TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Katariina Mahkonen

Efficient and Robust Methods for Audio and Video Signal Analysis

Julkaisu 1588 • Publication 1588

Tampere 2018

Tampereen teknillinen yliopisto. Julkaisu 1588
Tampere University of Technology. Publication 1588

Katariina Mahkonen

Efficient and Robust Methods for Audio and Video Signal Analysis

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Festia Building, Auditorium Pieni Sali 1, at Tampere University of Technology, on the 24th of October 2018, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2018

Doctoral candidate: Katariina Mahkonen
Laboratory of Signal Processing
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Finland

Supervisor: Tuomas Virtanen, professor
Laboratory of Signal Processing
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Finland

Instructor: Joni Kämäräinen, professor
Laboratory of Signal Processing
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Finland

Pre-examiners: Jorma Laaksonen, PhD
Department of Computer Science
Aalto University
Finland

Ville Hautamäki, PhD
Department of Computing Sciences
University of Eastern Finland
Finland

Opponents: Vesa Välimäki, PhD
Department of Signal Processing and acoustics
Aalto University
Finland

Jorma Laaksonen, PhD
Department of Computer Science
Aalto University
Finland

ISBN 978-952-15-4229-9 (printed)
ISBN 978-952-15-4234-3 (PDF)
ISSN 1459-2045

Abstract

This thesis presents my research concerning audio and video signal processing and machine learning. Specifically, the topics of my research include computationally efficient classifier compounds, automatic speech recognition (ASR), music dereverberation, video cut point detection and video classification.

Computational efficacy of information retrieval based on multiple measurement modalities has been considered in this thesis. Specifically, a cascade processing framework, including a training algorithm to set its parameters has been developed for combining multiple detectors or binary classifiers in computationally efficient way. The developed cascade processing framework has been applied on video information retrieval tasks of video cut point detection and video classification. The results in video classification, compared to others found in the literature, indicate that the developed framework is capable of both accurate and computationally efficient classification. The idea of cascade processing has been additionally adapted for the ASR task. A procedure for combining multiple speech state likelihood estimation methods within an ASR framework in cascaded manner has been developed. The results obtained clearly show that without impairing the transcription accuracy the computational load of ASR can be reduced using the cascaded speech state likelihood estimation process.

Additionally, this thesis presents my work on noise robustness of ASR using a non-negative matrix factorization (NMF) -based approach. Specifically, methods for transformation of sparse NMF-features into speech state likelihoods has been explored. The results reveal that learned transformations from NMF activations to speech state likelihoods provide better ASR transcription accuracy than dictionary label -based transformations. The results, compared to others in a noisy speech recognition -challenge show that NMF-based processing is an efficient strategy for noise robustness in ASR.

The thesis also presents my work on audio signal enhancement, specifically, on removing the detrimental effect of reverberation from music audio. In the work, a linear prediction -based dereverberation algorithm, which has originally been developed for speech signal enhancement, was applied for music. The results obtained show that the algorithm performs well in conjunction with music signals and indicate that dynamic compression of music does not impair the dereverberation performance.

Preface

This Thesis is a summary of the collection of my research papers based on the research that I have been involved in within the Audio Research Group and the Vision Research Group at the department of Signal Processing of Tampere University of Technology (TUT) during my doctoral studies. The research presented in this Thesis deals with analyzing sound and images automatically with computer algorithms.

I started the doctoral studies following my interest to understand the concepts of automatic signal manipulation and interpretation. I wanted to learn things as broadly as possible. I also wanted to understand each concept and algorithm as deeply as possible. These are the reasons why I have been willing to contribute in these different problems of signal processing, and also to teach at various courses during my years at TUT.

My work has been mostly supervised by professor Tuomas Virtanen, to whom I want to express my greatest gratitude. His thorough and clear scientific grasp to research problems will stay in my mind as a way to follow. He has been supportive and encouraging all the way through my moments of despair and disbelief, and he has also listened carefully, when I have expressed dissentive points of view.

Next I owe my gratitude to professor Joni Kämäräinen, who has been my superior for many years and who has also been significantly involved in supervising my work. He was an infinite source of new research ideas for me to try out and he had close to infinite belief in my skills. This lead to many unsuccessful research trials and rejected paper submissions during the years. However, with his relentless friendliness and faith in future success he kept me working on. Thanks to Joni, I also had a chance to have fruitful discussions and supervision from professor Jiři Matas, who has well earned my thankfulness.

I want to thank all my colleagues within the Audio Research Group and the Vision Research Group of TUT for providing me with an encouraging and pleasant environment for the work. My last, but the most important thanks belong to my dear husband, Marko Mahkonen, who has stayed beside me through all the good and the hard times working for the best of the family.

Katariina Mahkonen, Akaa, 31.1.2018

Contents

Acronyms

A/D	analog to digital
AI	artificial intelligence
AIR	acoustic impulse response
ANN	artificial neural network
ASR	automatic speech recognition
BOA	Boolean **OR** of **AND**s
BoW	bag of words
BSI	blind system identification
CD	contrastive divergence
CNN	convolutive neural network
DBN	deep belief network
DCT	discrete cosine transform
DFT	discrete Fourier transform
DNF	disjunctive normal form
DNN	deep neural network
DOA	direction of arrival
DoG	difference of Gaussian
DRC	dynamic range compression
DRR	direct-to-reverberant ratio
DTFT	discrete time Fourier transform
EDT	early decay time
EM	expectation maximization
ERR	early-to-reverberant ratio
FFPD	facial feature point detection
fps	frames per second
GMM	Gaussian mixture model
GSC	generalized sidelobe canceler
HOG	histogram of oriented gradients
HMM	hidden Markov model
ICA	independent component analysis
IFFT	inverse fast Fourier transform

ITU	International Telecommunication Union
ITU-T	ITU telecommunication standardization sector
ITU-R	ITU radiocommunications sector
k-NN	K nearest neighbors
LBP	local binary pattern
LDA	linear discriminant analysis
LLR	log-likelihood ratio
LoG	Laplacian of Gaussian
LP	linear prediction
LSTM	long short-term memory
LVCSR	large vocabulary continuous speech recognition
LVSR	large vocabulary speech recognition
MCLP	multi-channel linear prediction
MFCC	Mel-frequency cepstral coefficient
MINT	multiple input output theorem
MMI	maximal mutual information
MOS	mean opinion score
MPEG	moving pictures experts group
MUSHRA	multiple stimuli, hidden reference and anchor
MVDR	minimum variance distortionless response
NMD	non-negative matrix deconvolution
NMF	non-negative matrix factorization
NN	neural network
OLS	ordinary least-squares
PCA	principal component analysis
PDF	probability distribution function
PEAQ	perceptual evaluation of audio quality
PESQ	perceptual evaluation of speech quality
PLS	partial least-squares, projection to latent structures
PSD	power spectral density
RBF	radial basis function
RBM	restricted Bolzmann machine
RGB	red-green-blue color representation
RIR	room impulse response
RMS	root mean square
RNN	recurrent neural network
ROC	receiver operating characteristics
RT	reverberation time
SIFT	scale invariant feature transform
SDR	signal-to-distortion ratio
SNR	signal-to-noise ratio
SPL	sound pressure level
STFT	short time Fourier transform
SVD	singular value decomposition
SVM	support vector machine
TVG	time varying Gaussian
VAD	voice activity detection

List of Publications

I Hurmalainen A., Mahkonen K., Gemmeke J.F. and Virtanen T.,"Exemplar-based recognition of speech in highly variable noise", In *Proceedings of Computational Hearing in Multisource Environments (CHiME) Workshop*, 2011.

II Mahkonen K., Hurmalainen A., Virtanen T. and Gemmeke J.F., "Mapping Sparse Representation to State Likelihoods in Noise-Robust Automatic Speech Recognition", In *Proceedings of the 12th Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2011.

III Mahkonen K., Eronen A., Virtanen T., Helander E., Popa V., Leppänen J. and Curcio I.D.D., "Music dereverberation by spectral linear prediction in live recordings," In *Proceedings of the 16th International Digital Audio Effects Conference (DAFx)*, 2013.

IV Mahkonen K., Kämäräinen J-K and Virtanen T., "Lifelog Scene Change Detection Using Cascades of Audio and Video Detectors," In *Proceedings of the 12th Asian Conference on Computer Vision (ACCV)*, 2014.

V Mahkonen K., Virtanen T. and Kämäräinen J-K , "Cascade processing for speeding up sliding window sparse classification," In *Proceedings of the 24th European Signal Processing Conference (EUSIPCO)*, 2016.

VI Mahkonen K., Virtanen T. and Kämäräinen J-K , "Cascade of Boolean detector combinations," *EURASIP Journal on Image and Video Processing*, 2018(1), 61.

Author's contributions to the publications

The Publications II, III, IV, V and VI are completely written by the author, with the help of coauthors, except the part of Publication III which concerns musical beat tracking. That part is written by Antti Eronen, however, the topic is not included in this thesis. The publication I is written by Antti Hurmalainen. Tuomas Virtanen has been the supervisor for works of Publications I, II, III and V. The works for Publications IV and VI have been supervised by Tuomas Virtanen and Joni Kämäräinen together.

In the work concerning Publication II, the author was responsible for training the transformation functions from sparse features to speech state likelihoods, as well as evaluating the system performance with them. The idea of trying out the transformations came from Tuomas Virtanen. The NMF-based ASR framework into which the transformation functions were integrated was built by Antti Hurmalainen.

The best results in Publication I were achieved using the functions obtained by the author for a transformation from sparse features to speech state likelihoods, while the rest of the work for the paper was done by Antti Hurmalainen.

The work for the Publication III was a combined effort of all the authors of the paper. The fundamental idea came from Tuomas Virtanen. Elina Helander implemented the framework. The responsibility of the author was the parameter testing and evaluation. Antti Eronen ran the tests concerning musical beat tracking.

The work concerning publications IV and VI were based on Joni Kämäräinen's idea of utilizing Boolean combinations for computational efficiency. A C-language implementation by Jukka Lankinen was used for SIFT BoW feature extraction and RGB histogram extraction. The face point extractor used in Publication VI is from Zhu and Ramanan (2012).

The baseline NMF-based ASR framework utilized in Publication V was made by Antti Hurmalainen. The author is responsible for the idea, implementation and testing the cascaded version of state likelihood extraction.

All the simulations not mentioned above have been written and run by the author using computation tools of Matlab software.

1 Introduction

In this thesis I present my research in the field of signal processing. My contributions concerning audio and video signals are on different problems of the field and in the work I have used a wide range of computational methods. In this text I cover the parts of signal processing where my contributions are distributed within. Each problem and method that I have used or developed in my work is mentioned at the related part of the text, and the reference to the corresponding publication is given.

The scope of this thesis within the field of signal processing

Signal processing is a broad field of engineering falling between mathematics, physics and computer science. The most fundamental signal processing tools, many different transformations, are deeply rooted in mathematics. The physical world is the source of the signals to be processed, and the laws of physics are often part of intelligent signal processing systems to model the phenomena of the task at hand. Although signal processing existed, e.g. in the form of filtering, before the digital era, the digitalisation of signals has opened new possibilities in abundance for processing signals by means of computational technology.

The work leading to the collection of publications in this thesis belong to signal processing sub-fields of signal enhancement and signal information retrieval. The task of signal enhancement aims at improving the captured signal in some way, possibly reducing noise and/or accentuating some qualities of the signal, for example improving speech intelligibility of audio signal or brightening colors and sharpening edges in images. Among signal enhancement tasks, signal restoration particularly aims at restoring a deteriorated or noisy signal as close as possible to its genuine form. In this thesis, I present my work of Publication III on the signal restoration task of dereverberation, i.e. removing the detrimental effect of excess reverberation from an audio signal. Audio dereverberation is mathematically well defined, but very challenging in practice and there is a multitude of different approaches proposed in the literature. However, most of them assume and are examined with a speech signal. Algorithms for music signal dereverberation are rare and thus in the Publication III, dereverberation task for music signals is addressed.

Signal information retrieval is a vast field of research, where the core of many algorithms deal with classification problems. In a classification task the goal is to give one of predefined labels to the signal at hand, that is, to find out into which 'class' the phenomenon that the signal represents belongs to. Problem of classification in its most basic form is binary classification, which may be also seen as detection where the two predefined labels are *'the sample represents the quested phenomenon'* and *'the sample represents something else'*. Due to binary classification being a fundamental task within many frameworks of more complex signal information retrieval systems, it is crucial to be able to perform this task accurately, yet with small computational load. In my research of Publications IV and VI I have focused on developing a binary classification framework with these capabilities by combining multiple different binary classifiers in computationally efficient way. The framework has been evaluated with video cut point detection and video classification tasks. These tasks allow examination and combination of classifiers based on multiple signal modalities, namely audio and visual modalities, of videos, which is highly important if all the aspects of signal are to be utilized.

Automatic speech recognition task, considered in Publications II, I and V is a highly complex task of signal information retrieval, providing high level information from the signal. For a long time, the poor robustness of ASR -algorithms against non-stationary noise within the signal has been a severe issue degrading the performance of algorithms drastically. Only lately, thanks to neural networks (NN), particularly deep neural networks (DNN), the noise robustness of ASR has reached a level adequate for most applications. In the Publications I and II an NMF-based approach has been examined for providing noise robustness for an ASR framework.

The requirement of accurate and computationally efficient processing naturally applies also the ASR task. Due to NMF-processing being accurate, but computationally heavy method for ASR, in Publication V I have addressed the computational efficiency of the approach. Within most ASR frameworks, also the NMF-based framework used in Publications I and II, phoneme-class probabilities are estimated as an intermediate signal representation. This processing step within ASR frameworks is very close to classification. Thus, within the Publication V I have implemented a framework for estimating the phoneme-class probabilities by combining the NMF-approach with other approaches in computationally efficient way.

Research questions

As already revealed, several distinct research questions concerning signal processing algorithms on audio and video signals have been examined within this thesis.

