0.23$ equivalent input dropout percentage. For the sake of comparison, we evaluated conventional input dropout with dropout rates $P = 0.1, 0.2, 0.3$. The best result was obtained with $P = 0.2$, when the word error rate was 11.0%. Next, we repeated the evaluation with $P = 0.2$ using the batch-wise version of dropout. In this case, we got a slightly better score of 10.8%. While the scores obtained using dropout (10.8%, 11.0% and 10.8%) proved to be significantly better than the baseline score obtained with no dropout ($p < 0.0007$), these scores were not significantly different from each other. Hence, in this experiment the channel dropout method turned out to be useful, but no better than standard dropout.

Next, we examined how band dropout behaves in the clean training scneario. To test the robustness of our parameter selection strategy, we did not repeat the process of tuning, but worked with those parameters values found optimal previously ($P = 0.6$, $N = 6$). The last column of Table 5.9 shows the baseline word error rate obtained without dropout, and the score got with the proposed band dropout method (26.8%). For the sake of comparison, we evaluated conventional input dropout with dropout rates of $P = 0.1, 0.2, 0.3$ (see the rows in Table 5.9's rightmost column). The optimal performance was achieved at 0.2, with a word error rate of 31.4%. This shows that while input dropout reduced the error rate by 6.8%, band dropout decreased it by a much larger amount, namely 20.4%. In this case, band dropout was significantly better compared both to the baseline and to the standard input dropout ($p < 0.00001$). This accords with our expectations that band dropout is the most beneficial when we have no noisy training samples, so the network cannot adapt to the noise using just the training data. Lastly, we repeated the evaluation of standard dropout with $P = 0.2$ in a batch-wise manner. The 30.5% we obtained is lower than the 31.4% of frame-wise dropout, but much higher than the 26.8% of band dropout.

Table 5.10: Word error rates (WER) obtained with various versions of input dropout on the Aurora-4 task, using the ARMA features.

| Method | Implementation | Parameter values | Training scenario | |
|---|---|---|---|---|
| | | | Multi-conditional | Clean |
| band dropout | batch-wise | $P = 0.6, N = 6$ | 10.8% | 16.0% |
| band dropout | batch-wise | $P = 0.4, N = 5$ | 10.5% | 16.5% |
| standard input dropout | frame-wise | $P = 0.1$ | 11.1% | 19.2% |
| standard input dropout | frame-wise | $P = 0.2$ | 11.4% | 19.9% |
| standard input dropout | batch-wise | $P = 0.1$ | 11.1% | 19.0% |
| no dropout | – | – | 10.7% | 19.1% |

We first evaluated the ARMA features under the multi-conditional training scenario. As can be seen in Table 5.10, compared to the mel-spectral baseline score of 11.6%, the introduction of the ARMA features reduced the error by 7.8%, even without using band dropout. However, turning on band dropout with the parameters of $P = 0.6$ and $N = 6$ did not reduce the error rates any further. Then, we looked for the parameter values that would give the best result on the test set. The 10.5% we got with $P = 0.4$ and $N = 5$ is just slightly, but not significantly better than the baseline 10.7%. We also evaluated the standard input dropout with $P = 0.1$ and 0.2, as well as the batch-wise input dropout with $P = 0.1$, and we got worse results than the baseline. Overall, in contrast with the mel-spectral feature set, in the multi-conditional training scenario neither dropout method helped. We assume that the ARMA technique can already efficiently handle the noise when we have training samples from all the noisy conditions, but testing this would require a deeper analysis.

In the final set of experiments we used the ARMA features and only clean training data. As can be seen in Table 5.10, the change of features by itself led to an impressive error rate reduction of 43.3% (from 33.7% to 19.1%). Similar improvements were reported in Ganapthy's original study of ARMA features [57]. By using the band dropout with $P = 0.6$ and $N = 6$, we got an additional error rate reduction of 16.2% (with the WER being 16.0%). This means that while band dropout did not help the ARMA features under the multi-conditional training scenario, here it significantly improved the recognition ($p < 0.0001$). Again, we attempted to vary $P$ (between $0.4 - 0.8$) and $N$ (between $4 - 6$), but even the worst result obtained was 16.5%, proving that band dropout is quite stable across a wide range of parameter values. Lastly, we evaluated standard input dropout, and similar to the multi-conditional case, we got worse results than the baseline. In summary, while band dropout performed no better than standard input dropout in the multi-conditional training scenario, it significantly outperformed the standard method when only clean training data was used. This justifies our expectations that band dropout increases the generalisation capability of the model under mismatched training and test conditions.

Table 5.11: The performance of the CNN when operating on the mel-spectrum and the ARMA features without dropout and with band dropout, using only the clean training set.

| Data set | | FBANK | | | ARMA | | |
|---|---|---|---|---|---|---|---|
| | | no dropout | band dropout | improvement | no dropout | band dropout | improvement |
| Set A | Clean | 3.8% | 3.1% | 17.8% | 3.7% | 3.8% | -4.2% |
| Set B | *Car* | 13.0% | 7.2% | 44.2% | 6.1% | 5.5% | 9.5% |
| | *Babble* | 20.1% | 15.7% | 21.7% | 13.9% | 11.0% | 20.4% |
| | *Restaurant* | 29.3% | 24.2% | 17.2% | 18.1% | 16.9% | 6.6% |
| | *Street* | 30.7% | 21.0% | 31.4% | 13.9% | 12.1% | 13.3% |
| | *Airport* | 20.7% | 15.2% | 26.5% | 13.3% | 12.6% | 5.2% |
| | *Train* | 28.2% | 20.2% | 28.4% | 15.1% | 12.4% | 17.7% |
| | Average | 23.7% | 17.3% | 27.0% | 13.4% | 11.8% | 12.2% |
| Set C | Clean | 35.7% | 29.5% | 17.3% | 13.6% | 9.4% | 30.9% |
| Set D | *Car* | 43.1% | 35.7% | 17.2% | 21.1% | 15.3% | 27.5% |
| | *Babble* | 46.3% | 39.0% | 15.7% | 29.9% | 24.1% | 19.4% |
| | *Restaurant* | 48.9% | 40.2% | 17.7% | 31.0% | 27.6% | 11.2% |
| | *Street* | 55.0% | 44.5% | 19.1% | 28.8% | 24.6% | 14.6% |
| | *Airport* | 45.6% | 38.3% | 16.0% | 29.0% | 24.3% | 16.1% |
| | *Train* | 51.3% | 41.3% | 19.5% | 29.6% | 24.2% | 18.2% |
| | Average | 48.4% | 39.8% | 17.6% | 28.2% | 23.3% | 17.3% |
| Average | | 33.7% | 26.8% | 20.4% | 19.1% | 16.0% | 16.2% |

The word error rates of 26.8% and 16.0% reported in tables 5.9 and 5.10 respectively, are average scores over the four test sets. To get a deeper insight into the precise conditions where band dropout is the most beneficial, we performed a detailed analysis of how the error rate improvement varies under different testing conditions. Table 5.11 lists the word error rates obtained for each test set, and for each noise type (within sets B and D). The first thing we notice is that under ideal conditions when there is neither channel distortion nor additive noise present (i.e., set A), band dropout only improves the recognition accuracy in the case of mel-spectral features, while with ARMA features, it actually slightly degrades the performance. However, for test set B, when additive noise is present, band dropout yields a significant gain of 12.2% relative error rate reduction on ARMA features, and a 27.0% relative error rate reduction on mel-spectral features, which represents the biggest gain for the feature set in question. The biggest improvement on the ARMA features (30.9%) however is attained on set C, when the microphone transfer characteristics are different, but no noise is involved. The improvement got on the same set with mel-spectral features is 17.3%, which is almost identical to the improvements obtained on set D with either feature set. These scores accord with our hypothesis that band dropout will be beneficial under mismatched training and test conditions, especially for mismatched transfer channel characteristics, and in the presence of additive noise. The effect of band dropout in the latter case, however, is less beneficial when applied on ARMA features. Once again, this might be due to the noise robustness of ARMA features.

Table 5.12: Comparing the band dropout result on ARMA features with our baseline, our earlier results, and with results cited in the recent literature, when using the clean training scenario.

| Method | WER |
|---|---|
| CNN with ARMA features | **16.0%** |
| CNN with ARMA features, no dropout | 19.1% |
| Multi-band CNN with ARMA features | 17.8% |
| DNN with ARMA features plus DCT [57] | 18.5% |
| DNN with DNN speech enhancement of FBANK [102] | 17.5% |
| DNN with Spectral masking [136] | 22.8% |
| CNN with PNS features plus Gabor Filter Kernels [32] | 22.9% |
| DNN with Exemplar Based Enhancement [12] | 26.8% |
| CNN with FBANK features [90] | 28.9% |

Finally, in Table 5.12, we compare our best score with some recent results found in the literature. Our first base of comparison is our earlier paper where the ARMA features were evaluated using a fully connected DNN [57]. The results got with this model were slightly better than our baseline here, which was perhaps due to the DCT postprocessing of the feature trajectories before feeding them into the DNN. However, our CNN with channel dropout significantly outperformed this result ($p < 0.0006$). It is also interesting to compare our best WER scores attained here with those obtained using multi-band processing, especially given that the former method was partially inspired by the latter. We observe that while using the multi-band approach we were able to achieve better WER scores than that of our baseline and that reported by Ganapathy [57], the multi-band approach was significantly outperformed by the band dropout method ($p < 0.00005$).