In Publication III, concerning audio dereverberation task, an algorithm proposed in Furuya and Kataoka (2007) for speech signal dereverberation was evaluated with music signals. An additional question posed was, whether dynamic compression of audio would deteriorate the dereverberation performance.

In Publication IV, concerning the task of video cut point detection, Boolean functions for combining multiple video cut detectors were evaluated. The motive behind selecting Boolean functions was that Boolean functions naturally lend themselves for sequential deduction. That is, the idea was to find out, whether Boolean functions can successfully be used as a combination function and whether evaluating the Boolean functions sequentially would bring computational savings.

In Publication VI a framework for combining multiple classifiers in a computationally efficient way has been developed. The framework has been evaluated with video cut detection and video classification tasks. The system has been evaluated in terms of the classification accuracy and the computational load of classification. Additionally, a training algorithm proposed for the framework has been evaluated in respect to other suitable algorithms.

In Publication I, a NMF-based framework has been evaluated for noisy ASR task. The framework was evaluated in respect to other methodologies in the The PASCAL CHiME Speech Separation and Recognition Challenge (Barker et al. (2013)).

In Publication II, applying the NMF-based framework for noisy speech recognition task, different transform functions for converting the NMF-based sparse feature vectors into hidden Markov model (HMM) state likelihoods has been examined. Specifically, data induced ordinary least squares regression (OLS) and partial least squares regression (PLS) based transforms have been evaluated in respect to a dictionary label based transform.

In Publication V, cascade processing principle has been implemented for the noisy ASR task. It has been evaluated, whether the computationally heavy NMF framework can be utilized in a time sparse way to reduce its computational load, such that the ASR transcription does not deteriorate. Moreover, it has been evaluated whether a simple neural network would provide additional information for increasing the transcription accuracy, and would its use reduce the computational load further.

Outline of the thesis

In Chapter 1 of this thesis I introduce the field of signal processing and clarify the position of the research problems of my publications within this field. In Chapter 2 I discuss the essence of the audio and video signal. I explain the computational methods of measurement analysis, which are broadly utilized within frameworks of automatic audio and video signal processing. Methods of analysis specifically used in my publications are highlighted. Then, topics of my publications, namely dereverberation, computationally efficient binary classification and automatic speech recognition, are discussed in Chapters 3, 4, 5 and 6 in the order based on the hierarchical level of the information they deal with.

In Chapter 3 I discuss audio dereverberation. Signal enhancement, where the task of audio dereverberation belongs to, deals with signal level information about the processed audio, and is often used for front-end processing of signal before information retrieval tasks. In this chapter I explain the concepts related to reverberation and give an overview of the signal processing methods generally utilized for the dereverberation task. Then I explicate the experimental arrangement and the results obtained in Publication III.

Signal information retrieval is an area of signal processing which seeks for human cognition defined information about signal contents. The task of computationally efficient classification investigated in Publications IV and VI, and the task of automatic speech recognition examined in Publications I, II and V are this kind of tasks.

In Chapters 4 and 5 I talk about classification, which is one of the most important tasks of machine learning. In Chapter 4 I present the main processing steps and mathematical models generally used for classifying signal samples into different human defined categories. I also address methods for incorporating multiple models or classification functions within a single framework for increased classification accuracy. Finally I present the detector compound framework proposed in Publication IV and show the results obtained in video cut point detection.

In Chapter 5 I address the aspect of computational efficiency by using a sequential classification strategy. Specifically, I highlight how a cascade processing principle as a sequential evaluation principle is utilized in the literature and in the work of Publication VI. I present an algorithm, proposed in Publication VI, which has been developed for learning a cascade of Boolean combinations and present the results obtained in the task of video classification.

In Chapter 6 I address automatic speech recognition, which is the most complex signal information retrieval task studied within my work. I explain the processing steps traditionally utilized within the computational methods for the task, and I discuss about different approaches of acoustic modeling often used within the frameworks. I explain the sequential decision making strategy applied to acoustic modeling for ASR, used in Publication V. Then I report the work and results of Publications II, I and V.

Finally, in Chapter 7 I discuss shortly about the findings of each Publication, conclude the lessons learned and outline the possible future directions.

2 Audio and video signals

In this chapter I introduce characteristics of digital audio and video signals, as well as some low level signal analysis methods, which are broadly utilized for audio and image analysis.

2.1 Sound scape – audio signal

In physical terms, the sound scape consists of air pressure waves, which have been initiated by different sound sources (Rossing (1990)). Generally there are multiple sound sources present at the same time in a sound scape of any natural environment. E.g. in an urban environment there might be people speaking and playing music, noisy traffic, clatter noise of construction work etc. . To hear a sound from a single sound source without distractions of other sounds or echos is possible only in an acoustically isolated anechoic room.

By a term audio signal, I mean a captured sound wave. To capture the sound waves in the air, one or multiple microphones are used. A microphone converts the sound pressure waves into electric voltage changes (Rossing (1990)). To save this analog signal in a digital format, an A/D conversion is done (Proakis and Manolakis (1996)). The voltage level from the microphone is measured at regular interval $\Delta\tau$ and the acquired *samples* are saved. The higher the sampling frequency $f_S = 1/\Delta\tau$, the more details of the original waveform are captured. The audio signal, consisting of the stream of samples, may be either processed on-line in real time, or saved into a file for later usage. For research purposes, mostly lossless audio file types, instead of lossy audio file types, are used to avoid compression artifacts.

2.2 Moving picture – video signal

Video is a media that combines a sequence of images to form a moving picture. Usually a video recording additionally contains an audio stream with one (mono) or two (stereo) channels corresponding to the sequence of images. Video is thus a form of multimedia.

In digital videos, the images are represented as rectangular matrices of *pixels*, points of distinct color. The number of pixels in the matrix defines the amount of details the picture can represent - the more pixels the more details may be expressed. Ultra high definition 8K videos contain pixels in the order of 8000 horizontally and 4000 vertically, while the contemporary standard television screen resolution is 1920 x 1080 pixels. Videos of 160 x 120 pixels are among the lowest resolution videos shoot nowadays.

The color value for each pixel of a color image is typically represented as a vector of three numbers (Gonzalez and Woods (2008)). For RGB-images they represent intensities of red (R), green (G) and blue (B) color. For YCbCr-images they represent a magnitude for brightness, that is luminance (Y), and two values for color chrominance (Cb and Cr), which combine information of color hue and saturation. The chrominance values represent the color in axes from yellow to blue (Cb) and from green to red (Cr). Human eye is more sensitive to luminance than chrominance, and thus YCbCr representation enables higher quality images for the same bit rate.

In addition to the number and representation of pixels in each video frame the time lag used between displaying consecutive video frames, called a *frame rate*, has an effect on the visual appearance of the video. Consecutive images shown slower than 10 images per second, are perceived as individual images by most people according to Read and Meyer (2000). To achieve smooth appearance for object movement in video, Thomas Edison recommended a frame rate of 46 frames per second (Brownlow (1980)). However, the standard frame rates considered adequate in TV and cinema business are 24, 25 and 30 fps. Video games are sometimes designed for showing images 60 fps rather than using the cinema standard.

With respect to video signal, the storage capacity becomes an issue very quickly, if high resolution is needed or long video sequences are to be filed. The International Telecommunications Union (ITU) Telecommunications Standardization Sector (ITU-T) Moving Picture Experts Group (MPEG) has defined a widely used video compression, i.e. coding, standard H.264 (ITU-T Recommendation H.264 (2017)), which is also known as MPEG-4 part 10. Other contemporary video coding specifications are e.g. HEVC (ITU-T Recommendation H.265 (2018)), Theora (Foundation (2017)), VP9 (Grange et al. (2016)) and AV1 (de Rivaz and Haughton (2018)). Video coding is a multistage process where many different aspects of the image sequence are utilized to avoid saving redundant bits (Ghanbari (2003)). It is usually done in blocks, that is, a video frame is sliced both vertically and horizontally into equal sized blocks, and each partition forms an entity for coding. However, similarities between adjacent blocks of the frame as well as blocks in previous frames of the video are utilized to avoid redundancy in bits to be stored. Different strategies, e.g. color or texture similarity, may be used for estimating the contents of one block based on near by blocks for intra frame estimation. Motion estimation in terms of motion vectors is used to estimate the position of a moving object within the next frame for inter frame estimation. The difference between the actual contents of the block and the estimate obtained based on the intra and inter frame correlation estimation is computed. The remaining residual signal is further decorrelated and finally encoded into numbers to be quantized and stored as bits. When reconstructing the image from the encoded format, in addition to the decoding process consisting of inverse operations of the encoding process, deblocking filtering is performed to lessen the compression artifacts on block boundaries.

To store a video, not only the compressed image frames need to be saved, but the video file has to contain also the accompanying audio signal and the synchronization information for the two. The video file types are called video *containers* or *wrappers* (Beach (2010)). The container defines the file header and side information structures, and allows the actual data of compressed video frames and audio stream to be included in certain partitions of the file. The container may restrict the set of possible encoding formats for the image frames and audio stream. The variety of existing container formats, i.e. video file types, (and their file extensions) include at least ASF (.asf .wma .wmv), AVI (.avi), MOV (.mov), MPEG-4 part 14 (.mp4), FLV (.flv), RealMedia (.rm), and Matroska (.mkv .mk3d .mka .mks).

2.3 Audio signal representations

As already discussed, audio signal consists of samples of measured instantaneous sound pressure values. This stream of samples is the digital *time-domain* representation of the captured sound. When dealing with sampled sound waves rather than a continuous signal the details of the sound pressure wave falling between consecutive samples are lost. According to Nyquist-Shannon sampling theorem, the idea of which was initiated in Nyquist (1928), the frequency content of audio signal, which is above half the sampling frequency f_S may not be distinguished from the frequency components $0...f_S/2$. That is, the contents of sound frequencies above $f_S/2$ will be *aliased* over the contents at frequencies $0...f_S/2$. Fortunately, the real life sounds tend to have their main frequency content at low frequencies, and the power of sound wave components usually suppress significantly towards high frequencies. Also the microphones have a finite frequency response, naturally muting the high frequencies.

The above mentioned fact that sound pressure waves are initiated by vibrating sound sources leads to that sound is well represented also in *frequency-domain*, which will be introduced below. Another fact mentioned above, that different sounds are invariably mixed together in real life recordings, is the major problem in tasks of audio information retrieval. Thus the mathematical basics of how sounds intertwine together within a recorded signal are also considered below. Finally, the characteristics of human hearing and respective audio signal analysis methods, which are popularly utilized for audio signal analysis are considered. The presented methods have been found highly successful in many audio information retrieval tasks, and they have been utilized also in my publications.

Time-Frequency representation of audio signal

Since a sound wave is a consequence of vibrations of sound sources, it is reasonable to represent the audio signal in terms of the wave frequencies that it contains. However, the frequency content of the sound usually changes all the time, thus a *time-frequency* domain representation of the audio signal is widely used in audio signal processing. It is implemented in most audio signal processing frameworks by cutting the signal into clips, i.e. *frames*, of equal length and estimating the frequency contents of each frame $\boldsymbol{x}_n$ using the Discrete Fourier transform (DFT) (Oppenheim et al. (1999)), which in this setting may be called also as short-time Fourier transform (STFT). The DFT coefficients $X(k)$ for frequency bins $k=0...\lfloor (T_w-1)/2 \rfloor$, T_w being the number of samples within the audio frame, are given by

$$X(k) = \sum_{t=0}^{T_w-1} x(t)\, e^{-\mathrm{i}2\pi kt/T_w}. \tag{2.1}$$

The complex valued DFT coefficient $X(k)$ indicates the magnitude and the phase of the sound wave at frequency kF_S/T_w.

For smoother time-frequency domain representation, the consecutive signal frames are usually taken with overlap, and the frequency analysis is focused into the middle part of the frame using a window function. A popularly utilized window function in audio signal processing is e.g. the Hann window (Harris (1978)) which weighs the samples of the frame as

$$w(t) = 0.5\left(1 - \cos\left(\frac{2\pi t}{T_w - 1}\right)\right), t = 0...T_w - 1. \tag{2.2}$$

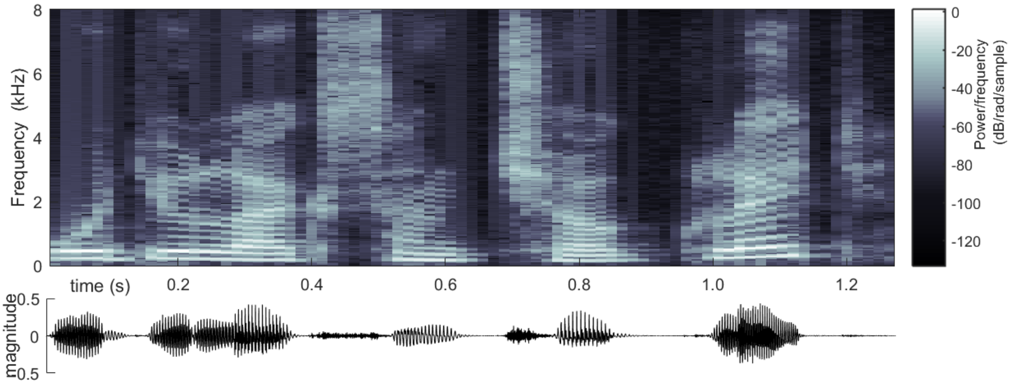


Figure 2.1: Audio signal (below), with a spoken sentence: "Would you like some chocolate?", and (above) a spectrogram representation of it as frame-wise STFT magnitudes.

Suitable length of the audio frame and the time shift between consecutive overlapping frames depends on application. They are set considering the requirements of time resolution, frequency resolution and the latency limits of an application. For automatic speech recognition, a coarse frequency information has been shown to give enough information, while the time accuracy has to be good as the pronunciation of the shortest phonemes last for only few milliseconds, and an average syllabic rate of speech is 4 Hz (Arnfield et al. (1995)). Thus, for ASR, the frame length 10-30 ms and frame shift 5-10 ms are generally used. In the ASR framework of publications II, I an V a frame length 25 ms and shift of 5 ms are used. In music, the important aspects are the rhythm, melody and harmonic progression, where shortest harmonic unities last about 100 ms. Thus for detailed frequency content analysis of music, longer frames up to even 200 ms may be used. In audio dereverberation task of Publication III we found out, that using the frame length 50 ms and shift of 25 ms gave the best results.