The remaining rows of the table list other recent results, all obtained with a DNN or a CNN (as there is a common agreement now that these methods outperform conventional HMM/GMMs). While Huang et al. applied a simple CNN on the FBANK features, others applied either a sophisticated feature extraction method [12, 136] or a refined CNN architecture [32]. We see that the ARMA feature set outperforms both these approaches by a large margin. And when combined with channel dropout, it even outperforms the results of Jun et al. although in their study the front-end utilises samples got from the multi-condition training set. The reason for this is that the training phase of their speech enhancement DNN required pairwise clean-noisy data, which they obtained from the multi-condition training set [102].

## 5.3 Conclusions

*This chapter can be divided into two distinct parts. In the first part we revisited the joint framework for the training of neural nets and the optimisation of spectro-temporal features coefficients. We proposed two modifications in this framework, and investigated their effects on the TIMIT phone recognition and Aurora-4 large vocabulary speech recognition tasks. Using the results of these experiments we demonstrated that both modifications lead to improvements in the speech recognition performance. What is more, we demonstrated that with these improvements the joint framework is capable of producing results that are competitive with the state-of-the-art on both corpora.*

*In the second part we introduced band dropout as a novel input dropout method that besides being straightforward to implement and having only a negligible additional computational cost, is also beneficial in combination with CNN-based acoustic modelling. The effectiveness of this method was demonstrated using the Aurora-4 database with different input representations.*

*Utilising the multi-condition training set and filterbank features, channel dropout yielded a relative error rate reduction of* 7.3% *vs. no dropout. More dramatic improvements were obtained in the case of train-test mismatch conditions. When only using the clean training data, for the filterbank features the channel dropout method yielded a relative error rate reduction of* 20.4% *over no dropout, and* 12.3% *over batch-wise input dropout, while for ARMA features it yielded a relative error rate reduction of approximately* 16% *compared to the no dropout and standard input dropout cases. Overall, we can say that channel dropout beats standard input dropout by a large margin. Furthermore, with ARMA features our system produced an absolute error rate of* 16.0%*, which is among the best results published so far for this task.*

# Chapter 6

# Summary

In this study, we examined the task of noise robust speech recognition from the perspective of spectro-temporal processing. For this, in Chapter 1 we first introduced core concepts of speech recognition for the HMM/ANN framework, such as the preprocessing of the speech signal, spectral representation, feature extraction, the HMM/ANN model itself, along with language models and their connection with the acoustic model. Here, we also discussed key methods and tools (such as speech corpora) that were also neccessary for our experiments. In the subsequent chapters we examined various spectro-temporal processing techniques. Here, each chapter paved the way for the next chapter, and hence the techniques described in each chapter were built on concepts introduced in the preceding chapters.

## 6.1 Spectro-Temporal Feature Extraction

Here, we examined a fundamental application of spectro-temporal processing. In this approach, from the different stages of the HMM/ANN hybrid model (outlined in the Introduction), only the secondary feature extraction stage is modified. In this stage the features are extracted from patches that are localised both in time and frequency. These features, however, after being concatenated are used in later stages just like any other features would be. There are various methods for extracting spectro-temporal features, two of which are briefly discussed in Chapter 1. Here two additional methods were elaborated on, namely the 2D DCT approach and the procedure using Gabor filters.

In our experiments with 2D DCT coefficients we first attempted to find optimal parameter values for spectro-temporal feature extraction. After these initial experiments we compared our 2D DCT features with MFCCs based on experiments conducted on the TIMIT speech corpus. We demonstrated that using 2D DCT features we can get similar or even better results than those got using MFCCs. In the case of noise contaminated speech for example, 2D DCT features performed better in each case than their MFCC counterparts. Furthermore with a simple but elegant method for the combination of the two feature sets, we achieved lower phone recognition error rates on clean speech as well.

When working with Gabor filters, we first examined several methods for the selection of a suitable filter set. Then, using the results of our experiments got on an English database and a Hungarian speech database, we compared the performance of the filter sets we created as well as the performance of filter sets found in the speech recognition literature. Here, we found that the filter set we created based on simple heuristics gave the best performance score. Not only did it outperform other filter sets, it also outperformed the MFCCs on both the clean and the noise contaminated version of TIMIT, and gave a similar performance on our Hungarian speech database. Using these results we also discussed potential problems of the automatic selection of Gabor filters.

## 6.2 The Joint Training of Spectro-Temporal Features and Neural Nets

Here, motivated by the obstacles that arose in automatic feature selection, we introduced a new method for filter selection. The joint training framework examined in this chapter works by integrating the feature extraction step into the lowest layer of a neural net, which means that by training the parameters in this layer via backpropagation, we also train the filters these layers are implementing. Using this method, we effectively combined the stages of secondary feature extraction and neural net training. This combination consistently led to lower phone error rates in our experiments compared to the error rates attained when the two stages were carried out separately. Futhermore, we added various modifications to this joint network training model, based on the current advances in neural network research. By introducing these modifications we managed to further lower the resulting error rates, demonstrating the framework's capability for improvement. These experiments also demonstrated that training the initial filter coefficients is indeed beneficial, and in most cases leads to a better recognition performance. In spite of the benefit in training our filters, it was also shown to be beneficial to begin this process with a good initial filter set.

## 6.3 The Multi-Band Processing of Speech using Spectro-Temporal Features

Here, motivated by the compatibility of spectro-temporal processing with the multi-band approach, we combined the two in a set of experiments. First, we demonstrated on the TIMIT database that using 2D DCT coefficients and our handcrafted Gabor filter set in conjuction with the multi-band approach leads to lower error rates than using the same features without the multi-band approach. And what is more, contrary to some earlier findings, this was true for both clean speech and for speech contaminated with a wide variety of noise-types. Furthermore, with the introduction of Deep Neural Networks and the technique of convolution, these error rates decreased farther. In these experiments we also reexamined a technical detail – the mirroring of low frequency bins – in our process, and decided to omit it from later experiments.

In this chapter, we also successfully incorporated the multi-band approach into our joint training framework (introduced in the previous chapter). When evaluating the resulting method on the clean training scenario of the Aurora-4 speech recognition task, we found that besides significantly improving the results we got with the joint-framework, the addition of the multi-band approach also enabled us to attain error rates that – at the time of their publication – were among the best results published for the given task. The experiments conducted on the combination of the joint training framework with the multi-band approach, however, provided other lessons as well. They revealed the difficulties associated with finding a proper recombination method. Because of this, our goal in the next chapter was to find a better way of integrating joint training with multi-band processing.

## 6.4 Band dropout

Here, we first reexamined the parameters of the joint training framework. We did so in order to demonstrate that with the proper settings of its parameter values, the framework can not only achieve significant relative error rate reductions, but it can also provide competitive results to those published in the speech recognition literature. For this, we introduced two enhancements into our framework, and examined their effect in experiments carried out on the TIMIT speech corpus and the Aurora-4 database. The results of these experiments demonstrated that these modifications lead to both relative and absolute improvements.

Lastly, our joint training technique was supplemented with a method that was inspired by input dropout and multi-band processing. Here, the input dropout was applied in such a way that in complete batches whole frequency bands were ignored. With this method we strive to improve the robustness of the trained model in the same way we did in the multi-band case, by forcing the network to rely less on the whole spectrum. We evaluated this method on the Aurora-4 database, using both the clean and the noisy training scenarios, with mel-spectral features as well as ARMA features. The results indicate that in the multi-condition training scenario the band dropout method either does not decrease the error rates (when used with ARMA features), or it does decrease error rates, but its results are not significantly better than those obtained with the standard input dropout (mel-spectral features). In the clean training scenario, however, band dropout significantly improves our results in both cases compared to those got using no dropout or standard input dropout. And what is more, when used in conjuction with ARMA features, the band dropout method achieves significantly better results than those presented in the previous chapter, leading to a performance score that is among the best reported for the given task.

## 6.5 Conclusions and Future work

In this study, we examined various spectro-temporal techniques used for noise robust speech recognition. With each phase, experiments were carried out with more complex (but also more successful) methods. On the TIMIT corpus for example, our initial error rate of 29% fell by more than 10%, to a final error rate of 18.5%, constituting a relative error rate reduction of approximately 36%. And what is more, in the last phase the results we attained on the Aurora-4 corpus were competitive with similar results given in the literature. There is still room, however, for improvement. For one, instead of dropping whole frequency bands, some heuristics could be applied to select the least noise-robust parts of the spectrogram, and drop only those. Furthermore, in our later experiments filter coefficients were initialised randomly, as it is likely that because of the parameter values having been changes, our earlier filter sets would not have performed as well. However, it may be possible in the future to find a better initial filter set, as we saw earlier that a proper initialisation can make a difference to the performance.