The time-frequency representation of audio signal, also called a *spectrogram* and illustrated in Figure 2.1, results in a trade-off between the time and frequency resolution of the representation. The longer signal frame, the more frequencies for the representation may be estimated, but since the values are averages over the frame, the resolution in time dimension becomes smoothed. On the other hand, the shorter the time frame, the resolution in time dimension is well preserved, but approximation of frequency parameter values becomes coarser.

Characteristics of combination sound from multiple sources

The air pressure waves of sounds from multiple sources, which build up the recorded sound wave, intertwine together as a sum of the individual waves. The recorded audio signal is thus

$$x(t) = s_1(t) + s_2(t) + s_3(t) + ..., \tag{2.3}$$

where s_i is the wave component from one sound source. In terms of the frequency representation of the sound combination and the individual sources

$$X(k) = S_1(k) + S_2(k) + S_3(k) + ..., \tag{2.4}$$

the complex valued sound components $S_i(k)$ from different sources may either reinforce or suppress each other at a certain frequency bin k of the combination. This depends on the phases of the individual components $S_i(k)$ at this frequency.

From audio application point of view, this enables sound canceling by generating audio wave in opposite phase as the original sound. On the other hand, the audio signal processing task of sound source separation can be seen to be very challenging.

Human hearing based audio analysis

Characteristics of human hearing are often incorporated into audio signal analysis. It has been shown to be advantageous for many signal processing tasks on audio signal, the most prominent examples being audio coding and automatic speech recognition. For efficient audio coding all the details of the sound wave that are not perceived by human, may be discarded without causing any noticeable degradation on the sound. For audio interpretation tasks, including automatic speech recognition, taking into account the non-even sensitivity of human ear to different frequencies (Zwicker (1961)) has been found out to be crucial for the algorithms O'Shaughnessy (2000). The cochlea of human ear reacts to sound in terms of the frequency components that the sound contains. The normal frequency range of hearing for human is 20 Hz - 20 kHz, and the sensitivity of ear to distinguish different frequencies within this range is not even. The frequency resolution capability of ear has been found out to be more specific for low audio frequencies than for high audio frequencies. That is, human ear tends to integrate sound frequencies very close to each other. This phenomenon is called frequency masking. Concerning the low frequency sounds, the frequency masking causes the frequencies only very close to each other to be integrated. Regarding the high frequency components, the frequency masking phenomenon integrates the perception over broad frequency range. Human ear performs also time-domain masking, and masking effect is different depending on the overall sound contents (Zwicker and Fastl (1999)), but those aspects are generally not considered for audio information retrieval tasks.

Reflecting the nonlinear frequency resolution of human hearing, a Mel-frequency scale

$$\mathrm{MEL} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.5}$$

has been introduced in Stevens et al. (1937). For audio signal analysis to approximate the frequency masking of ear, a fixed set of band pass filters based on the Mel frequency scale are generally utilized. Traditionally the Mel-filter bank analysis is implemented on the signal discrete time Fourier transform (DTFT) magnitude spectrum using triangular basis functions, i.e. filters, which perform the spectral integration according to Mel scale. The spectral energy $E(b)$ of frequency band b is obtained using a triangular band-filter $\mathbf{f}_b$ as

$$E(b) = \sum_{k=0}^{K-1} |X(k)| \cdot \mathbf{f}_b(k), \tag{2.6}$$

where $X(k)$ is the DTFT coefficient by (2.1). This kind of DTFT-based Mel scale magnitude spectrograms have been used as audio *features* for automatic speech recognition task in Publications II,I, III, and V.

For audio information retrieval tasks, immensely utilized Mel-filter bank energy based features are the Mel-frequency Cepstral Coefficients (MFCC), which have been found out to be highly useful already in Davis and Mermelstein (1980). To compute the MFCC features, the above presented Mel-scale energy spectrum $E(b), b = 1, 2, ..., B$ is further transformed using the discrete cosine transform (DCT). The DCT transform provides decorrelation of features, uncorrelatedness being a desirable property of a feature for many algorithms. Often, to incorporate information about the context of an audio frame, the amount of change in each MFCC coefficient among a few consecutive audio frames are considered as additional features called ΔMFCC (deltaMFCC). I have used MFCC and ΔMFCC features for the experiments in Publications IV, V, and VI.

2.4 Computational image analysis

Variability of presentation within an image or a video frame is huge. That is, multiple objects may be located anywhere in the image, in any size and with any pose or orientation. Objects are often partially occluded by other objects. Depending on lighting conditions, the color hues captured by the camera vary tremendously. Human brain is capable of dealing with the complexity of this visual information without a problem, but for a computer, image analysis is a highly demanding task. In the following I explain some computational procedures for automatically analyzing the contents of a digital image.

2.4.1 Filters for recognizing simple shapes

For automatic image analysis, the standard approach is to first search for, i.e. *detect*, simple details and patterns from an image, e.g. lines, edges, corners, blobs etc. . Each subimage at every possible position, size and orientation from the whole image is evaluated separately for the quested shapes. Simple basis functions, i.e. filters, are designed based on the visual appearance of the quested shapes. They are specifically suited for the preliminary edge detection, corner detection, blob detection, etc. from image patches. They are in the core of nearly every higher level feature extraction scheme for computer vision. A linear filter response within image $\mathbf{I}$ at pixel position (h, v) is computed as

$$x(h,v) = \sum_{i=1}^{N_h} \sum_{j=1}^{N_v} \mathbf{I}\left(h - \tfrac{N_h+1}{2} + i, v - \tfrac{N_v+1}{2} + j \right) \cdot \mathbf{F}(i,j), \tag{2.7}$$

where $\mathbf{F}$ denotes the 2-dimensional shape detection filter of size $N_h \times N_v$. The filter dimensions N_h and N_v are assumed to be odd, e.g. 3x3, 5x5 etc. .

For detecting edges from an image, simple linear filters approximating the image derivative are Prewitt, Sobel, and Roberts filters. However, these filters are usually not enough for reliable interest point detection and some post processing is necessary for noise robustness of detection. For example, an old, nevertheless the state-of-the-art Canny edge detector by Canny (1986) is a dominant approach. It uses first two linear filters to extract the horizontal and a vertical components of gradient at each position of the image. These gradient images are combined to provide a likelihood of an edge existing at each pixel position as well the direction of the potential edge. The edge likelihood values are post processed first by non maximum suppression, which damps majority of them to be zeros. Then thresholding with hysteresis is used to provide crisp edges form the remaining line likelihood values.

For blob detection, a linear Laplacian of Gaussian (LoG) filter is a standard choice. The difference of Gaussians (DoG), computed as a difference between responses to Gaussian filters with different scales, is also used as an approximation of LoG. To detect the blobs, maxima and minima of scale normalized LoG or DoG responses are searched.

Haar-like filters developed by Viola and Jones (2001a) are very popular due to their computationally efficient applicability and versatility to detect different patterns. The filters consist only values -1 and 1, which are arranged as rectangular areas. An efficient implementation of Haar-like feature computation by Viola and Jones (2001b) utilizes a sum-up table called *integral image*.

An example of a non-linear simple pattern detecting filter is a local binary pattern (LBP) filter proposed by Ojala et al. (2002). When computing an LBP response for a pixel, values of pixels within the neighbourhood of the central pixel are compared to each other to obtain directional information about local intensity differences. The pixel value differences are converted to binary values based on the sign of the difference – thus the name binary pattern – giving one pattern for each pixel which is used as a neighbourhood centrum.

2.4.2 Using histograms to compact information

When analyzing the image with many different filters, the amount of numbers characterizing the image becomes even bigger than the number of pixels. This data explosion must obviously be suppressed and only relevant information should be retained. One simple and popular solution for reducing the amount of data is to collect a *histogram* of attributes to describe the whole image or some part of it. The idea is extensively utilized also in text processing, and thus a name Bag-of-Words (BoW) (Harris (1954)) is also used. A histogram or BoW is a collection of counts of selected attributes from an image. Each number in the histogram refers to occurrences of a certain attribute within the image. One attribute may account for e.g. strong enough responses from a certain type of filter, where an occurrence is determined using a threshold value.

The simplest of the histogram-based image feature vectors, also used in Publications IV and VI, is an RGB -color histogram. A color histogram feature (Novak and Shafer (1992)) is popular in analyzing video frames particularly because of its fractional computational load. Histogram of oriented gradients (HOG) (McConnell (1986)) is another very popular histogram based feature. To collect HOG, multiple linear filters which detect gradual changes of luminance values in many scales and orientations within the image are used, and notable responses of each filter type of gradient are accumulated into the HOG vector. The LBP presentation (Ojala et al. (1994)) is eventually a histogram based feature. After the initial nonlinear filtering stage, a histogram of binary pattern representations is collected from an image patch to form an LBP feature vector.

In the work of Publications IV and VI on video analysis I have utilized histograms of scale invarian feature transform (SIFT) (Lowe (2004)) features, which are explained in the next section. A large set of SIFT feature vectors, i.e. SIFT descriptors, are first computed from training material. Then the K-means algorithm is applied to this set of SIFT descriptors and the K mean SIFT-descriptors are collected into the codebook. When analyzing a new video, the SIFT descriptors are extracted from each video frame and they are compared to those of the codebook. Counts of descriptors that match the codebook items closely enough are returned as a SIFT BoW feature vector for each video frame.

2.4.3 Crafted features for image analysis

There are many hand-crafted image feature extraction algorithms, which are designed to tackle some specific problems of computer vision. The scale-invariant feature transform -descriptors, which are utilized in Publications IV and VI, are an example of a crafted analysis method which detects simple shapes and patterns invariantly to changes of scale, orientation and illumination. Another example of a crafted image feature computation scheme discussed here is facial point extraction. A facial point extraction algorithm by Zhu and Ramanan (2012) is utilized in Publication VI.

The scale invariant feature transform is an image analysis scheme used successfully ever since its invention by Lowe (2004). SIFT analysis gives localized information for the image by returning feature vectors, called *descriptors*, for the image to be analyzed. Each descriptor is associated with a specific location, a *key point* of interest, in the image. Detection of key points within an image starts with computing different smoothed versions of the image using Gaussian filters of different sizes, i.e. scales. Between each pair of adjacent scales, a difference of these smoothed images are used to produce DoG responses. Local minima and maxima of scale normalized DoG responses in the three dimensions; width, height and scale, are detected to be potential key points. The final set of key points is selected from those by excluding the poor ones in the areas of low contrast and on edge lines. For each key point, a descriptor is constructed. It consists of histograms of 8 orientations from 16 small image patches within the neighborhood of the key point, making a descriptor length to be 128.

Facial feature point detection (FFPD) is an example of crafted very high level feature extraction scheme. There are many different FFPD algorithms specifically designed for tracking and analyzing faces in images. FFPD algorithm finds image coordinates corresponding to a set of points of a human face in an image using some lower level features, e.g. SIFT-descriptors. This kind of high level feature extraction schemes are highly complex and utilize many filters, transforms, and heuristically derived selection schemes to achieve their output. The face point detector of Zhu and Ramanan (2012), which is used in Publication VI, utilizes HOG descriptors to find potential interest points and models different facial poses with trees.

2.4.4 Feature selection

The huge pool of different low level analysis methods, i.e. features, available for image analysis calls for methods to select the best features for an application at hand. Using all the available features is not feasible due to the *curse of dimensionality*, discussed first by Bellman (1957). The term is used to describe the difficulty of dealing with very high dimensional data. Thus to select the best performing features or filters from the huge pool of them available, *feature selection* must be performed. The feature selection methods are generally divided into three categories of methods, namely, filter, wrapper and embedded methods (Kumar and Minz (2014)).

The filter methods are the simplest feature selection methods, mainly used only as a preprocessing step to exclude the distinctly useless features (Kumar and Minz (2014)). The filter methods evaluate each feature individually, not taking correlations between different features into account and thus failing in reducing redundancy in the remaining feature set. The "filter" in this case is some particular way of computing a statistical score for the usefulness of each potential feature to describe the data at hand. Examples are Chi squared test, information gain and correlation coefficient score. The features with highest scores are then selected for use.

The idea in wrapper type feature selection methods is to try out different feature sets by training the system with each subset of features, and then selecting the best performing set for use. The drawback of these methods is high computational cost. Often greedy solutions are proposed, as in total there are an unsustainable number $2^N - 1$ of possible solutions for selecting a subset of features out of N available, N being a big number. The greedy solutions include forward selection and backward elimination type algorithms.

Embedded feature selection methods are embedded into the analysis algorithm design, such that all the features are given for the system training and the algorithm itself makes decisions about using a feature or not. Regularization algorithms like LASSO (Tibshirani (1996)) and Tikhonov regularization (Tikhonov et al. (1995)), which is often called ridge regression, are considered as embedded feature selection methods. Decision tree building algorithms are also sometimes considered embedded feature selection methods, although the feature selection process within the tree construction resembles the process of wrapper methods.

Many of the object detection cascades discussed in Section 5.2 utilize the AdaBoost algorithm (Freund and Schapire (1997a)) for selecting the most suitable Haar-filters to be used for features form the huge pool of Haar wavelets available. The process of some object detection cascades may be considered as an embedded procedure to concurrently select the features and build the detector with the AdaBoost algorithm. On the other hand, some object detection cascades perform first greedy forward selection wrapper type feature selection process with AdaBoost, and then fine tune the cascade classifier function by some another method for training.