# Chapter 7

# Summary in Hungarian

Ebben a tanulmányban a zajtűrő beszédfelismerést vizsgáltuk a spektro-temporális feldolgozáson keresztül. Ennek érdekében az első fejezetben először ismertettük a HMM/ANN rendszerben történő beszédfelismerés fontosabb fogalmait, úgymint a beszédjel előfeldolgozása, a spektrális reprezentáció, a jellemzőkinyerés, és maga a HMM/ANN modell. Bemutattuk továbbá a nyelvi modellt, és annak kapcsolatát az akusztikus modellhez. Valamint tárgyaltunk több kulcsfontosságú módszert és eszközt (úgymint beszédadatbázisok), amelyek szintén szükségesek voltak a kísérleteinkhez. Az ezt követő fejezetekben több spektro-temporális feldolgozási módszert vizsgáltunk. Ezen fejezetek közül mind megelőlegezi a rákövetkezőket, hasonlóan, minden új fejezet épít az azt megelőző fejezetekben megismert elgondolásokra.

## 7.1 Spektro-temporális jellemzőkinyerés

A fejezetben a spektro-temporális feldolgozás alapvető felhasználását vizsgáltuk. Ebben a megközelítésben a bevezető fejezetben felvázolt szokványos feldolgozási lépések közül egyedül a másodlagos jellemzőkinyerésen változtattunk. A jellemzőket ezen szakaszban olyan ablakokból nyertük ki, melyek mind az idő-, mind a frekvenciatartományban korlátozottak. Miután azonban kinyertük ezeket a jellemzőket, a későbbi szakaszokban ugyan úgy használtuk fel őket, mint tettük azt korábban, spektro-temporális feldolgozás nélkül. A jellemzők kinyerésére több módszer adott, melyek közül kettőt röviden tárgyaltunk is a bevezető fejezetben. Jelen fejezetben két további módszert vizsgáltunk: a 2D DCT megközelítést, és a Gábor szűrők alkalmazását.

A 2D DCT együthatókkal végzet kísérleteinkben először megpróbáltuk megtalálni az optimális paraméter értékeket a spektro-temporális jellemzőkinyeréshez. Ezen kezdeti kísérletek után összehasonlítottuk a 2D DCT jellemzőinkkel elért felismerési eredményeinket a TIMIT adatbázison azokkal, melyeket a hagyományos MFCC jellemzőkkel értünk el. Megmutattuk, hogy a 2D DCT jellemzők használatával hasonló vagy jobb eredményeket tudunk elérni, mint az MFCC jellemzőkkel. Továbbá a két jellemzőkészlet egyszerű de elegáns kombinációjával a tiszta beszéd esetén is sikerült javítani a felismerési eredményeket.

A Gábor szűrőkkel végzett munka során először különböző jellemzőkiválasztási módszereket vizsgáltunk a megfelelő szűrőkészlet előállításához. Ezután egy angol és egy magyar nyelvű beszédadatbázison végzett kísérletek eredményeinek felhasználásával összehasonlítottuk különböző szűrőkészletek teljesítményét. Azt találtuk, hogy az általunk egyszerű heurisztikákat követve megalkotott szűrőkészlet érte el a legjobb eredményeket. Nem csak a többi szűrőkészletnél teljesített jobban a TIMIT adatbázison tiszta és zajjal szennyezett beszéd esetén is, de jobban teljesített az MFCC jellemzőkészletnél is. Hasonló eredményeket kaptunk a “Szeged” magyar híradós beszédadatbázison is. Ezen eredményeket felhasználtuk az automatikus jellemzőkiválasztás esetleges problémáinak tárgyalása során.

## 7.2 A Spektro-temporális szűrők és neuronhálók együttes tanítása

Az automatikus jellemzőkiválasztás nehézségei motiváltak abban, hogy egy új jellemzőkiválasztási módszert vezessünk be. A jelen fejezetben vizsgált együttes tanítási keretrendszer oly módon működik, hogy a jellemzőkinyerési lépést a neuronháló alsó rétegébe integrálja, így az adott réteg súlyainak tanításával egyúttal a megfelelő szűrőket is tanítjuk. E módszer használatával hatékonyan kombináltuk a másodlagos jellemzőkinyerési és a neuronháló tanítási fázist. Így kísérleteinkben konzisztensen alacsonyabb fonémafelismerési hibaszázalékokat tudtunk elérni mint amikor a két fázis külön került végrehajtásra. Továbbá, a neuronhálókkal kapcsolatos új eredmények felhasználásával számos módosítást vezettünk be az együttes tanítási rendszerbe. Ezen módosítások segítségével sikerült tovább csökkentenünk az elért hibaszázalékokat, demonstrálva a keretrendszer flexibilitását. Kísérleteink megmutatták, hogy a kezdeti szűrők tanítása valóban előnyös, és a legtöbb esetben jobb felismerési eredményhez vezet. Ám azt is megmutatták, hogy a kezdeti szűrők tanításának hatékonysága ellenére hasznos, ha a szűrőket megfelelően inicializáljuk a folyamat elején.

## 7.3 A beszéd többsávos feldolgozása spektro-temporális jellemzőkinyeréssel

Jelen fejezetben a többsávos és a spektro-temporális feldolgozás nyilvánvaló kompatibilitása által indíttatva kombináltuk a két megközelítést. Először a TIMIT adatbázison egy fonémafelismerési feladat segítségével demonstráltuk, hogy mind a 2D DCT jellemzőkészlettel, mind a második fejezetben bemutatott Gábor szűrőkészlettel jobb eredményeket érhetünk el a többsávos megközelítés felhasználásával, mint nélküle. Szemben több, az irodalomban található megállapítással, azt találtuk, hogy a többsávos feldolgozás előnye fennáll tiszta beszédjel esetén, valamint zajok széles skálájával szennyezett beszédjel esetén is. Azt is demonstráltuk, hogy mély hálók használatával és a konvolúció bevezetésével az elért hibaszázalékok tovább csökkenthetők. Ezen kísérletekben egy technikai részletet – az alacsony frekvenciatartomány tükrözése – is megvizsgáltunk amit a felismerési folyamatban korábban alkalmaztunk, és mivel azt találtuk, hogy alkalmazása előnnyel nem jár, a további kísérletekből elhagytuk ezt a részletet.

A többsávos feldolgozás megközelítését ötvöztük a harmadik fejezetben bemutatott együttes tanítási módszerrel is. Az így kapott módszer teljesítményét az Aurora-4 adatbázison vizsgáltuk meg, tiszta tanítási adatok esetére. Azt találtuk, hogy a többsávos feldolgozás nem csak jelentősen javítja az együttes tanítás eredményeit, de olyan alacsony hibaszázalékok elérését teszik lehetővé, melyek – publikálásuk idején – a legjobb eredmények között voltak az adott feladatra. A kísérletek során arra is fény derült, milyen nehézségekkel jár a megfelelő rekombináció, ezért a következő fejezetben egyik célunk jobb módszert találni a többsávos feldolgozás és az együttes tanítás integrációjára.

## 7.4 Sáv "dropout"

A fejezet első részében az együttes tanítási módszer paraméter értékeit vizsgáltuk felül. Ezt azzal a céllal tettük, hogy demonstráljuk, a módszer megfelelő paraméterezéssel nem csak a korábbi eredményeken tudunk javítani, de olyan eredményeket is el tudunk érteni, melyek versenyképesek a szakirodalomban publikált más módszerek eredményeivel. Ennek érdekében bevezettünk két módosítást a keretrendszerbe, majd a TIMIT valamint Aurora-4 beszédadatbázisok használatával megvizsgáltuk ezen módosítások hatását felismerési eredményekre. A kapott eredmények demonstrálták, hogy a módosítások jó eredményre vezettek mind a korábbi eredményeinkkel, mind az irodalomban található eredményekkel összevetve.

Végül az együttes tanítási keretrendszert egy olyan módszerrel egészítettük ki, melyet részben az úgynevezett input dropout módszer, részben pedig a többsávos feldolgozás inspirált. Az input dropout módszert úgy módosítottuk, hogy az teljes frekvenciatartományokat hagyott figyelmen kívül tanítás közben. Ezzel a módszerrel azt próbáltuk elérni, hogy – a többsávos feldolgozáshoz hasonlóan – javítsuk a betanított rendszer hibatűrését arra kényszerítve azt, hogy ne támaszkodjon a teljes frekvenciatartományra. Módszerünket az Aurora-4 beszédadatbázison értékeltük ki tiszta valamint zajos tanítási adatok esetére, mind mel spektrális jellemzők, mind pedig ARMA jellemzők használatával. Az eredmények azt mutatják, hogy zajos tanítási adatok elérhetősége esetén a javasolt módszer nem csökkenti, vagy a hagyományos input módszernél nem jobban csökkenti az elért hibaszázalékokat. Ám amikor csak tiszta tanítási adatok állnak rendelkezésre, a javasolt módszer jelentősen csökkenti a kapott hibaszázalékokat mindkét vizsgált spektrális reprezentáció esetében. Végeredményben az általunk leírt módszer az ARMA jellemzőkkel használva olyan hibaszázalékokat produkál, melyek legjobb tudásunk szerint a legjobb eredmények között tarthatók számon.

## 7.5 Következtetések és jövőbeni munka

Ebben a tanulmányban számos spektro-temporális technikát vizsgáltunk meg a zajtűrő beszédfelismerés érdekében. Az egymást követő kísérletekben használt módszerek egyre összetettebbek, ám egyre eredményesebbek is. A TIMIT adatbázison például a kezdeti huszonkilenc százalékos hibaarány végül több mint tíz százalékpontot csökkent (tizennyolc és fél százalékra), ami közel harminchat százalékos relatív hibaarány-csökkenésnek felel meg. Ami még fontosabb, az utolsó szakaszban elért eredmények az Aurora-4 adatbázison versenyképesek az irodalomban talált hasonló módszerek eredményeivel. Számos területen lehetne azonban még előre lépni. Például ahelyett, hogy a frekvenciatartományok egészére alkalmaznánk az input dropout módszerét, heurisztikák segítségével megkísérelhetnénk megtartani azokat a részeket, melyekről feltételezhetjük, hogy a zajnak a leginkább ellenállnak. Továbbá, mivel láthattuk a jótékony hatását annak, amikor az együttes tanítási keretrendszerben a szűrőket megfelelően inicializáltuk, érdemes lehet a keretrendszer megváltozott paramétereihez illeszkedő kezdeti szűrőkészletet keresni, az eredmények további javítása érdekében.

# Bibliography

[1] NVIDIA Tesla K20c. `https://www.techpowerup.com/gpudb/564/tesla-k20c`. Accessed: 2017-07-07.

[2] A. Waibel, T. Hanazawa, G. H. K. S. K. L. Phoneme recognition using time-delay neural networks. *IEEE Trans. ASSP 37*, 3 (1989), 328–339.

[3] Abdel-Hamid, O., Deng, L., and Yu, D. Exploring Convolutional Neural Network structures and optimization techniques for speech recognition. In *Proc. Interspeech* (2013), pp. 3366–3370.

[4] Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G., and Yu, D. Convolutional Neural Networks for speech recognition. *IEEE/ACM Trans. ASLP 22*, 10 (2014), 1533–1545.

[5] Aertsen, A. M., and Johannesma, P. I. M. Spectro-temporal receptive fields of auditory neurons in the grassfrog. *Biol. Cybern. 38*, 4 (November 1980), 223–234.

[6] Aertsen, A. M., Johannesma, P. I. M., and Hermes, D. J. Spectro-temporal receptive fields of auditory neurons in the grassfrog. *Biol. Cybern. 38*, 4 (November 1980), 235–248.

[7] Allen, J. B. How do humans process and recognize speech? *IEEE Transactions on Speech and Audio Processing 2*, 4 (October 1994), 567–577.

[8] Allison, B., Guthrie, D., and Guthrie, L. Another look at the data sparsity problem. In *Proc. TSD* (2006), pp. 327–334.

[9] Applebaum, T., Hanson, B., and Wakita, H. Weighted cepstral distance measures in vector quantization based speech recognizers. In *Proc. ICASSP* (1987), vol. 12, pp. 1155–1158.

[10] Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B. Deep Neural Network language models. In *Proc. NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT* (Stroudsburg, PA, USA, 2012), WLM '12, Association for Computational Linguistics, pp. 20–28.

[11] Aubert, X. L. An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech & Language 16*, 1 (2002), 89–114.

[12] Baby, D., Gemmeke, J. F., Virtanen, T., and Van hamme, H. Exemplar-based speech enhancement for Deep Neural Network based automatic speech recognition. In *Proc. ICASSP* (2015), pp. 4485–4489.

[13] Baby, D., and Van hamme, H. Investigating modulation spectrogram features for Deep Neural Network-based automatic speech recognition. In *Proc. Interspeech* (2015), pp. 2479–2483.

[14] Beal, M., Ghahramani, Z., and Rasmussen, C. The infinite Hidden Markov Model. In *Proc. NIPS* (2002), pp. 577–584.

[15] Besacier, L., and Bonastre, J. Subband architecture for automatic speaker recognition. *Signal Processing 80*, 7 (2000), 1245–1259.

[16] Biem, A., Mcdermott, E., and Katagiri, S. A discriminative filter bank model for speech recognition. In *Proc. ICASSP* (1996), pp. 545–548.

[17] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., New York, NY, USA, 1995.

[18] Bourland, H., and Wellekens, C. J. Links between markov models and multilayer perceptrons. *IEEE Trans. Pattern Anal. Mach. Intell. 12*, 12 (December 1990), 1167–1178.

[19] Bourlard, H., and Dupont, S. A new ASR approach based on independent processing and recombination of partial frequency bands. In *Proc. ICSLP* (1996), pp. 426–429.

[20] Bourlard, H., and Dupont, S. Subband-based speech recognition. In *Proc. ICASSP* (1997), pp. 1251–1254.

[21] Bourlard, H., Heřmanský, H., and Morgan, N. Towards increasing speech recogognition error rates. *Speech Communication 18*, 3 (1996), 205–231.

[22] Bourlard, H. A., and Morgan, N. *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[23] Bouthillier, X., Konda, K., Vincent, P., and Memisevic, R. Dropout as data augmentation. *ArXiv e-prints* (2015).

[24] Bouvrie, J., Ezzat, T., and Poggio, T. Localized spectro-temporal cepstral analysis of speech. In *Proc. ICASSP* (2008), pp. 4733–4736.

[25] Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. Large language models in machine translation. In *Proc. EMNLP-CoNLL* (2007), pp. 858–867.

[26] Brill, E., Florian, R., Henderson, J. C., and Mangu, L. Beyond n-grams: Can linguistic sophistication improve language modeling? In *Proc. ACL* (1998), pp. 186–190.

[27] Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. Class-based n-gram models of natural language. *Computational Linguistics 18*, 4 (1992), 467–480.

[28] Carpenter, B. Human versus machine: Psycholinguistics meets ASR. In *Proc. ASRU* (1999), pp. 225–228.

[29] Cerisara, C., and Fohr, D. Multi-band automatic speech recognition. *Computer Speech and Language 15*, 2 (2001), 151–174.

[30] Chang, S., and Wegmann, S. On the importance of modeling and robustness for Deep Neural Network feature. In *Proc. ICASSP* (2015), pp. 4530–4534.

[31] Chang, S.-Y., Meyer, B. T., and Morgan, N. Spectro-temporal features for noise-robust speech recognition using power-law nonlinearity and power-bias subtraction. In *Proc. ICASSP* (2013), pp. 7063–7067.

[32] Chang, S.-Y., and Morgan, N. Robust CNN-based speech recognition with Gabor filter kernels. In *Proc. Interspeech* (2014), pp. 905–909.

[33] Chen, S. F., and Goodman, J. An empirical study of smoothing techniques for language modeling. In *Proc. ACL* (1996), pp. 310–318.

[34] Chen, W., Hsieh, C., and Lai, E. Multiband approach to robust text-independent speaker identification. *IJCLCLP 9*, 2 (2004), 63–76.

[35] Chen, W., and Pratt, W. Scene adaptive coder. *IEEE Transactions on Communications 32*, 3 (March 1984), 225–232.

[36] Chi, T., Ru, P., and Shamma, S. A. Multiresolution spectrotemporal analysis of complex sounds. *J. Acoust. Soc. Am. 118*, 2 (2005), 887–906.

[37] Church, K. W., and Gale, W. A. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech and Language 5* (1991), 19–54.

[38] Clarkson, P., and Moreno, P. J. On the use of support vector machines for phonetic classification. In *Proc. ICASSP* (1999), pp. 585–588.

[39] Cox, S. Speaker normalization in the MFCC domain. In *Proc. Interspeech* (2000), pp. 853–856.

[40] Cui, J. *Integrating linguistic and statistical knowledge in language modeling.* PhD thesis, Johns Hopkins University, 2008.

[41] Cui, X., Goel, V., and Kingsbury, B. Data augmentation for Deep Neural Network acoustic modeling. In *Proc. ICASSP* (2014), pp. 5619–5623.

[42] Dahl, G. E., Sainath, T. N., and Hinton, G. E. Improving Deep Neural Networks for LVCSR using Rectified Linear Units and dropout. In *Proc. ICASSP* (2013), pp. 8609–8613.

[43] Damper, R., and Higgins, J. E. Improving speaker identification in noise by subband processing and decision fusion. *Pattern Recogn. Lett. 24*, 13 (2003), 2167–2173.

[44] Deng, L., Abdel-Hamid, O., and Yu, D. A Deep Convolutional Neural Network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In *Proc. ICASSP* (2013), pp. 6669 – 6673.

[45] Depireux, D., Simon, J., Klein, D., and Shamma, S. Spectro-temporal response field characterization with dynamic ripples in ferret primary auditory cortex. *J. Neurophysiol. 85* (2001), 1220–1234.

[46] Driesen, J. *Discovering Words in Speech using Matrix Factorization.* PhD thesis, Arenberg Doctoral School of Science, Engineering & Technology; Faculty of Engineering; Departement Electrotechniek — ESAT, 2012.