2.5 Features from learned linear transformations

In addition to the analysis methods specific to audio and image data, there are lots of general purpose methods, which are commonly utilized for data of audio, image or any other measurement modality. Among feature extraction methods, many different types of linear transformations are extensively utilized for preliminary signal analysis. Linear transformation of vector $\boldsymbol{x}$, which is the raw signal or some other feature representation of it, is performed as

$$\boldsymbol{x}_{\text{new}} = \mathbf{B}\,\boldsymbol{x} \tag{2.8}$$

where $\mathbf{B}$ is the transformation matrix, and $\boldsymbol{x}_{\text{new}}$ is the new vector representation. In addition to transformations like DFT and the many linear image analysis filters, where the transformation matrix $\mathbf{B}$ is predetermined, the basis for the transformation may be learned from training data. The transformation may be either underdetermined or overdetermined to induce the new representation $\boldsymbol{x}_{\text{new}}$ to have either less or more elements than the original representation $\boldsymbol{x}$. An underdetermined basis, which have fewer basis vectors than there are measurements in a set to be analyzed, are used to detect the salient properties of data and reducing its dimensionality. On the other hand, overdetermined basis which consist of a big pool linearly dependent basis vectors are used for applications based on *sparse coding*.

For example the simple machine learning approach, K-means algorithm (Hartigan (1975)), is often used for learning a basis $\mathbf{B}$ for a linear transform (2.8). This approach has been used for learning the transformation matrix for the SIFT -histogram features in Publication VI.

Algorithms like principal component analysis (PCA) and independent component analysis (ICA) produce linear transformations for distinguishing the most salient components of data. They find a basis $\mathbf{B}$ which better reveals the variability specific to the data at hand. These methods are also often used for *dimensionality reduction*, for constructing an underdetermined transformation basis. Dimensionality reduction, similarly to feature selection, is done for discarding redundancy and noise-like information of the data. Dimensionality reduction is achieved such that the basis vectors in $\mathbf{B}$ are ordered in the diminishing order of data variance. The basis vectors corresponding to directions of smallest data variance are discarded to obtain $\mathbf{B}_{\text{cropped}}$ for the transformation (2.8). The discrepancy of information in data $\mathbf{x}$ representation according to the corresponding new feature vector $\boldsymbol{x}_{\text{cropped}}$ is left as modeling error $\mathbf{e}$ as

$$\boldsymbol{x} = \mathbf{B}^{+}_{\text{cropped}} \cdot \boldsymbol{x}_{\text{new,cropped}} + \mathbf{e}, \tag{2.9}$$

where $\mathbf{B}^{+}_{\text{cropped}}$ is the Moore-Penrose inverse (Atkinson (1979)) of the reduced basis and $\boldsymbol{x}_{\text{new,cropped}}$ is the correspondingly shorter feature vector. In Publication VI I have utilized PCA for reducing the length of a face point feature vector.

When using linear transformations for sparse coding, the equation (2.8) is utilized the other way round, as

$$\boldsymbol{x} = \mathbf{D}\,\boldsymbol{x}_{\text{new}}, \tag{2.10}$$

where $\mathbf{D}$ is an overdetermined feature dictionary. The new feature vector $\boldsymbol{x}_{\text{new}}$ becomes higher dimensional than the original data $\boldsymbol{x}$, but the clue in the sparse coding approach is that the feature vectors $\boldsymbol{x}_{\text{new}}$ are enforced sparse, i.e. to have many of its values zero.

One of the first algorithms for finding an overdetermined dictionary $\mathbf{D}$ which fits together with the sparsity constraint on feature vectors $\boldsymbol{x}_{\text{new}}$ is singular value decomposition (SVD) based K-SVD -algorithm (Aharon et al. (2006)). The K-SVD algorithm finds components, i.e. *atoms*, into a dictionary via a process similar to the K-means algorithm. At each iteration of the algorithm, the dictionary $\mathbf{D}$ and the new data representation $\boldsymbol{x}_{\text{new}}$ are updated for smaller reconstruction error $\mathbf{e} = \boldsymbol{x} - \mathbf{D}\,\boldsymbol{x}_{\text{new}}$ using SVD such that maximally K elements within each new data representation vector $\boldsymbol{x}_{\text{new}}$ are nonzero.

Sometimes, a non-negativity constraint is set for the dictionary $\mathbf{D}$ and the new feature vectors $\boldsymbol{x}_{\text{new}}$. This is a heuristic constraint, which is justified if the measurement data is non-negative by nature. E.g. for audio spectral magnitudes this is the case, as the sound may be quiet or loud, but the sound energy can not be negative. NMF-algorithms perform decomposition $\mathbf{X} \approx \mathbf{D}\,\mathbf{X}_{\text{new}}$ of original data matrix $\mathbf{X}$ into two non-negative matrices $\mathbf{D}$ and $\mathbf{X}_{\text{new}}$. There are many NMF -algorithms, setting the non-negativity constraint on $\mathbf{D}$, $\boldsymbol{x}_{\text{new}}$ or both of them. The non-negativity constraint may be set as well on the reconstruction error $\mathbf{E}$ for the optimization of $\mathbf{X} = \mathbf{D}\mathbf{X}_{\text{new}} + \mathbf{E}$. The algorithms perform in iterative way using different constraints and different measures on the reconstruction error. A selection of different NMF-algorithms is presented in Lee and Seung (2001).

An NMF approach has been used in the studies of Publications II and I on automatic speech recognition. However, in frameworks of Publications II, I and V, the non-negative dictionaries have been obtained by sampling from training data rather than using an optimization algorithm.

Algorithms to achieve sparse feature vectors $\boldsymbol{x}_{\text{new}}$ when having the fixed overdetermined dictionary $\mathbf{D}$ aim minimizing a cost function

$$\mathcal{C} = \sum_{\boldsymbol{x} \in \mathbf{X}} ||\boldsymbol{x} - \mathbf{D}\boldsymbol{x}_{\text{new}}||_2^2 + \lambda_1 ||\boldsymbol{x}_{\text{new}}||_1 + \lambda_2 ||\boldsymbol{x}_{\text{new}}||_2^2, \tag{2.11}$$

where $\mathbf{X}$ is the training dataset and $||\mathrm{v}||_p$ denotes the ℓ^p-norm of vector v. For the LASSO algorithm (Tibshirani (1996)) $\lambda_2 = 0$, for the Ridge regression algorithm (Tikhonov et al.

(1995)) $\lambda_1 = 0$ while the elastic net regression (Zou and Hastie (2005)) utilizes both the constraints with $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$.

2.6 Features using neural networks

Recently the deep neural networks (LeCun et al. (2015)) have been used for nearly every task of machine learning, the feature extraction making no exception. Once the problems in learning DNN weights had been solved, the DNNs have turned out to be very efficient non-linear feature extraction functions.

Deep belief net (DBN) is a type of DNN, used for learning feature extraction functions, as proposed by Hinton and Salakhutdinov (2006). An efficient algorithm for training DBN weights is presented by Hinton et al. (2006). DBM is built by stacking multiple one-hidden-layer restricted Boltzmann machines (RBM) (Salakhutdinov et al. (2007)). Each RBM-layer is trained generatively as a stochastic network, where an energy function defined by network weights is associated with each training data vector. The RBM weights are to be set such that in terms of the training data the energy function is minimized and thus the likelihood of the data is maximized. An efficient learning algorithm for obtaining RBM weights is the contrastive divergence (CD) -algorithm developed by Hinton (2002). To train multiple RBMs for a DBM, the outputs of the previous layer RBM, used in deterministic rather than stochastic way, are used as training data for the next layer RBM. The trained DBM feature extractor is eventually utilized deterministically as a feed-forward type neural net. It has been found out, that by adding new layers to the DBN network, increasingly higher level features can be obtained (Hinton (2014)).

Autoencoder is a special type of DNN used to obtain so called bottle-neck features Goodfellow et al. (2016). Autoencoders are effectively feed-forward nets, which may be trained with any back-propagation type algorithm. An autoencoder consists of multiple feed-forward type layers of neurons, the middle layer being as small as the new feature vector is desired to be. This small layer forms a bottle-neck for the information. For the small bottle-neck layer, an autoencoder is thus enforced to learn a small dimensional representation of the input, the short representation discarding as little as possible of the information present in the original vector. The vectors obtained from the bottle-neck layer of an autoencoder thus provide the new features.

Convolutive neural networks (CNN) are feed forward networks, which are trained discriminatively for classification from the beginning, but the network structure is specifically designed from the feature extraction point of view. Layers of a CNN are restricted in comparison to general feed forward network structure such that not every input of each layer is connected to every neuron of the layer. It is the matter of choice for the algorithm developer to decide about which inputs are connected to which neurons. A CNN consists of multiple pairs of layers providing higher and higher level representations of the input in a position invariant way. The neurons of the first layer of each pair compose the basis for the convolutive property of the net. Thus it is called a *convolutive layer*. The convolutive layer contains plenty of similar neurons sharing the same connection weights, but being connected to different sets of the layer inputs. The second layer of each pair, called a *pooling layer*, has restricted connectivity as well. The neurons of this layer are designed to provide the property of the position invariance for the net. Each neuron is connected to those convolutive layer outputs which share the same weights, i.e. implement a certain type of filter. Usually a max-pooling strategy is used such that the neuron outputs the largest value among its inputs. CNNs are utilized in computer vision with huge popularity and success at least since the publication of Krizhevsky et al. (2012). Lately they have been successfully tuned also for audio analysis e.g. by Zhang et al. (2017).

3 Dereverberation

Dereverberation is one of the signal processing tasks dealing with audio signals. Specifically, dereverberation is an audio signal enhancement task to reduce the detrimental effect of reverberation within a sound. In the following, characteristics and measures of reverberation are discussed in Section 3.1. The evaluation metrics used for dereverberation tasks are presented in Section 3.2, and an overview of dereverberation methods proposed in the literature is given in Section 3.3. My own work in blind dereverberation of music is explained in Section 3.4.

3.1 About reverberation

The term reverberation is used to denote the sound energy caused by the sound waves of the sound source, which are reflecting towards the listener from surrounding obstacles and objects.

In many environments, reverberation is considered improving the sound quality. Specifically concert halls are designed to produce optimally pleasant reverberation for the performed music. Pleasant reverberation is described in Beranek (2012) with terms like acoustical fullness, warmth, softness, definition/clarity, liveness, intimacy/presence, spaciousness, timbre/tone color etc. . A space with no or very little reverberation is appointed dry or dead, and people generally consider overly dry acoustics to be unpleasant. However, too much reverberation is considered to deteriorate the sound. With excess reverberation, sounds smudge i.e. clarity of the sound degenerates. This causes the speech intelligibility to reduce and music to become more like broadband noise.

Properties of reverberation in a space are determined by the acoustical characteristics of materials of all the barriers and objects in the space (Kuttruff (2016)). A hard surface reflects most of the sound energy hitting it. On the contrary, a material with porous surface and non-rigid structure absorbs a good amount of sound energy which encounters it. Some materials, which are fluffy enough, are acoustically transparent and have no effect in sound propagation. In addition to overall reflective / absorbing capability, every material treats different wave frequencies differently. Most materials attenuate high frequency waves more than low frequency waves.

All the above mentioned aspects suggest that the behavior of sound waves in a bounded space is very complex. The sound scape also depends much on positions of the sound sources. Within the sound scape, the perceived sound depends on position of the listener. For a specified positions of a sound source and a listener, an acoustic impulse response (AIR), called also as room impulse response (RIR), $\mathbf{h}_{\mathrm{SL}}$ can be measured (Kuttruff (2016)). AIR captures acoustical properties of the sound path from the sound source to the position of the listener in the space. The sound wave induced by a sound source is heard

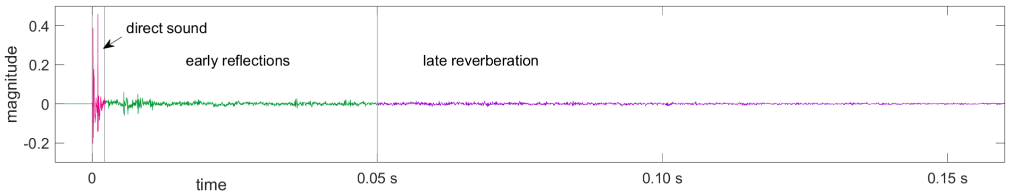


Figure 3.1: Acoustic impulse response of a lecture room from Aachen Impulse Response Database (Jeub et al. (2009)). AIR may be divided into parts of the direct sound, early reflections up to following 50-80 ms and late reverberation after that.

by the listener as a reverberant sound, which builds up as a convolution of the original sound and AIR as

$$x(t) = [\mathbf{s} \circledast \mathbf{h}_{\text{SL}}](t) = \sum_{\mathtt{t}=0}^{\infty} s(t-\mathtt{t})h_{\text{SL}}(\mathtt{t}), \tag{3.1}$$

where s denotes the sound source, $\circledast$ is the mathematical convolution operator and $x(t)$ denotes the heard reverberant sound. The Figure 3.1 shows an example of a room impulse response. The first peak of AIR corresponds to the direct sound path from the source to the listener. The following distinguishable peaks up to 50-80 ms correspond to early reflections, i.e. the sound paths with just a few reflections on the way from the source to the listener. The rest of the AIR corresponds to late reverberation, which usually forms a diffuse sound field, where the direction of sound is indistinguishable.

Since exact AIR is dependent in the position of sound source as well as position of the listener, some more general measures of reverberation are often used. Gross measures of reverberation are e.g. reverberation time (RT) and early decay time (EDT). Both of them are defined based on instantaneous sound pressure level SPL$= 20\log_{10}(P_{\text{RMS}}/20\mu Pa)$ dB with P_{RMS} being the root mean square (RMS) sound pressure. Instantaneous SPL levels for RT and EDT may be defined via AIR, or by measuring the space response to abruptly ending sound source. The reverberation time is denoted as RT60 or RT30 depending on the manner of defining the time needed for SPL to decay 60 dB. These SPL decay times may be defined for a broad band signal as well as separately for different frequency bands. EDT is defined as six times the time needed for SPL to quiet down 10 dB.