[47] Duchnowski, P. *A New Structure for Automatic Speech Recognition.* PhD thesis, MIT, 1993.

[48] Evermann, G., and Woodland, P. C. Large vocabulary decoding and confidence estimation using word posterior probabilities. In *Proc. ICASSP* (2000), pp. 1655–1658.

[49] Ezzat, T., Bouvrie, J. V., and Poggio, T. A. Spectro-temporal analysis of speech using 2D Gabor filters. In *Proc. Interspeech* (2007), pp. 506–509.

[50] Falk, T. H., and Chan, W. Spectro-temporal features for robust far-field speaker identification. In *Proc. Interspeech* (2008), pp. 634–637.

[51] Fletcher, H. *Speech and hearing in communication.* Bell Telephone Laboratories series. Van Nostrand, 1953.

[52] Fontaine, V., and Bourlard, H. Speaker-dependent speech recognition based on phone-like units models — application to voice dialing. Idiap-RR Idiap-RR-09-1996, IDIAP, 1996.

[53] Fux, T., and Jouvet, D. Evaluation of PNCC and extended spectral subtraction methods for robust speech recognition. In *Proc. EUSIPCO* (2015), pp. 1416–1420.

[54] Gabor, D. Theory of communication. *Journal IEE 93*, 26 (November 1946), 429–457.

[55] Gale, W. A., and Church, K. W. Estimation procedures for language context: poor estimates are worse than none. In *Proc. COMP-STAT* (1990), pp. 69–74.

[56] Gale, W. A., and Church, K. W. What's wrong with adding one? In *Corpus-Based Research into Language*, N. Oostdijk and P. de Haan, Eds. Rodolpi, 1994.

[57] Ganapathy, S. Robust speech processing using ARMA spectrogram models. In *Proc. ICASSP* (2015), pp. 5029–5033.

[58] Ganchev, T., Fakotakis, N., and Kokkinakis, G. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proc. SPECOM* (2005), pp. 191–194.

[59] Garofolo, J., Lamel, L. F., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., and Zue, V. TIMIT acoustic-phonetic continuous speech corpus, 1993.

[60] Gelbart, D., Kleinschmidt, M., and Meyer, B. T. Gabor feature extraction for automatic speech recognition. `http://www1.icsi.berkeley.edu/Speech/papers/gabor/`. Accessed: 2013-10-22.

[61] Geoffrey Zweig, M. P. Advances in large vocabulary speech recognition. *Advances in Computers, Elsevier Science* (January 2004).

[62] Gerosa, M., and Federico, M. Coping with out-of-vocabulary words: Open versus huge vocabulary ASR. In *Proc. ICASSP* (2009), pp. 4313–4316.

[63] Glorot, X., and Bengio, Y. Understanding the difficulty of training Deep Feedforward Neural Networks. In *Proc. AISTATS* (2010), pp. 249–256.

[64] Glorot, X., Bordes, A., and Bengio, Y. Deep Sparse Rectifier Neural Networks. In *Proc. AISTATS* (2011), G. J. Gordon and D. B. Dunson, Eds., vol. 15, Journal of Machine Learning Research - Workshop and Conference Proceedings, pp. 315–323.

[65] Golik, P., Doetsch, P., and Ney, H. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Proc. Interspeech* (2013), pp. 1756–1760.

[66] Gramss, T. Fast algorithms to find invariant features for a word recognizing Neural Net. In *Proc. ICANN* (1991), pp. 180–184.

[67] Gramss, T., and Strube, H. W. Recognition of isolated words based on psychoacoustics and neurobiology. *Speech Communication 9*, 1 (1990), 35–40.

[68] Graves, A., Mohamed, A., and Hinton, G. E. Speech recognition with Deep Recurrent Neural Networks. In *Proc. ICASSP* (2013), pp. 6645–6649.

[69] Hagen, A., Bourlard, H., and Morris, A. Adaptive ML-weighting in multi-band recombination of Gaussian mixture ASR. In *Proc. ICASSP* (2001), pp. 257–260.

[70] Hagen, A., Morris, A., and Bourlard, H. Subband-based speech recognition in noisy conditions the full combination approach. Tech. Rep. Idiap-RR-15-1998, IDIAP, 1998.

[71] Hagen, A., Morris, A., and Bourlard, H. From multi-band full combination to multi-stream full combination processing in robust ASR. In *Proc. ISCA Tutorial and Research Workshop ASR* (2000).

[72] Halavati, R., and Shouraki, S. B. Reducing speech recognition costs: By compressing the input data. In *Proc. IEEE Conf. of Intelligent Systems* (2012), pp. 102–107.

[73] Halberstadt, A. K., and Glass, J. R. Heterogeneous Measurements and Multiple Classifiers for Speech Recognition. In *Proc. ICSLP* (1998).

[74] Hamamoto, Y., Uchimura, S., Watanabe, M., Yasuda, T., Mitani, Y., and Tomita, S. A Gabor filter-based method for recognizing handwritten numerals. *Pattern Recognition 31*, 4 (1998), 395–400.

[75] Hariharan, R., Kiss, I., and Viikki, I. Noise robust speech parameterization using multiresolution feature extraction. *IEEE Trans. Speech and Audio Processing 9*, 8 (2001), 856–865.

[76] Harris, F. J. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE 66*, 1 (January 1978), 51–83.

[77] He, L., Lech, M., Maddage, N. C., and Allen, N. Stress detection using speech spectrograms and sigma-pi neuron units. In *Proc. ICNC* (2009), pp. 260–264.

[78] Heřmanský, H. Human speech perception: Some lessons from automatic speech recognition. In *Proc. TSD* (2001), pp. 187–196.

[79] Heřmanský, H. Perceptual linear predictive (PLP) analysis of speech. *J. Acoust. Soc. Am. 57*, 4 (1990), 1738–52.

[80] Heřmanský, H. Should recognizers have ears? *Speech Communication 25*, 1-3 (1998), 3–27.

[81] Heřmanský, H., and Sharma, S. TRAPS-classifiers of temporal patterns. In *Proc. ICSLP* (1998), pp. 1003–1006.

[82] Heřmanský, H., Timbrewala, S., and Pavel, M. Towards ASR on partially corrupted speech. In *Proc. ICSLP* (1996), pp. 464–465.

[83] Hewlett, N., and Beck, J. *An Introduction to the Science of Phonetics.* An Introduction to the Science of Phonetics. Taylor & Francis, 2006.

[84] Hinton, G., Deng, L., Yu, D., Abdel-rahman, M., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Dahl, G., and Kingsbury, B. Deep Neural Networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine 29*, 6 (2012), 82–97.

[85] Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving Neural Networks by preventing co-adaptation of feature detectors. *CoRR abs/1207.0580* (2012).

[86] Hirsch, H.-G. Fant: Filtering and noise-adding tool. http://dnt.kr.hs-niederrhein.de/download.html.

[87] Hönig, F., Stemmer, G., Hacker, C., and Brugnara, F. Revising perceptual linear prediction (plp). In *Proc. Interspeech* (2005), pp. 2997–3000.

[88] Hoffmeister, B., Heigold, G., Rybach, D., Schlüter, R., and Ney, H. WFST enabled solutions to ASR problems: Beyond HMM decoding. *IEEE Transactions on Audio, Speech, and Language Processing 20*, 2 (February 2012), 551–564.

[89] Huang, G., Zhu, Q., and Siew, C. Extreme learning machine: A new learning scheme of feedforward Neural Networks. In *Proc. INT. JOINT CONF. NEURAL NETW* (2006), pp. 985–990.

[90] Huang, J.-T., Li, J., and Gong, Y. An analysis of Convolutional Neural Networks for speech recognition. In *Proc. ICASSP* (2015), pp. 4989–4993.

[91] Huang, L.-L., Shimizu, A., and Kobatake, H. Robust face detection using Gabor filter features. *Pattern Recogn. Lett. 26*, 11 (August 2005), 1641–1649.

[92] Huang, X., Acero, A., and Hon, H.-W. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.

[93] Huang, X., and Lee, K. F. On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition. *IEEE Transactions on Speech and Audio Processing 1*, 2 (1993), 150–157.

[94] Huang, Z., Chang, Y., Long, B., Crespo, J.-F., Dong, A., Keerthi, S., and Wu, S.-L. Iterative viterbi a* algorithm for k-best sequential decoding. In *Proc. ACL* (2012), pp. 611–619.

[95] Jaitly, N., and Hinton, G. E. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *Proc. ICASSP* (2011), IEEE, pp. 5884–5887.

[96] Janin, A., Ellis, D., and Morgan, N. Multi-stream speech recognition: ready for prime time? In *Proc. Eurospeech* (1999), pp. 591–594.

[97] Jelinek, F., Bahl, L. R., and Mercer, R. L. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Trans. Information Theory 21*, 3 (1975), 250–256.

[98] Jelinek, F., Lafferty, J. D., and Mercer, R. L. Basic methods of probabilistic context-free grammars. In *Speech Recognition and Understanding. Recent Advances, Trends, and Applications*, P. Laface and R. DeMori, Eds., vol. 75. Springer-Verlag, 1992, pp. 345–360.

[99] Jelinek, F., and Mercer, R. L. Interpolated estimation of markov source parameters from sparse data. In *Proc. Pattern Recognition in Practice* (1980), pp. 381–397.

[100] Johnson, W. Probability: deductive and inductive problems. *Mind 41* (1932), 421–423.

[101] Jones, J. P., and Palmer, L. A. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology 58*, 6 (1987), 1233–1258.

[102] Jun, D., Qing, W., Tian, G., Yong, X., Li-Rong, D., and Chin-Hui, L. Robust speech recognition with speech enhanced Deep Neural Networks. In *Proc. Interspeech* (2014), pp. 616–620.

[103] Junqua, J.-C. Evaluation of ASR front ends in speaker-dependent and speaker-independent recognition. *J. Acoust. Soc. Am. 81*, S1 (1987), S93–S93.

[104] Jurafsky, D., and Martin, J. H. *An introduction to natural language processing, computational linguistics, and speech recognition.* Prentice Hall, 2007.

[105] Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchman, G., and Morgan, N. Using a stochastic context-free grammar as a language model for speech recognition. In *Proc. ICASSP* (1995), pp. 189–192.

[106] Kanedera, N., Arai, T., Heřmanský, H., and Pavel, M. On the relative importance of various components of the modulation spectrum for automatic speech recognition. *Speech Communication 28*, 1 (1999), 43–55.

[107] Katz, S. M. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing 35*, 3 (1987), 400–401.

[108] Kelly, F., and Harte, N. Auditory features revisited for robust speech recognition. In *Proc. ICPR* (2010), pp. 4456–4459.

[109] Kelly, F., and Harte, N. A comparison of auditory features for robust speech recognition. In *Proc. EUSIPCO* (2010), pp. 1968–1972.

[110] Ketabdar, H., and Bourlard, H. Enhanced phone posteriors for improving speech recognition systems. *IEEE Transactions on Audio, Speech, and Language Processing 18*, 6 (2010), 1094–1106.

[111] Kim, C. *Signal processing for robust speech recognition motivated by auditory processing.* PhD thesis, Carnegie Mellon University, 2010.

[112] Kim, C., and Stern, R. M. Feature extraction for robust speech recognition using a power-law nonlinearity and power-bias subtraction. In *Proc. Interspeech* (2009), pp. 28–31.

[113] Kim, C., and Stern, R. M. Power-normalized cepstral coefficients (PNCC) for robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 24*, 7 (July 2016), 1315–1329.

[114] Kipyatkova, I., and Karpov, A. Lexicon size and language model order optimization for russian LVCSR. In *Proc. SPECOM* (2013), pp. 219–226.

[115] Kleinschmidt, M. Methods for capturing spectro-temporal modulations in automatic speech recognition. *Acustica united with acta acustica 88* (2002), 416–422.

[116] Kleinschmidt, M. *Robust Speech Recognition Based on Spectro-Temporal Processing.* PhD thesis, Carl-von-Ossietzky Universität Oldenburg, 2002.

[117] Kleinschmidt, M. Spectro-temporal Gabor features as a front end for automatic speech recognition. In *Proc. 3rd European Congress on Acoustics - Forum Acusticum 2002* (2002), pp. CD–ROM, 6 pages.

[118] Kleinschmidt, M., and Gelbart, D. Improving word accuracy with Gabor feature extraction. In *Proc. ICSLP* (2002).

[119] Kleinschmidt, M., and Hohmann, V. Sub-band SNR estimation using auditory feature processing. *Speech Communication 39*, 1-2 (2003), 47–63.

[120] Kneser, R., and Ney, H. Improved backing-off for M-gram language modeling. In *Proc. ICASSP* (1995), pp. 181–184.

[121] Ko, T., Peddinti, V., Povey, D., and Khudanpur, S. Audio augmentation for speech recognition. In *Proc. Interspeech* (2015), pp. 3586–3589.

[122] Kocsor, A., and Tóth, L. Application of kernel-based feature space transformations and learning methods to phoneme classification. *Appl. Intell. 21*, 2 (2004), 129–142.

[123] Kovács, G., and Tóth, L. Phone recognition experiments with 2D-DCT spectro-temporal features. In *Proceedings of the International Symposium on Applied Computational Intelligence and Informatics* (2011), pp. 143–146.

[124] Kovács, G., and Tóth, L. The joint optimization of spectro-temporal features and Neural Net classifiers. In *Proc. TSD* (2013), vol. 8082 of *Lecture Notes in Computer Science*, Springer, pp. 552–559.

[125] Kovács, G., and Tóth, L. Joint optimization of spectro-temporal features and Deep Neural Nets for robust automatic speech recognition. *Acta Cybernetica 22*, 1 (2015), 117–134.

[126] Kovács, G., Tóth, L., and Van Compernolle, D. Selection and enhancement of Gabor filters for automatic speech recognition. *IJST 18*, 1 (2015), 1–16.

[127] Kryter, K. D. Speech bandwith compression through spectrum selection. *J. Acoust. Soc. Am. 32* (1960), 547.

[128] Lamel, L. F. Some perspectives on speech database development. In *Proc. of the ESCA Workshop on Speech Input/Output Assessment and Speech Databases* (1989).

[129] Lamel, L. F., Kassel, R., and Seneff, S. Speech database development: design and analysis of the acoustic-phonetic corpus. In *Proc. DARPA Speech Recognition Workshop, Report no. SAIC-86/1546* (1986).

[130] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.

[131] Lee, C., Hyun, D., Choi, E., Go, J., and Lee, C. Optimizing feature extraction for speech recognition. *IEEE Transactions on Speech and Audio Processing 11*, 1 (2003), 80–87.

[132] Lee, H., Pham, P., Largman, Y., and Ng, A. Y. Unsupervised feature learning for audio classification using Convolutional Deep Belief Networks. In *Proc. Adv. Neural Inf. Process. Syst.* (2009), p. 1096–1104.

[133] Lee, K.-F., and Hon, H.-W. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on accoustics, speech and signal processing 37*, 11 (1989), 1641–1648.

[134] Lee, S., Fang, S., Hung, J., and Lee, L. Improved MFCC feature extraction by PCA-optimized filter-bank for speech recognition. In *Proc. ASRU* (2001), pp. 49–52.

[135] Levinson, S., Rabiner, L., and Sondhi, M. Speaker independent isolated digit recognition using Hidden Markov Models. In *Proc. ICASSP* (1983), pp. 1049–1052.

[136] Li, B., and Sim, K. C. Improving robustness of Deep Neural Networks via spectral masking for automatic speech recognition. In *Proc. ASRU* (2013), pp. 279–284.

[137] Li, S., Sun, L., and Lee, L. Improved phoneme recognition by integrating evidence from spectro-temporal and cepstral features. In *Proc. Interspeech* (2010), pp. 1177–1180.

[138] Lidstone, G. J. Note on the general case of the Bayes–Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries 8* (1920), 182–192.

[139] Lopes, C., and Perdigao, F. Phoneme recognition on the timit database. In *Speech Technologies*, P. I. Ipsic, Ed. InTech, 2011.

[140] Maganti, H. K., and Matassoni, M. Auditory processing-based features for improving speech recognition in adverse acoustic conditions. *EURASIP Journal on Audio, Speech, and Music Processing 2014*, 1 (2014), 21.

[141] Mallidi, S. H., and Heřmanský, H. Novel Neural Network based fusion for multistream ASR. In *Proc. ICASSP* (2016), pp. 5680–5684.

[142] Manolakis, D. G., Ingle, V. K., and Kogon, S. M. *Statistical and adaptive signal processing: spectral estimation, signal modeling, adaptive filtering, and array processing.* Norwood: Artech House, 2005.

[143] Marks, R. *Introduction to Shannon Sampling and Interpolation Theory.* Springer-Verlag, 1991.

[144] Martin, S., Liermann, J., and Ney, H. Algorithms for bigram and trigram word clustering. *Speech Communication 24*, 1 (1998), 19–37.

[145] Martínez, A. M. C., Moritz, N., and Meyer, B. T. Should Deep Neural Nets have ears? The role of auditory features in deep learning approaches. In *Proc. Interspeech* (2014), pp. 2435–2439.

[146] Martínez, A. M. C., and Schädler, M. Why do ASR systems despite Neural Nets still depend on robust features. In *Proc. Interspeech* (2016), pp. 1883–1887.

[147] Meseguer, N. A. Speech analysis for automatic speech recognition. Master's thesis, Department of Electronics and Telecommunications, Norwegian University of Science and Technology, 2009.

[148] Mesgarani, N., Thomas, S., and Heřmansky, H. A multistream multiresolution framework for phoneme recognition. In *Proc. Interspeech* (2010), pp. 318–321.

[149] Mesgarani, N., Thomas, S., and Heřmanský, H. Adaptive stream fusion in multistream recognition of speech. In *Proc. Interspeech* (2011), pp. 2329–2332.