The highly important property of sound respected by a listener, which is heavily affected by reverberation, is the clarity of the sound. Clarity of sound field is often measured as early-to-reverberant ratio (ERR), i.e. a ratio between the direct and early reflections part of the sound and the late reverberation part of the sound. The definition of clarity as ERR corresponds to the definition of the signal-to-noise ratio (SNR) considering the direct sound and early reflections before the mixing time, $T_{\text{early}} = 50...80$ ms, as the good quality signal and the late reverberation after T_{early} as noise. Sound clarity as ERR is given by

$$\text{ERR}_{\tau} = 10\log\left(\frac{E_{0\ldots T_{\text{early}}}}{E_{T_{\text{early}}\ldots\infty}}\right)\text{dB} \quad \text{with} \quad E_{a\ldots b} = \sum_{\mathtt{t}=a}^{b} h_{\text{SL}}^2(\mathtt{t}) \tag{3.2}$$

where $E_{0\ldots T_{\text{early}}}$ is the energy of the direct sound and the early reflections up to time T_{early} in AIR, and $E_{T_{\text{early}}\ldots\infty}$ is the energy of the late reverberation after that. Clarity is a commonly used measure of reverberation conditions and an estimated ERR is used within some dereverberation algorithms. For speech, the clarity C_{50} with $T_{\text{early}} = 50$ ms is used as a standard measure. For music, C_{80} with $T_{\text{early}} = 80$ ms is commonly used.

3.2 Evaluation metrics for audio dereverberation

The goal in the dereverberation task varies depending whether the reverberation should be removed altogether, i.e. to perform perfect deconvolution of the sound to the source signal and the room impulse response, or the goal is only to remove excess reverberation to make the sound more pleasant and clear. Sometimes audio enhancement and dereverberation are done as a preprocessing step in frameworks for automatic signal interpretation. In that case the signal enhancement should be optimized for maximizing the interpretability of the signal for classification algorithms. Thus it is not obvious how to obtain a score for the success in the dereverberation task.

The existing evaluation methods differ in many aspects. A distinguishing property being whether a methods expects a clean source signal to be available for comparison, or the method is to be used for signals from the wild where the clean source signal is out of reach. Some of the audio quality metrics are subjective, i.e. they rely on human opinion about the dereverberated audio quality, while other metrics are objective, providing a numeric rating for the dereverbarated audio by computational means. Generally, subjective assessment should be preferred over objective metrics as none of the known metrics captures all the aspects of sound that human does.

The International Telecommunications Union (ITU) Telecommunications Standardization Sector (ITU-T) and Radiocommunications sector (ITU-R) both have given guidelines for audio quality assessment. They define different subjective evaluation setups depending on which aspect of the signal is to be evaluated, as well as scales for scoring to be used. A popularly utilized evaluation scale is for example the mean opinion score (MOS) with scoring: 1-bad 2-poor 3-fair 4-good 5-excellent. In the REVERB challenge (Kinoshita et al. (2016)), a contest for dereverberation algorithms arranged in 2014, a subjective MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) test, described in ITU-R Recommendation BS.1534-1 (2003), was used. In a MUSHRA test, an assessor is commissioned to set multiple audio samples in an order of increasing quality and also to give a numerical quality score for each sample. The test includes a reference signal as a benchmark for the best possible quality, and an anchor signal as a benchmark for the worst possible quality to enhance the comparability of numerical scores.

Computational methods for audio enhancement assessment are very popular due to their effortless applicability. A widely used audio quality evaluation measure by ITU-T is PESQ (perceptual evaluation of speech quality) introduced in Rix et al. (2001). In PESQ computation, many properties of human hearing are taken into account by utilizing Bark frequency scale, intensity conversion to loudness as Sones, discarding small differences where auditory masking in frequency and time take place, and emphasizing sparsity of disturbances both in frequency and time. PESQ score intends to mimic the mean opinion score of subjective evaluation. Similar, although less popular, evaluation metric which also takes into account many aspects of human hearing, is a Perceptual Evaluation of Audio Quality (PEAQ) measure defined in ITU-R Recommendation BS.1387-1 (2001).

A simple computational audio quality measure that we have used in our work on music dereverberation in Publication III is signal-to-distortion ratio (SDR). It is adapted from audio source separation field, where it is popularly used (Vincent et al. (2006)). SDR is given by

$$\mathrm{SDR} = 10\ \log_{10}\left(\frac{||\mathbf{s}_{\mathrm{clean}}||_2^2}{||\hat{\mathbf{s}} - \mathbf{s}_{\mathrm{clean}}||_2^2}\right)\mathrm{dB} \tag{3.3}$$

where $\mathrm{s}_{\mathrm{clean}}$ and $\hat{\mathrm{s}}$ are the time aligned clean reference and the dereverberated signal vectors, respectively.

Other simple computational measures for dereverberation assessment proposed in the literature are e.g, the cepstral coefficient based Cepstrum distance (Kubichek (1993)), the linear prediction coefficient based log-likelihood ratio (LLR) (Quackenbush et al. (1988)) and DFT spectral magnitude based frequency-weighted segmental signal-to-noise ratio (FWsegSNR) (ITU-T Recommendation P.262 (2005)). All these methods utilize a reference signal for the comparison. Computational dereverberation assessment methods, which do not require a reference signal are e.g. speech to reverberation modulation energy ratio (SRMR) (Falk et al. (2010)) and direct-to-reverberant ratio (DRR) (Naylor and Gaubitch (2010)), normalized signal-to-reverberation ratio (NSRR) (Naylor et al. (2010)) and reverberation decay tail (RDT) (Wen et al. (2006)).

3.3 Dereverberation methods within literature

The research on algorithms for removing reverberation from audio signal concentrates mostly on speech audio (Naylor and Gaubitch (2010)), on improving its intelligibility. This is well justifiable, as the most essential applications facilitate communication via mobile devices and serve the hearing impaired for improved quality of life. However, there exists also research on focusing music dereverberation where aesthetical viewpoint is more important. Dereverberation is also often considered as pre-processing step for audio content analysis algorithms (Watanabe et al. (2018)).

There are a variety of different methods available for dereverberation, depending on what is assumed to be known about recording conditions, e.g. number of microphones, room impulse responses, reverberation time, noise statistics etc. . The most efficient methods utilize multiple microphones, but there exists also many single channel dereverberation algorithms. Majority of methods operate in time-frequency domain, as time-frequency domain processing provides efficient means for robustness against changes in acoustic channel.

In the following, some popular processing principles utilized for audio dereverberation are presented. The methods are dealt within six topics. First it is discussed how human vocalization model is utilized for dereverberation. Then the variety of ways the linear predictive (LP) analysis is utilized within dereverberation algorithms is explained. The third topic concerns the methodology used for rigorous reverberation cancellation via AIR estimation end deconvolution filtering. The fourth topic discusses multi-microphone spatial filtering approaches to dereverberation. Then, the introduction turns to statistically inspired dereverberation methods, which use weight based suppression of DTFT magnitudes. Finally, the dereverberation implementations by neural networks are addressed.

In the discussion below, the reverberant signal $x(t)$ is represented in terms of parts $s(t)$, the clean source signal, $r(t)$ the reverberation and $n(t)$ additional noise as

$$x(t) = s(t) + r(t) + n(t). \tag{3.4}$$

In case of STFT-domain processing, the frame spectra $X(n, f)$, $S(n, f)$, $R(n, f)$ and $N(n, f)$ are assumed to sum up similarly as

$$X(n, f) = S(n, f) + R(n, f) + N(n, f). \tag{3.5}$$

Speech dereverberation utilizing source-filter vocal-tract model

The first implementations of speech dereverberation in 1970's (Allen (1974)) were based on the source-filter model of speech production. The speech signal $s(t)$ was estimated directly without modeling $r(t)$ and $n(t)$ at all. The source-filter model of speech, presented e.g. in Stevens (1998), contains an all-pole filter, which represents the vocal track and gives the characteristics of each phone and human voice in general. The source signal of the model represents the glottal sound, which is assumed to be a time series of pulses in case of voiced sounds, or white noise in case of fricative phones.

Dereverberation of reverberant speech based on the source-filter model is performed leaning on an observation that the all-pole vocal tract filter coefficients are not much affected by reverberation, but most disturbance resides in the estimated glottal source signal (Gaubitch et al. (2006)). The vocal tract filter coefficients $g(t)$ are thus estimated from the reverberant signal using e.g. LP-estimation on the reverberant signal $x(t)$ as

$$x(t) = \sum_{\mathtt{t}=1}^{T} g(\mathtt{t}) x(t - \mathtt{t}) + e(t), \tag{3.6}$$

where T is the prediction filter length and error $e(t)$ is to be minimized. The prediction error $e(t)$ is then taken as the noisy glottal signal. To obtain dereverberation, it is cleaned up of noise. This approach, often called LP-residual processing, has been used e.g. by Gaubitch et al. (2003).

Characteristics of voiced vowels, particularly the harmonic structure of them, following the fundamental frequency f_0 of the glottal pulse signal, is utilized in the HERB (Harmonicity based dEReverBeration) approach by Nakatani et al. (2007). The source-filter model of human speech production is utilized in conjunction with other processing principles presented below for speech dereverberation e.g. by Yoshioka et al. (2007) and Kinoshita et al. (2009).

3.3.1 Linear predictive analysis for reverberation estimation

Auto-regressive signal model is extensively utilized for estimating the reverberation part of the recorded signal (Naylor and Gaubitch (2010) Chapter 4). The prediction error is then considered as the clean source audio, which is not possible to be predicted based on previous observations. The signal model, in absence of noise, is thus in time domain or in STFT-domain respectively as

$$x(t) = s(t) + r(t) = s(t) + \sum_{\mathtt{t}=d}^{T} g(\mathtt{t}) x(t - \mathtt{t}) \quad \text{and} \tag{3.7}$$

$$X(n, f) = S(n, f) + R(n, f) = S(n, f) + \sum_{\mathtt{n}=D}^{L} G(\mathtt{n}, f)^* X(n - \mathtt{n}, f) \tag{3.8}$$

where * denotes complex conjugate, t and n are time indices, T and L denote the length of prediction filter, d and D denote prediction delay and g and G denote the linear prediction coefficients for $x(t - \mathtt{t})$ and $X(n - \mathtt{n}, f)$ in time and DTFT-domains respectively. Dereverberation then actualizes as

$$\hat{s}(t) = x(t) - \hat{r}(t) \quad \text{or} \quad \hat{S}(n, f) = X(n, f) - \hat{R}(n, f). \tag{3.9}$$

The parameters g or G of (3.7) are solved minimizing the power of the estimation error $e(t) = x(t) - \hat{r}(t)$ or $E(n, f) = X(n, f) - \hat{R}(n, f)$, e.g. using LP-analysis (Jackson (1989)). The time-domain model is highly sensitive to changes in AIR, but the DTFT-domain model has shown to perform nicely already in case of single channel signal, both in case of both speech (Padaki et al. (2013)) and music signals (the Publication III).

In case of multi-channel recording, reverberation r_1 (R_1) in the first channel signal x_1 (X_1) is estimated using all the M channel signals $x_m(t)$ ($X_m(n, f)$) for , $m\!=\!1...M$ as

$$\hat{r}_1(t) = \sum_{m=2}^{M}\sum_{\mathtt{t}=d}^{T} g_m(\mathtt{t})x_m(t-\mathtt{t}) \quad \text{or} \quad \hat{R}_1(n, f) = \sum_{m=2}^{M}\sum_{\mathtt{n}=D}^{L} G_m(\mathtt{n}, f)X_m(n-\mathtt{n}, f), \tag{3.10}$$

where x_m or X_m denote the signal on the m:th channel of the recording and g_m or G_m denote the corresponding regression coefficients. The parameters $g_m(\mathtt{t}), \mathtt{t} = d...T$ or $G_m(\mathtt{n}, f)$, $\mathtt{n} = D...L$ for $m = 1...M$ are similarly to above solved by minimizing the estimation error $e(t) = x_1(t) - \hat{r}_1(t)$ or $E(n, f) = X_1(n, f) - \hat{R}_1(n, f)$. A solution minimizing the squared error e^2 (E^2) is obtained using multi-channel linear prediction (MCLP) (Naylor and Gaubitch (2010) Chapter 5). The problem with the generally used least squares minimization of linear prediction analysis is that the error signal becomes white Gaussian. This assumption does not hold for most real life audio source signals. To fix this, e.g. prewhitening of signals in Triki and Slock (2005), or increasing prediction delay in Kinoshita et al. (2009) have been proposed. The time-domain implementation of the MCLP-method is also known as linear-predictive multi-input equalization (LIME), and the STFT-domain MCLP has been also referred to as weighted prediction error method (WPE). MCLP is extensively utilized in STFT-domain due to its robustness to changes in AIR.

In addition to the better robustness, STFT domain processing allows statistical models to be utilized for the signal, giving rise to a technique called variance-normalized delayed MCLP (NDLP) proposed in Nakatani et al. (2010). For NDLP, the complex-valued STFT coefficients $S(n, f)$ of the source signal are modeled using a time-varying Gaussian (TVG) model. The TVG setup usually chosen, defines the complex Gaussian distributions of $S(n, f)$ to be zero mean, each with its own variance. Then the regressors $G_m(n, f)$ of the MCLP and the variances $\sigma^2_{n,f}$ of the TVG-model are optimized in a maximum-likelihood (ML) sense, with expectation-maximization (EM) type alternating optimization algorithm. Due to combining the statistical TVG model to MCLP, NDLP is also referred to as Bayesian blind deconvolution.

3.3.2 Blind system identification and deconvolution methods

Dereverberation algorithms called *reverberation cancellation* (Naylor and Gaubitch (2010)) methods utilize the audio channel characteristics, that is AIR (or RIR), for the dereverberation solution. Usually AIR $\mathbf{h}$ is unknown and it must be estimated using some *blind system identification* (BSI) approach. Then the estimated AIR $\hat{\mathbf{h}}$ is utilized for estimating inverse (i.e. equalization, deconvolution) parameters $\mathbf{g}$ to cancel the effect of AIR from the signal as

$$\hat{s}(t) = \sum_{\mathtt{t}=0}^{T-1} g(\mathtt{t})x(t-\mathtt{t}). \tag{3.11}$$

These time-domain methods are mathematically and physically faithful to true sound propagation laws encoded in (3.1) and thus they have high potential for very high fidelity dereverberation. Unfortunately rigorous modeling is often accompanied with poor robustness against channel estimation errors and small changes in AIR, which are serious issues when working with real life signals.

For AIR estimation, multi-channel recording is necessary. AIRs may be estimated based on differences and similarities between signals from different microphones. There are two mainstream BSI methods available for AIR-estimation One is based on minimizing averaged cross-relation error over all pairs of M microphone signals $\mathbf{x}_m, m = 1...M$. The cross-relation error between microphone signals $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is defined as $\mathbf{e}_{ij} = \mathbf{s} \circledast \mathbf{h}_i \circledast \mathbf{h}_j - \mathbf{s} \circledast \mathbf{h}_j \circledast \mathbf{h}_i = \boldsymbol{x}_i \circledast \mathbf{h}_j - \boldsymbol{x}_j \circledast \mathbf{h}_i$, where $\mathbf{h}_m$ denotes AIR between the source and microphone m. All the M AIRs are then approximated minimizing the combined error of $\binom{M}{2}$ cross-relation error equations of $\mathbf{e}_{i,j}$ $i = 1...M-1$, $j = i+1...M$. The other mainstream time-domain BSI-approach finds AIRs from the null space $\{\, \mathbf{v} \mid \mathbf{R}\,\mathbf{v} = \mathbf{0} \,\}$ of multi-channel data correlation matrix $\mathbf{R}$ of size $(MT) \times (MT)$, which consists of $M \times M$ blocks of cross-correlation matrices $\boldsymbol{\rho}_{ij}$ between channel signals $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $\boldsymbol{\rho}_{ij}$ being of size $T \times T$.

To estimate the equalization parameters in $\mathbf{g}_m$ based on $\hat{\mathbf{h}}_m$, more sophisticated methods than direct inversion are necessary. This is due to that the noise level in estimated AIR-parameters $\hat{\mathbf{h}}_m$ is generally high and AIRs $\mathbf{h}_m$ are generally non-minimum phase (Habets and Naylor (2018)). For robust estimation of $\mathbf{g}_m$, many solutions utilize an equalized impulse response

$$\text{EIR}(t) = \sum_{m=1}^{M} \sum_{\mathtt{t}=0}^{T-1} \mathbf{h}_m(t - \mathtt{t})\mathbf{g}_m(\mathtt{t}), \tag{3.12}$$

which ideally produces an impulse function EIR(t), possibly with a delay. A Multiple input/output theorem (MINT) in Miyoshi and Kaneda (1988) provides a least squares solution for EIR(t) to become an impulse function. This solution only solves the problem of AIR being non minimum phase, but is not robust against AIR estimation errors. Solutions for greater robustness against errors in $\hat{\mathbf{h}}$ implement different ways of relaxation from targeting the perfect impulse EIR(t). The different relaxation approaches include e.g. channel shortening in Zhang et al. (2010) and Lim et al. (2014), partial equalization in Kodrasi and Doclo (2012) and sparse optimization in Mertins et al. (2010).

3.3.3 Clean signal estimation using spatial filtering

Spatial filtering, i.e. beamforming, is dedicated for extracting a high quality signal from a certain geometrical direction by suppressing sounds and echoes coming from other directions. The necessity for spatial filtering is a microphone array, whose geometry is known. Also the direction of arrival (DOA) of the source signal must be known or estimated for the spatial filter optimization. A minimum variance distortionless response (MVDR) design Capon (1969) provides a beamforming filter to steer the focus to certain direction while minimizing the sound energy from other directions. An MVDR-filter $\mathbf{g}_{\text{MVDR}}$ (or $\mathbf{G}_{\text{MVDR}}$) may be used for dereverberation as

$$\hat{s}(t) = \sum_{m=1}^{M} \sum_{\mathtt{t}=0}^{T-1} \mathbf{g}_{\text{MVDR}}(m, \mathtt{t})\boldsymbol{x}_m(t - \mathtt{t}) \quad \text{or} \quad \hat{S}(n, f) = \sum_{m=1}^{M} \sum_{\mathtt{n}=0}^{L-1} \mathbf{G}_{\text{MVDR}}(m, \mathtt{n}, f) X_m(n - \mathtt{n}, f) \tag{3.13}$$

for time-domain or STFT-domain processing, respectively. These equations perform very similar operation as MCLP presented above, as discussed in Dietzen et al. (2016), although the MVDR filter design process is based on an estimated DOA and signal statistics rather than linear prediction analysis Haykin (2008). In the dereverberation overview of Habets and Naylor (2018) MVDR-filtering is grouped as a *signal independent spatial filtering* method although its beamforming optimality is based on statistics of the data. This is likely said to contrast MVDR-filtering to beamformers, where the coefficients of an MVDR beamformer are further tuned in terms of Wiener filtering for instantaneous signal contents. The methods implementing this kind of enhanced beamformer optimization on are called *signal dependent beamforming* methods in Habets and Naylor (2018).

An alternative structure, which implements a beamformer similar to MVDR, but provides insight as well as simplifies the beamformer implementation, is the generalized side-lobe canceler (GSC) structure. The idea of GSC is to use a fixed beamformer for steering the attention at the desired direction, and another filter for minimizing the noise and reverberation power. The fixed beamformer is entirely independent of data as well as its statistics and provides the main lobe directivity similarly to a MVDR filter. The noise reducing filter on the other hand is optimized to minimize the noise power in the particular signal, the function which in MVDR is integrated within the steering beamformer coefficients. The GSC structure provides an opportunity to set larger variety of constraints and design parameters on the noise reduction filter, than what is possible to obtain with MVDR -filter design methodology. For example AIR characteristics are incorporated in the generalized side lobe canceler design procedure in Gannot et al. (2001).

3.3.4 Spectral enhancement and time varying Gaussian signal model

The dereverberation algorithms called *spectral enhancement*, *spectral subtraction* or *reverberation suppression* methods operate in time-spectral STFT-domain. These methods assume that the reverberant signal $X(n,f) = S(n,f) + R(n,f) + N(n,f)$ and all of its parts $S(n,f)$, $R(n,f)$ and $N(n,f)$ are coming from complex zero-mean time varying Gaussian distributions $\mathcal{N}_{\mathbb{C}}\{0, \sigma_t^2\}$. Dereverberation with these methods is done by a multiplicative operation, instead of subtraction, on power spectral densities in each signal frame as

$$|\hat{S}(n,f)|^2 = \max\big(\ \lambda, \quad W(n,f) \cdot |X(n,f)|^2\ \big), \tag{3.14}$$

where a spectral floor λ is set to diminish a musical noise problem often encountered with STFT-processing, and $W(n,f)$ is the gain, or suppression coefficient, for the time-spectral bin. Often, the signal parts S, R and N are assumed to be uncorrelated, and thus maximum likelihood gains $W(n,f)$ for single-channel dereverberation are given by

$$W(n,f) = \frac{\sigma_S^2(n,f)}{\sigma_X^2(n,f)} = \frac{\sigma_X^2(n,f) - \sigma_R^2(n,f) - \sigma_N^2(n,f)}{\sigma_X^2(n,f)}, \tag{3.15}$$

where σ_X^2, σ_S^2, σ_R^2 and σ_N^2 denote the power spectral densities (PSD), of the recorded signal and the source, reverberation and noise signal parts, respectively. An estimate of the recorded signal PSD is obtained as $\hat{\sigma}_X^2(n,f) = |X(n,f)|^2$. In most of the algorithms, the noise PSD σ_N^2 is assumed known, which means in practice that it is estimated from frames $X(n)$ of silence within the recording. For detecting the frames of silence, any voice activity detection (VAD) technique may be utilized. The reverberation PSD σ_R^2 of the reverberation part may be estimated based on the recording PSD for example as

$$\sigma_R^2(n,f) = e^{-v}\ |X(n-D,f)|^2, \tag{3.16}$$

where D is set to correspond to late reverberation start time and the decay rate υ depends on reverberation time T_{60} among other system parameters. A frequency dependent decay rate $\upsilon(f)$ has been utilized in Habets (2004). Alternatively, delayed auto-regressive model (3.7) is also widely utilized for reverberation PSD estimation as $\hat{\sigma}_R^2(n, f) = \hat{R}^2(n, f)$. This LP-based approach has been utilized in Publication III, where we found that utilizing only one predictive term, i.e. $L = D$, was the most effective setup, which relates the approach to that of Habets (2004). When multiple channels are available, reverberation PSD σ_R^2 may be estimated using a residual $R_1(n, f) = X_1(n, f) - \hat{S}_1(n, f)$ based on the multi-channel estimate $\hat{S}_1(n, f)$, or alternatively using non-coherence among the time aligned microphone signals.

In case of a multi-channel signal, spectral enhancement is often utilized after beamforming, on the obtained single-channel signal $\hat{s}$ (or $\hat{S}$). The spectral subtraction type post-filtering after beamforming is then done for improved dereverberation and noise reduction.

3.3.5 Dereverberation using neural networks

Neural networks have been used for dereverberation task, e.g. by Xiao et al. (2014), as well as for so many other signal processing tasks also. To train a neural network to output an estimate of clean audio, the NN must be trained using target audio material with the clean audio available. The training material should naturally represent the scenario of the anticipated NN usage as closely as possible, thus making this approach dependent on relevant training material. In case AIRs of the space, the dereverberation is to be used within, is possible to be measured, simulation is an efficient means for generating parallel training data for derevereration NN training. Simulation is done by convolving clean source material with measured or approximated AIRs according to (3.1) and adding different kinds of noise signals on the reverberant sound.

Audio signal is a form of time series, rather than consisting of independent samples. In order a neural network to estimate the reverberation and noise free sample value or STFT frame, the context of each input sample or STFT frame should be encoded in the NN input. In Han et al. (2015), a feed forward DNN for estimating log spectral magnitudes $\log(|S(n, f)|)$ of clean speech has been tried out. Five log spectral magnitude frames $\log(|X(n + \mathtt{n})|)$, $\mathtt{n} = -5 ... + 5$ anterior and posterior to frame n are utilized as an input to the DNN to encode the context around $S(n)$. In Weninger et al. (2014) a recurrent neural network (RNN), specifically a long short-term memory network (LSTM) (Hochreiter and Schmidhuber (1997)), has been utilized for reverberant feature enhancement for ASR. A recurrent type neural network implicitly holds information of the past samples, thus single frame features are used for the LSTM input. However, it is not obvious how the dereverberated audio would be extracted based on the enhanced Mel-spectral features.

3.4 Results in blind dereverberation of music

The work of the Publication III pursues enhancing music recordings subject to reverberation and dynamic range compression (DRC). We produced clean music from MIDI-representations and then applied reverberation and dynamic range compression to the signal to synthesize the needed data. For reverberation suppression, we use no external knowledge about the acoustic conditions of the recording environment. Thus the solution of choice is a blind dereverberation method, namely a method proposed in Furuya and

Kataoka (2007) for speech dereverberation, which is applied to music signals in our work. The aim was to test whether this method performs well with music material distorted by DRC-processing.

We estimate the reverberation within the STFT signal representation using LP-analysis with signal model

$$|X(n,f)| = |S(n,f)| + |R(n,f)| \approx |S(n,f)| + \sum_{\mathtt{n}=1}^{L} a_f(\mathtt{n}) \cdot |X(n-\mathtt{n},f)|, \tag{3.17}$$

where $|X(n,f)|$, $|S(n,f)|$ and $|R(t,f)|$ denote respectively spectral magnitudes of the reverberant signal, the clean signal and the reverberation in frequency band f in signal frame n, and L is the length of the LP-filter with frequency band specific coefficients $a_f(\mathtt{n}),\ \mathtt{n} = d...L$. The frequency band specific model parameters $\mathbf{a}_f = [\, a_f(1),\ a_f(2), ...$ $...,\ a_f(L)\,]'$ were estimated based on all the magnitude spectra $|X(n)|$ of the recording at hand. The standard least squares solution

$$\begin{aligned} \mathbf{a}_f &= (\, V_f{}' \, V_f \,)^{-1} \, V_f{}' \, \mathbf{v}_f(L{+}1) \quad \text{where} \\ V_f &= [\, \mathbf{v}_f(L),\ \mathbf{v}_f(L{-}1),\ ...,\ \mathbf{v}_f(1)\,] \quad \text{and} \\ \mathbf{v}_f(n) &= [\, |X(n,f)|,\ |X(n{+}1,f)|,\ ...,\ |X(n{+}\mathtt{N}{-}L{-}1,f)|\,]', \end{aligned} \tag{3.18}$$

where $\mathtt{N}$ is the number of audio frames in the recording was used. The dereverberated STFT magnitude spectrum was then obtained as

$$|\hat{S}(n,f)| = |X(n,f)| - \beta_f \cdot \sum_{\mathtt{t}=1}^{L} a_f(\mathtt{t}) \cdot |X(n-\mathtt{n},f)|, \tag{3.19}$$

where frequency dependent weights β_f are used to limit the amount of dereverberation. The dereverberated time domain signal was obtained by utilizing the phase of $X(n,f)$ as $\hat{S}(n,f) = |\hat{S}(n,f)| \cdot X(n,f)/|X(n,f)|$, and performing inverse fast Fourier transform (IFFT) for all $\hat{S}(n),\ n = 1...\mathtt{N}$ and combining consecutive audio frames by windowed overlap add processing.