[150] Meyer, B. T. *Human and automatic speech recognition in the presence of speech-intrinsic variations.* PhD thesis, Carl-von-Ossietzky Universität, 2009.

[151] Meyer, B. T. What's the difference? Comparing humans and machines on the Aurora2 speech database. In *Proc. Interspeech* (2013), pp. 2634–2638.

[152] Meyer, B. T., and Kollmeier, B. Optimization and evaluation of Gabor feature sets for ASR. In *Proc. Interspeech* (2008), pp. 906–909.

[153] Meyer, B. T., Wächter, M., Brand, T., and Kollmeier, B. Phoneme confusions in human and automatic speech recognition. In *Proc. Interspeech* (2007).

[154] Meyer, B. T., Wesker, T., Brand, T., Mertins, A., and Kollmeier, B. A human-machine comparison in speech recognition based on a logatome corpus. In *Proc. SRIV* (2006).

[155] Miao, Y., and Metze, F. Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training. In *Proc. Interspeech* (2013), pp. 2237–2241.

[156] Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent Neural Network based language model. In *Proc. Interspeech* (2010), pp. 1045–1048.

[157] Milner, B. A comparison of front-end configurations for robust speech recognition. In *Proc. ICASSP* (2002), pp. 797–800.

[158] Minsky, M., and Papert, S. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA, USA, 1969.

[159] Mirghafori, N. *A Multi-Band Approach to Automatic Speech Recognition.* PhD thesis, International Computer Science Institute, 1999.

[160] Mirghafori, N., and Morgan, N. Combining connectionist multi-band and full-band probability streams for speech recognition of natural numbers. In *Proc. ICSLP* (1998), pp. 743–746.

[161] Misra, H. *Multi-stream Processing for Noise Robust Speech Recognition.* PhD thesis, Swiss Federal Institute of Technology, 2006.

[162] Mohamed, A., Dahl, G. E., and Hinton, G. Acoustic modeling using deep belief networks. *Trans. Audio, Speech and Lang. Proc. 20*, 1 (January 2012), 14–22.

[163] Mohri, M. Weighted Finite-State Transducer algorithms. An overview. In *Formal Languages and Applications*, C. Martín-Vide, V. Mitrana, and G. Păun, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 551–563.

[164] Mohri, M., Pereira, F., and Riley, M. Weighted finite-state transducers in speech recognition. *Computer Speech and Language 16*, 1 (2002), 69 – 88.

[165] Mohri, M., Riley, M., Hindle, D., Ljolje, A., and Pereira, F. Full expansion of context-dependent networks in large vocabulary speech recognition. In *Proc. ICASSP* (1998), pp. 665–668.

[166] Morgan, N., and Bourlard, H. Continuous speech recognition using multilayer perceptrons with Hidden Markov Models, 1990.

[167] Morris, A., Hagen, A., Glotin, H., and Bourlard, H. Multi-stream adaptive evidence combination for noise robust ASR. *Speech Communication 34*, 1-2 (2001), 25–40.

[168] Müller, F. *Invariant features and enhanced speaker normalization for automatic speech recognition.* PhD thesis, University of Lübeck, 2013.

[169] Murveit, H., Butzberger, J. W., Digalakis, V. V., and Weintraub, M. Large-vocabulary dictation using SRI's decipher speech recognition system: Progressive-search techniques. In *Proc. ICASSP* (1993), pp. 319–322.

[170] Nakariyakul, S., and Casasent, D. P. Improved forward floating selection algorithm for feature subset selection. In *2008 International Conference on Wavelet Analysis and Pattern Recognition* (2008), vol. 2, pp. 793–798.

[171] Narayanan, A., and Wang, D. Joint noise adaptive training for robust automatic speech recognition. In *Proc. ICASSP* (May 2014), pp. 2504–2508.

[172] Niesler, T. R., Whittaker, E. W. D., and Woodland, P. C. Comparison of part-of-speech and automatically derived categor-ybased language models for speech recognition. In *Proc. ICASSP* (1998), pp. 177–180.

[173] Okawa, S., Bocchieri, E., and Potamianos, A. Multi-band speech recognition in noisy environments. In *Proc. ICASSP* (1998), pp. 641–644.

[174] Palaz, D., Collobert, R., and Magimai-Doss, M. End-to-end phoneme sequence recognition using Convolutional Neural Networks. *CoRR abs/1312.2137* (2013).

[175] Parihar, N., and Picone, J. DSR front end LVCSR evaluation. Aurora Working Group AU/384/02, Institue for Signal and Information Processing, December 2002.

[176] Patterson, R. D., Robinson, K., Holdsworth, J., Mckeown, D., Zhang, C., and Allerhand, M. Complex sounds and auditory images. In *in Proc. 9th Int. Symp. Hearing Audit., Physiol. Perception* (1992), pp. 429–446.

[177] Paul, D., and Baker, J. The design of wall street journal-based CSR corpus. In *Proc. ICSLP* (1992), pp. 899–902.

[178] Peddinti, V., Chen, G., Povey, D., and Khudanpur, S. Reverberation robust acoustic modeling using i-vectors with Time Delay Neural Networks. In *Proc. Interspeech* (2015), pp. 3214–3218.

[179] Pickles, J. *An introduction to the Physiology of Hearing.* Academic Press, New York, USA, 1988.

[180] Picone, J. W. Signal modeling techniques in speech recognition. *Proceedings of the IEEE 81*, 9 (1993), 1215–1247.

[181] Pinto, J., Garimella, S., Magimai-Doss, M., Heřmanský, H., and Bourlard, H. Analysis of MLP-based hierarchical phoneme posterior probability estimator. *IEEE Transactions on Audio, Speech, and Language Processing 19*, 2 (February 2011), 225–241.

[182] Plahl, C., Sainath, T. N., Ramabhadran, B., and Nahamoo, D. Improved pre-training of deep belief networks using sparse encoding symmetric machines. In *Proc. ICASSP* (2012), pp. 4165–4168.

[183] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. The Kaldi speech recognition toolkit. In *Proc. ASRU* (2011).

[184] Pudil, P., Novovičová, J., and Kittler, J. Floating search methods in feature selection. *Pattern Recogn. Lett. 15*, 11 (November 1994), 1119–1125.

[185] Pundak, G., and Sainath, T. Lower frame rate Neural Network acoustic models. In *Proc. Interspeech* (2016), pp. 22–26.

[186] Qian, Y., Yin, M., You, Y., and Yu, K. Multi-task joint learning of Deep Neural Networks for robust speech recognition. In *Proc. ASRU* (2015), pp. 310–316.

[187] Qiu, A., Schreiner, C. E., and Escabí, M. A. Gabor analysis of auditory midbrain receptive fields: Spectro-temporal and binaural composition. *Journal of Neurophysiology 90*, 1 (2003), 456–476.

[188] rahman Mohamed, A., Hinton, G. E., and Penn, G. Understanding how deep belief networks perform acoustic modelling. In *Proc. ICASSP* (2012), IEEE, pp. 4273–4276.

[189] Rao, K. S., and Sarkar, S. *Stochastic Feature Compensation for Robust Speaker Verification.* Springer International Publishing, Cham, 2014, pp. 49–76.

[190] Rao, S., and Pearlman, W. A. Analysis of linear prediction, coding, and spectral estimation from subbands. *IEEE Transactions on Information Theory 42*, 4 (July 1996), 1160–1178.

[191] Renals, S., Morgan, N., Bourlard, H., Cohen, M., and Franco, H. Connectionist probability estimators in HMM speech recognition. *IEEE Trans. Speech and Audio Processing 2*, 1 (1994), 161–174.

[192] Rennie, S. J., Dognin, P. L., Cui, X., and Goel, V. Annealed dropout trained maxout networks for improved LVCSR. In *Proc. ICASSP* (2015), pp. 5181–5185.

[193] Riener, K. R., Warren, R. M., and Bashford, Jr., J. A. Novel findings concerning intelligibility of bandpass speech. *J. Acoust. Soc. Am. 91*, 4 (1992), 2339.

[194] Robinson, D. W., and Dadson, R. S. A re-determination of the equal-loudness relations for pure tones. *British Journal of Applied Physics 7* (May 1956), 166–181.

[195] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review 65*, 6 (1958), 386–408.

[196] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. MIT Press, Cambridge, MA, USA, 1988, pp. 673–695.

[197] Sainath, T. N., Kingsbury, B., rahman Mohamed, A., and Ramabhadran, B. Learning filter banks within a Deep Neural Network framework. In *Proc. ASRU* (2013), pp. 297–302.

[198] Sainath, T. N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A., Dahl, G., and Ramabhadran, B. Deep Convolutional Neural Networks for large-scale speech tasks. *Neural Networks 64* (2015), 39–48.

[199] Sainath, T. N., Ramabhadran, B., and Picheny, M. An exploration of large vocabulary tools for small vocabulary phonetic recognition. In *Proc. ASRU* (2009), pp. 359–364.

[200] Saon, G., Kurata, G., Sercu, T., Audhkhasi, K., Thomas, S., Dimitriadis, D., Cui, X., Ramabhadran, B., Picheny, M., Lim, L., Roomi, B., and Hall, P. English conversational telephone speech recognition by humans and machines. *CoRR abs/1703.02136* (2017).