The numerical values of used audio frame length for the frame spectral representations $X(n)$, LP-model length L and the weighting function β_f were set using validation data. Utilizing audio frame lengths from 20 ms to 160 ms with different linear prediction lengths L were tested. According to improvements in signal-to-distortion ratio within validation data, the longer the audio frame, the better the dereverberation quality appeared. In all cases, LP-model length larger than $L = 3$ did not give improvements. Often, model length $L = 1$ gave the best results. This is reasonable, if we compare the used signal model to the reverberation estimation according to (3.16). The model (3.16) assumes that all the information of the reverberation is present in the frame located around 50 ms – the limiting time between early reflections and late reverberation – after the frame under consideration. Thus for audio frame lengths longer than 50 ms, an LP-model length $L = 1$ should be enough. Our experiments confirmed that this is true.

Our tests with different weighting functions revealed that weights β_f for low frequency bins dominate the performance. Thus we finally resorted to using constant weighting $\beta_f = \beta$. The value of β that gave best results on average was between 0.2-0.3 depending on whether the distortion by dynamic range compression was present or not. Slightly larger value of β seemed to be the best for dynamically compressed signals. Using the best performing system parameters, the signal to distortion ratio of the reverberant signals was improved from 6.1 dB to 6.4 dB, and the SDR of signals suffering also DRC distortion improved from 5.2 dB to 5.6 dB.

Thus in Publication III we showed that this dereverberation framework performs well with music signals. Another finding was that the dynamic range compression of audio does not deteriorate the dereverberation performance. In contrast, dereverberation performance with dynamic range compressed audio appeared even better than with the non-compressed audio.

4 Classifying independent samples

In artificial intelligence (AI), computational methods are used to identify different phenomena from signal, e.g. image or a sound clip. After preliminary analysis of the signal, described in Section 2, an AI-system makes decisions according to internal models and action rules, which are utilized for providing the solution. The basic building blocks of an AI-system, which are responsible for interpreting the signal, are called classifiers. They provide phenomenon level information for making the decisions of action. Thus classification, or categorization, is in the core of almost every intelligent application.

To make the problem of automatic categorization, i.e. classification, of a signal tractable, usually a small set of possible categories, called *classes*, is predefined, instead of evaluating among all the feasible taxonomies. The signal is analyzed to solve whether it represents one of the pre-defined categories of phenomena. In the most fundamental form of classification, *binary classification*, the sample is assigned to one of only two categories. Binary classification may also be considered as *detection*, where the categories are simply the 'target' class, which represents the inquired phenomenon, and the 'non-target' class, which is associated with everything else. Via utilization of multiple binary classifiers or detectors, these methods may be also utilized to produce classification decisions among more than only two categories. In that case multiple binary classifiers or detectors, at least one for spotting each class, are utilized, and their outputs are combined to perform the final classification.

In this chapter I first discuss how performance of a classification framework is evaluated in Section 4.1. Then, in Section 4.2 I discuss about different methodology for computational classification, and in Section 4.3 I present different ways to combine different methodologies into single classification framework. In Section 4.4 I present a binary classifier combination function named BOA, which is proposed in Publication IV and elaborated in Publication VI.

4.1 Classification result evaluation

To evaluate the classification result of a computational framework, the true categorical class memberships of the signal samples must be known. The success is evaluated against the knowledge about this true class membership. The class label given by a classifier can only be correct or wrong. To get statistical information about the performance of a classifier, a bunch of signal samples must be classified with the framework, and statistics about correct and incorrect classifications must be collected. The fundamental statistics, which are also used to define other evaluation metrics for binary classification are the counts of

true positives (tp), i.e. test samples correctly classified as 'target' true negatives (tn), i.e. test samples correctly classified as 'non-target'

false positives (fp), i.e. test samples incorrectly classified as 'target'

false negatives (fn), i.e. test samples incorrectly classified as 'non-target'

In case of multi-category classification, the counts of true positives (tp_c), false positives (fp_c) and false negatives (fn_c) are separately computed for each 'target' class c, considering all the other classes as 'non-target'. Statistics about decisions and errors between different classes in terms of tp_c and fp_c for each class are often presented in the form of a *confusion matrix* shown in Figure 4.1.

The simplest statistic often reported as the value for general success of a classification framework is accuracy

$$\mathrm{ACC} = \frac{tp + tn}{N} = \frac{1}{N}\sum_{c} tp_c \tag{4.1}$$

where N is the total number of classified samples. Closely related to accuracy, an often reported statistics for evaluating binary classification, are true positive rate (*tpr*) and true negative rate (*tnr*)

$$tpr = \frac{tp}{tp + fn} \qquad tnr = \frac{tn}{tn + fp}. \tag{4.2}$$

The terms *sensitivity* and *specificity* are also used alternatively for the *tpr* and *tnr*, respectively. In multi-category classification frameworks the tpr_c may be computed separately for each class c. In binary classification *tpr* is also called *recall*. Recall R is usually reported in conjunction with *precision* P

$$P = \frac{tp}{tp + fp}. \tag{4.3}$$

Precision is about how precisely the framework is able to distinguish the class in question from the other class or classes.

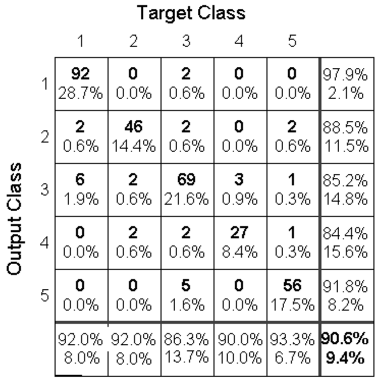
A measure combining the information from precision P and recall R is F$_\beta$-score

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R} = \frac{(1+\beta^2)\,tp}{(1+\beta^2)\,tp + \beta^2 fn + fp}. \tag{4.4}$$

F_β may be adjusted with β^2 to take into account the true class distribution and the cost of incorrect classification. Using value $\beta^2 > 1$ penalizes more for not detecting samples of the 'target' class. Thus big β^2 is justifiable if the cost of false negative is high or the 'target' class forms a minority of test samples. Again, $\beta^2 < 1$ is a rational choice if the 'target' class is prevalent, or if the cost of false positive is high. For even class distribution or equal error cost $F_1 = 2PR/(P + R) = 2\,tp/(2\,tp + fn + fp)$ is used. In addition to F_1, which is the harmonic mean of P and R, the geometric mean $G = \sqrt{PR}$ of P and R is sometimes used as an evaluation metric called G-score.

Often a classification framework is parametrized such that the predisposition of the system to assign a certain label can be tuned. In binary classification such a parametrization accounts for changing the prior probability of the framework to assign another label against the other one. In this kind of situation equal error rate (EER) is commonly reported as a measure of overall performance of a system. Using the false positive rate and false negative rate

$$fpr = \frac{fp}{fp + tn} \qquad fnr = \frac{fn}{fn + tp}, \tag{4.5}$$



Output Class \ Target Class	1	2	3	4	5	
1	**92** 28.7%	**0** 0.0%	**2** 0.6%	**0** 0.0%	**0** 0.0%	97.9% 2.1%
2	**2** 0.6%	**46** 14.4%	**2** 0.6%	**0** 0.0%	**2** 0.6%	88.5% 11.5%
3	**6** 1.9%	**2** 0.6%	**69** 21.6%	**3** 0.9%	**1** 0.3%	85.2% 14.8%
4	**0** 0.0%	**2** 0.6%	**2** 0.6%	**27** 8.4%	**1** 0.3%	84.4% 15.6%
5	**0** 0.0%	**0** 0.0%	**5** 1.6%	**0** 0.0%	**56** 17.5%	91.8% 8.2%
	92.0% 8.0%	92.0% 8.0%	86.3% 13.7%	90.0% 10.0%	93.3% 6.7%	**90.6%** **9.4%**

Figure 4.1: Confusion matrix of classification result of data from 5 classes. The number of samples from classes 1, 2, 3, 4 and 5 are 100, 50, 80, 30 and 60, respectively. The *tp* -counts of correctly classified samples can be seen in bold in the diagonal bins of the matrix. The bold numbers in the off-diagonal bins denote *fn* -counts of incorrect classifications. Percentages of the respective *tp* and *fp* -counts in respect to all the 320 samples are shown below each count. The true positive rate (*tpr*) (or *recall*) and false negative rate (*fnr*) for each class are shown in the bottom bins as percentages. The *precision* P and $1 - P$ of the classification result in respect to each class are given in the rightmost bins.

the system is tuned to function at an operating point where *fpr* = *fnr*, which then becomes the value for EER.

With different parameter settings of a binary classification framework, curves of different pairs of performance scores, which express different aspects of the performance, may be outlined. Commonly used curves of classifier performance, depicted in Figure 4.2, are the precision-recall (P-R) -curve and the receiver operating characteristics (ROC) -curve. P-R -curve plots the scores of precision P versus the scores of recall R at each possible operating point, that is, parametrization, of the system. ROC -curve plots the true positive rate (*tpr*) against the false positive rate (*fpr*) at different operating points

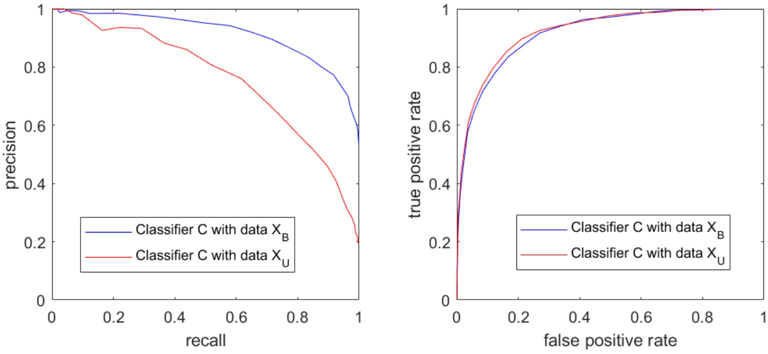


Figure 4.2: Precision vs recall -curves (left) and ROC -curves (right) with classifier C, using data X_B which has even, 1:1, class distribution and data X_U which has class distribution 1:5. It is to be noted how the precision of classification seen in the P-R -curve is affected by the class distribution due to increased count of false positives (*fp*) in respect to the count of true positives (*tp*).

of the framework. To evaluate a classification framework in terms of its whole space of operating points, an are under curve (AUC) -measure is commonly used. It may be defined in terms of the P-R or the ROC -curve respectively as

$$\mathrm{AUC}_{\text{P-R}} = \int_0^1 P \;\, dR, \text{ or} \qquad \mathrm{AUC}_{\text{ROC}} = \int_0^1 tpr \;\, d\,fpr \tag{4.6}$$

4.2 Classification functions

The better the signal analysis and data modeling has been done, the easier is the job of the classifier. At simplest, classifier is a comparison function, which assigns a class label to an input sample by comparing two numbers against each other. In conjunction with detection, this stands for comparing the likelihood value obtained for the target category to a threshold value, which is set to decide about the detection. In conjunction with classification, the decision is made to the class for which an obtained likelihood value is the largest among classes. The likelihood values for classes are usually obtained using a class model, which has been trained using examples from the categories involved in the task. In Section 4.2.1 I present some of this kind of data models popularly used in classification. In Section 4.2.2 I discuss the functions for class discrimination, when class models are not utilized, but a classification function of data features provides the classification.

4.2.1 Modeling phenomena

Numerical models of real world phenomena are in the core of machine learning. Features are models of low level phenomena or general data appearance in specific measurement modality. In this section I talk about models intended for modeling high level phenomena, like face, human gait, car, airplane, musical genre, spoken language etc. . In the literature data models are categorized as either *generative* or *discriminative*. A generative model structure expresses the properties of the phenomenon as faithfully as possible and it is useful for many machine intelligence tasks encompassing that particular phenomenon. Each category specific model is learned separately from data representing that phenomenon. A discriminative model on the other hand is specific to the task it is intended for. The discriminative character of the model is trained and utilized for segregating the phenomenon from the other phenomena present in the specific task. Thus I associate discriminative models with classifies, which are discussed in the next Section. Here I consider only generative models.

Probably the most widely used technique to acquire a static generative data model is the Gaussian mixture model (GMM). GMM is a model for a probability distribution of feature vectors of samples representing a certain phenomenon. The distribution function is a sum of multiple (N) multivariate normal distributions $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_n,\boldsymbol{\Sigma}_n)$ for $n=1,...,N.$, each with unique mean $\boldsymbol{\mu}_n$, covariance matrix $\boldsymbol{\Sigma}_n$ and a magnitude w_n. The likelihood of a sample $\boldsymbol{x}$ to represent the modeled class c is given by

$$l(\boldsymbol{x}) = \sum_{n=1}^{N} w_n \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_n,\boldsymbol{\Sigma}_n) = \sum_{n=1}^{N} w_n \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_n|}} e^{-\frac{1}{2}(\boldsymbol{x}-\mu_n)\boldsymbol{\Sigma}_n^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_n)'}, \tag{4.7}$$

where d denotes the dimensionality of feature space $\boldsymbol{x} \in \mathbb{R}^d$. An example of GMM probability distribution is illustrated in Figure 4.3. The model parameters N, w_n, $\boldsymbol{\mu}_n$ and

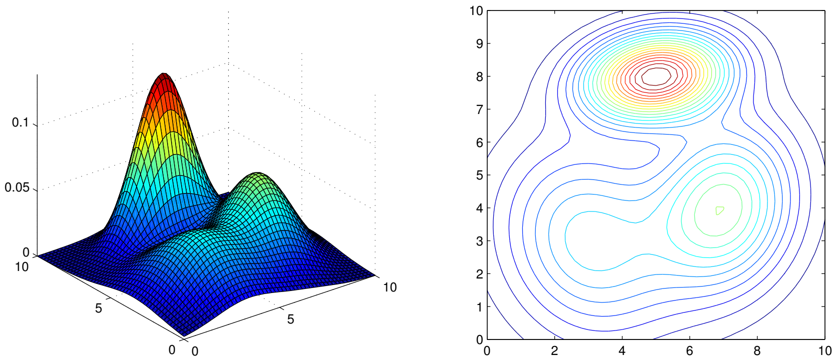


Figure 4.3: Gaussian mixture model of three two-variate Gaussian distributions for modeling the distribution of two element feature vectors that represent a certain phenomenon.