[201] Schädler, M. R., Meyer, B. T., and Kollmeier, B. Spectro-temporal modulation subspace-spanning filter bank features for robust automatic speech recognition. *J. Acoust. Soc. Am. 131* (2012), 4134–4151.

[202] Scharenborg, O. Reaching over the gap: A review of efforts to link human and automatic speech recognition research. *Speech Communication 49*, 5 (2007), 336–347.

[203] Scherer, D., Müller, A., and Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proc. ICANN* (2010), pp. 92–101.

[204] Schwartz, R., and Chow, Y.-L. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proc. ICASSP* (1990), pp. 81–84.

[205] Schwenk, H., and Gauvain, J. L. Neural Network language models for conversational speech recognition. In *Proc. ICSLP* (2004), pp. 1215–1218.

[206] Seltzer, M. L., Yu, D., and Wang, Y. An investigation of Deep Neural Networks for noise robust speech recognition. In *Proc. ICASSP* (May 2013), pp. 7398–7402.

[207] Shiffman, D., and Marsh, Z. Neural Networks. In *The nature of code*, F. Shannon, Ed. D. Shiffman, 2012.

[208] Siniscalchi, S. M., Yu, D., Deng, L., and Lee, C. Exploiting Deep Neural Networks for detection-based speech recognition. *Neurocomputing 106* (2013), 148–157.

[209] Somol, P., Novovičová, J., and Pudil, P. Efficient feature subset selection and subset size optimization. In *Pattern Recognition Recent Advances*, A. Herout, Ed. InTech, 2010, ch. 4.

[210] Somol, P., Pudil, P., Novovičová, J., and Paclík, P. Adaptive floating search methods in feature selection. *Pattern Recogn. Lett. 20*, 11–13 (1999), 1157 – 1163.

[211] Stahl, S. *Stahl's Essential Psychopharmacology: Neuroscientific Basis and Practical Applications*. Cambridge medicine. Cambridge University Press, 2008.

[212] Stern, R. M. Applying physiologically-motivated models of auditory processing to automatic speech recognition. In *Proc. ISAAR* (2011), pp. 283–293.

[213] Stevens, S. S. On the Psychophysical Law. *Psychological Review 64*, 3 (1957), 153–181.

[214] Student. The probable error of a mean. *Biometrika 6*, 1 (1908), 1–25.

[215] Sun, Z., Bebis, G., and Miller, R. Evolutionary Gabor filter optimization with application to vehicle detection. In *Third IEEE International Conference on Data Mining* (2003), pp. 307–314.

[216] Tarján, B., and Mihajlik, P. On morph-based LVCSR improvements. In *2nd Workshop on Spoken Language Technologies for Under-Resourced Languages, SLTU* (2010), pp. 10–16.

[217] Theodoridis, S., and Koutroumbas, K., Eds. *Pattern Recognition*, 2nd ed. Elsevier Academic Press, 2003.

[218] Theunissen, F. E., Sen, K., and Doupe, A. J. Spectral-temporal receptive fields of nonlinear auditory neurons obtained using natural sounds. *J. Neurosci. 20* (March 2000), 2315–2331.

[219] Thompson, C., and Shure, L. *Image Processing Toolbox: For Use with MATLAB;[user's Guide]*. MathWorks, 1995.

[220] Tiitinen, H., Miettinen, I., Alku, P., and May, P. J. C. Transient and sustained cortical activity elicited by connected speech of varying intelligibility. *BMC Neuroscience 13*, 1 (2012), 157.

[221] Tino, P., Benuskova, L., and Sperduti, A. Artificial Neural Network models. In *Springer Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz, Eds. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 455–471.

[222] Tóth, L. Convolutional Deep Rectifier Neural Nets for phone recognition. In *Proc. Interspeech* (2013), pp. 1722–1726.

[223] Tóth, L. Phone recognition with Deep Sparse Rectifier Neural Networks. In *Proc. ICASSP* (2013), pp. 6985–6989.

[224] Tóth, L. Combining time- and frequency-domain convolution in Convolutional Neural Network-based phone recognition. In *Proc. ICASSP* (2014), pp. 190–194.

[225] Tóth, L. Phone recognition with hierarchical convolutional deep maxout networks. *EURASIP Journal on Audio, Speech and Music Processing 25* (2015).

[226] Tóth, L., and Grósz, T. A comparison of Deep Neural Network training methods for large vocabulary speech recognition. In *Proc. TSD* (2013), pp. 36–43.

[227] Tsai, D. Optimal Gabor filter design for texture segmentation using stochastic optimization. *Image and Vision Computing 19* (2001), 299–316.

[228] Tufekci, Z., and Gowdy, J. N. Subband feature extraction using lapped orthogonal transform for speech recognition. In *Proc. ICASSP* (2001), pp. 149–152.

[229] V. Peddinti, D. Povey, S. K. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Proc. Interspeech* (2015), pp. 3214–3218.

[230] Varga, A., and Steeneken, H. J. M. Assessment for automatic speech recognition ii: Noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech Commun. 12*, 3 (July 1993), 247–251.

[231] Veselý, K., Karafiát, M., and Grézl, F. Convolutive bottleneck network features for LVCSR. In *Proc. ASRU* (2011), IEEE Signal Processing Society, pp. 42–47.

[232] Vinyals, O., and Deng, L. Are sparse representations rich enough for acoustic modeling? In *Proc. Interspeech* (2012), pp. 2570–2573.

[233] Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor. 13*, 2 (September 2006), 260–269.

[234] von Ossietzky, C. Gabor filter bank features. `http://medi.uni-oldenburg.de/GBFB`. Accessed: 2013-09-15.

[235] Wickens, A. *Foundations of Biopsychology*, 2 ed. Pearson Education, Essex, UK, 2005.

[236] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X. A., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. *The HTK Book (for HTK Version 3.4).* Cambridge University Engineering Department, 2006.

[237] Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., and Hinton, G. E. On rectified linear units for speech processing. In *Proc. ICASSP* (2013), pp. 3517–3521.

[238] Zhao, S. Y., Ravuri, S. V., and Morgan, N. Multi-stream to many-stream: using spectro-temporal features for ASR. In *Proc. Interspeech* (2009), pp. 2951–2954.

[239] Zhao, X., and Wang, D. Analyzing noise robustness of MFCC and GFCC features in speaker identification. In *Proc. ICASSP* (2013), pp. 7204–7208.

[240] Zheng, N., Wang, N., Lee, T., and Ching, P. C. Speaker verification using complementary information from vocal source and vocal tract. In *Proc. ISCSLP* (2006), pp. 518–528.

[241] Zhu, Q., and Alwan, A. On the use of variable frame rate analysis in speech recognition. In *Proc. ICASSP* (2000), pp. 1783–1786.

[242] Zongker, D., and Jain, A. Algorithms for feature selection: An evaluation. In *Proc. ICPR* (1996), vol. 2, pp. 18–22.

# List of Publications

## Articles in Journals

1. Kovács, G., Tóth, L., Van Compernolle, D., and Ganapathy, S. Increasing the robustness of CNN acoustic models using autoregressive moving average spectrogram features and channel dropout. *Pattern Recognition Letters (2017), http://dx.doi.org/10.1016/j.patrec.2017.09.023*

2. Kovács, G., Tóth, L., and Van Compernolle, D. Selection and enhancement of Gabor filters for automatic speech recognition. *International Journal of Speech Technology 18*, 1 (2015), 1–16.

3. Kovács, G., and Tóth, L. Joint optimization of spectro-temporal features and Deep Neural Nets for robust automatic speech recognition. *Acta Cybernetica 22*, 1 (2015), 117–134.

## Articles in International Conferences

1. Kovács, G., Tóth, L., and Grósz, T. Robust multi-band ASR using Deep Neural Nets and spectro-temporal features. In *Proceedings of the International Conference on Speech and Computer (SPECOM)* (2014), Vol. 8773 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 386–393.

2. Kovács, G., and Tóth, L. The joint optimization of spectro-temporal features and neural net classifiers. In *Proceedings of the 16th International Conference on Text, Speech, and Dialogue (TSD)* (2013), Vol. 8082 of *Lecture Notes in Computer Science*, Springer, pp. 552–559.

3. Kovács, G., and Tóth, L. Phone recognition experiments with 2D-DCT spectro-temporal features. In *Proceedings of the IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)* (2011), pp. 143–146.

4. Kovács, G., and Tóth, L. Localised spectro-temporal features for noise-robust speech recognition. In *Proceedings of the IEEE International Joint Conferences on Computational Cybernetics and Technical Informatics (ICCC-CONTI)* (2010), pp. 481–485.

## Articles in Other Conferences

1. Kovács, G., and Tóth, L. Optimisation of spectro-temporal feature selection method integrated in Deep Neural Networks (in Hungarian). In *Proceedings of MSZNY* (2017), pp. 158–169.

2. Kovács, G., and Tóth, L. Multi-band noise robust speech recognition using Deep Neural Networks (in Hungarian). In *Proceedings of MSZNY* (2016), pp. 287–294.