$\boldsymbol{\Sigma}_n$ for $n = 1...N$ are learned from representative training data. The model weights w_n are restricted to be positive and sum up to unity, that is $w_n \geq 0,\ n = 1...N$ and $\sum_{n=1}^{N} w_n = 1$, for the GMM to represent a probability distribution. Often the covariance matrices $\boldsymbol{\Sigma}_n$ are approximated by only their diagonals, that is, using an assumption of independence among feature vector elements. The algorithm suitable for GMM parameter estimation is an EM-algorithm introduced by Dempster et al. (1977). GMMs have been heavily utilized for automatic speech recognition task for modeling characteristics of different phonemes in speech.

The statistical GMM models are often used for Bayesian decision making where the category decision, i.e. classification, is made comparing the posterior probabilities

$$P(c \,|\, \boldsymbol{x}) = \frac{P(\boldsymbol{x} \,|\, c)P(c)}{P(\boldsymbol{x})} \tag{4.8}$$

of the classes, where $P(c)$ and $P(\boldsymbol{x})$ are the prior probabilities of the category c and the data vector $\boldsymbol{x}$, respectively, and the GMM likelihood $l_c(\boldsymbol{x})$ (4.7) of a class c is used for providing the conditional probability $P(\boldsymbol{x} \,|\, c)$.

More expressive models than the static models presented above are needed for modeling events which happen in time with varying speed, e.g. speech, musical progression, human gait and gestures, weather, etc. . That is, to model the feature vector progression in time, dynamic models must be utilized.

A popular generative dynamic data model for modeling sequences of data samples is a hidden Markov model (HMM) (Duda et al. (2012)). A hidden Markov model consists of a number of hidden states $s_i,\ i = 1...S$. Each data vector is assumed to represent one of the states. The states are designed specifically for the application to be such that each of them represents some phase of the episode to be modeled, e.g. phonemes in speech. The number of states depends on the length and variability of the episode, e.g. in automatic speech recognition the long and complex words are modeled with HMMs with more states than the short and simple words. Within a traditional ASR framework, presented in Section 6.1, a HMM state is modeled using a GMM model of MFCC features of audio frames (Rabiner and Juang (1993)). To model the dependence of consecutive data vectors of each other, a transition probability $a_{ij} = P(s_j|s_i)$ is assigned for each pair of states

s_i, s_j within the HMM. In HMM, the probability of the system to be in a certain state s_j depends only on the system state at the previous step and the data feature vector fit $P(\boldsymbol{x} \mid s)$ to the feature distributions of HMM states. Probability of HMM to transit from state s_i to s_j, given that the HMM emits $\boldsymbol{x}$ at state s_j is

$$P(s_j|s_i, \boldsymbol{x}) = \frac{a_{ij} P(\boldsymbol{x}|s_j)}{\sum_{k=1}^{N} a_{ik} P(\boldsymbol{x}|s_k)}. \tag{4.9}$$

An example of HMM for speech recognition is depicted in Figure 4.4. An underlying assumption within the HMM, called Markov property, is that the state progression depends only on one step state transition probabilities. This is not always realistic. However, due to the mathematical tractability, this model is extensively used within many disciplines.

4.2.2 Classification functions beyond value comparison

Beyond the comparison of class probabilities according to data compatibility to above discussed models, the simplest function for binary classification among classes $c \in \{-1, 1\}$ is a linear function on the features $\boldsymbol{x}$ of the sample with thresholding with signum function as

$$\hat{c} = \text{sign}(f(\boldsymbol{x})), \quad \text{where} \quad f(\boldsymbol{x}) = \mathbf{w}'\boldsymbol{x} + b. \tag{4.10}$$

There are many algorithms for learning the feature mapping vector $\mathbf{w}$ and bias b Duda et al. (2012).

Fisher's discriminant and Linear discriminant analysis (LDA) find $\mathbf{w}$ and b via statistics of data. That is, for $c \in \{0, 1\}$, they utilize the means $\boldsymbol{\mu}_c$ and covariance matrices $\boldsymbol{\Sigma}_c$ for the two classes. Fisher's solution is $\mathbf{w} = (\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)$ (Bishop (2006)), while LDA reaches the same solution via Bayesian optimization assuming the classes to be homoscedastic, that is $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_1$. The threshold parameter b is left to be set in some other means. It may be set to the midpoint between the class centers as $b = 1/2 \cdot (\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1)'\mathbf{w}$.

Regression algorithms are used for solving $\mathbf{w}$ and b of (4.10) by setting the class labels 1 and -1 to be the regression model target values. The ordinary least squares (OLS) regression Duda et al. (2012) finds the parameters $\mathbf{w}$ and b by minimizing the modeling

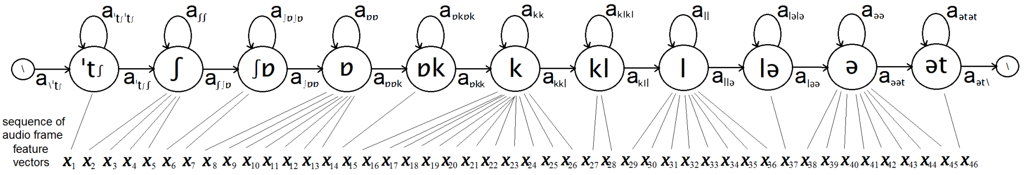

a'tʃ'tʃ
aʃʃ
aʃɒʃɒ
aɒɒ
aɒkɒk
akk
aklkl
all
aləlǝ
aəə
aətət
\
a\'tʃ
'tʃ
a'tʃʃ
ʃ
aʃʃɒ
ʃɒ
aʃɒɒ
ɒ
aɒɒk
ɒk
aɒkk
k
akkl
kl
akll
l
allə
lə
aləə
ə
aəət
ət
aət\
\
sequence of audio frame feature vectors
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20 x21 x22 x23 x24 x25 x26 x27 x28 x29 x30 x31 x32 x33 x34 x35 x36 x37 x38 x39 x40 x41 x42 x43 x44 x45 x46

Figure 4.4: An illustration how a hidden Markov model is utilized for modeling sequences of feature vectors in terms of their phoneme contents in the context of automatic speech recognition. The HMM states are shown as circles, each with a bi-phoneme label enclosed, e.g. 'tʃ. The arrows indicate possibilities of state progression, and each weight $a_{s_1 s_2}$ indicates the probability of transition from state s_1 to state s_2. The thin lines indicate how the feature vectors $\boldsymbol{x}_t$ of the stream of audio frames might be aligned with the HMM states.

error $\sum_{n=1}^{N}(f(\boldsymbol{x}_n) - c_n)^2$ for N training examples $(\boldsymbol{x}_n, c_n)$ with the closed-form solution

$$[\mathbf{w}, b]' = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{c}, \tag{4.11}$$

where matrix $\mathbf{X}$ contains the vectors $X_n = [\boldsymbol{x}_n{}', 1]$ as its rows and $\mathbf{c} = [c_1, c_2, ..., c_N]'$ contains the corresponding class labels 1 or -1 as its elements. A nonlinear decision boundary may be produced with regression analysis, if the vectors X_n are built using nonlinear functions of features in $\boldsymbol{x}_n$.

The partial least squares (PLS) regression, or as better called the projection to latent structures, finds the parameters w and b via an iterative procedure. It picks projections of the feature vectors, via which the regression target scores are pursued, one by one. The procedure allows regularization on the number of projections used and avoids computing the inverse of $\boldsymbol{x}'\boldsymbol{x}$, thus bypassing the possible problem of collinearity of features. There are multiple algorithms to perform the computation for PLS parameters, comparison of which is presented in Andersson (2009).

Logistic regression (Cox (1958)) utilizes the sigmoid function $\sigma(y) = 1/(1 + e^{-y})$ on the linear class estimator $y = f(\boldsymbol{x})$ when looking for suitable parameters w and b for linear classification with (4.10). The error of classification in logistic regression is given by $\mathrm{e}_{-1}(y) = \sigma(y)$ for the samples from class -1, and $\mathrm{e}_{+1}(y) = 1 - \sigma(y)$ for samples of class 1. Now the error is close to zero for every sample falling clearly on the side of its own class on the regression line. This error function is naturally much more faithful to the task of finding the regression model for binary classification than the squared error used by OLS. The cost function to be minimized is formulated logarithmically as $C = \sum_{n=1}^{N}\log(1/(1 - \mathrm{e}_{c_n}(y_n)))$ to make it differentiable. The parameters w and b are then obtained using an iterative maximum likelihood estimation method, for example a Newton-Raphson -algorithm (Atkinson (1979)).

Regularization – To obtain either sparse or smooth distribution of values in w of (4.10), a technique called *regularization* may be used for regression analysis. For L$_P$ regularization on w, the cost function to be minimized is

$$C = \frac{1}{N}\sum_{n=1}^{N}\mathrm{e}(\boldsymbol{x}_n, t_n) + \lambda||\mathbf{w}||_P^P, \tag{4.12}$$

where $\mathrm{e}(\boldsymbol{x}_n, t_n)$ is the error of the regression value $f(\boldsymbol{x}_n)$ in respect to target value t_n and λ is a trade-off parameter between the regression error and regularization error. Similarly to classification function training with the other regression techniques, the targets t_n are usually set to be $t_n \in \{-1, 1\}$ or $t_n \in \{0, 1\}$. Regression using L$_1$ regularization promotes sparsity of w and it is called LASSO. Regression with L$_2$-regularization alleviates possible problems due to collinearity of data and promotes regression vector w to have small Euclidean length. It is called Tikhonov regularization or more commonly as ridge regression and has a closed form solution similar to OLS solution (4.11) as $[\mathbf{w}, b]' = (\boldsymbol{x}'\boldsymbol{x} + \lambda\Gamma)^{-1}\boldsymbol{x}'\mathbf{c}$, where Γ is an identity matrix. So called elastic net regression combines both the L$_1$ and L$_2$ regularizers into the cost function.

In the work of Publications II and I I have utilized OLS with Tikhonov regularization as well as PLS regression for estimating speech state likelihoods within an automatic speech recognition framework.

Support vector machine (SVM) is a successful and abundantly utilized method for binary classification (Steinwart and Christmann (2008)). Formulation of SVM may be used to find parameters for the linear model (4.10) as well as more expressive models which use so called kernel transformations for the distance between the decision boundary and the data points (Bishop (2006)). SVM algorithm sets the decision boundary parameters to maximize the margin between classes. That is, to find the best decision boundary parameters, SVM considers specifically those samples that reside close to a potential boundary. The samples that are classified correctly anyhow are regarded as redundant and are left out of calculations. A versatile and often utilized kernel function for SVM classifier, by which it is possible to produce highly complex partitions of the input space, is the radial basis function (RBF). The SVM with RBF -kernel utilizes e.g. a Gaussian distance function $d(\boldsymbol{p}, \boldsymbol{x}) = \exp(-||\boldsymbol{x} - \boldsymbol{p}||^2/(2\sigma^2))$ to express the deviation of a point $\mathbf{p}$ from a feature vector $\boldsymbol{x}$. The parameter σ affects the smoothness of the resulting decision boundary and must be set via cross validation. Some of the data vectors are selected as support vectors $\mathbf{v}_i,\ i = 1...V$ and the class decision for a new data vector $\boldsymbol{x}$ is then given by $c(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{V} c_i\, d(\boldsymbol{x}, \mathbf{v}_i)\right)$ where $c_i \in \{-1, 1\}$ depending on the class label of each support vector.

Multifaceted, piece-wise linear decision boundaries for separating two or more categories, realize with K nearest neighbors (k-NN) -algorithm (Duda et al. (2012)) and different decision tree algorithms. k-NN functions, similarly to SVM, provides the class label via computing distances to known data samples. However, k-NN requires all the training data to be saved and used for classification of a new sample, while SVM utilizes only the samples selected to be support vectors. Although this is not efficient computationally nor from storage point of view, k-NN is fairly popular due to its simplicity.

Decision tree classifier is a flowchart-like structure which has a root node as a starting point, multiple decision nodes, where intermediate decisions about which node to enter next are made, and plenty of leaf nodes, which finally assign a class label for the input. For most of the decision tree building algorithms, each decision node is designed to be a function of only one variable. The univariate decision makers of a binary decision tree are selected such that the training data would be divided according to the node function into subsets as homogeneous as possible. The selection of an univariate decision maker means selecting an input feature to be evaluated within the node. In case of continuous valued features, also a threshold value for the feature must set to make a binary decision. Hihgly popular Algorithms for decision tree learning are for example ID3 (Quinlan (1986)), C4.5 (Quinlan (1993)) and CART (Breiman et al. (1984)). The ID3 and C4.5 algorithms utilize information gain based on data subset entropies for selecting the node data partition functions. The ID3 -algorithm has been developed for nominal data, where each input attribute may only have one of predefined labels. C4.5 is a successor of ID3, which has a capability of utilizing continuous valued data. Also the CART algorithm (Breiman et al. (1984)) has the capability of working with continuous valued data. It utilizes Gini-impurity or Twoing -criterion to decide about the attributes used in nodes of the tree.

Neural networks are the most versatile classifier functions. Multitude of non-linear intermediate functions within an NN makes the mathematical analysis of the entire NN-function intractable. Currently, research on DNNs pursue understanding the transformations performed by a complex DNN. The research community is furiously, through trial and error, trying to find out guidelines for determining effective DNN structures for each problem, as well as creating new training algorithms for them.

A discriminatively trained DNN is effectively an integrated feature extractor and classifier.

An input to a DNN may be the raw measured signal or a set of simple features, and each layer of a DNN may be considered as a new feature transform leading to the output layer of DNN, which provides the classification. Indeed, as mentioned in Section 2 it has been perceived, that first DNN layers perform transformations similar to low level feature, and the subsequent DNN layers tend to represent higher level structures specific to the data at hand. The output layer might then be considered as the classifier function, which is responsible for the class decision. However, if the DNN is trained in a supervised manner, all the functions are oriented towards solving the particular classification problem at hand. Thus the features obtained with such a DNN are discriminative and they might not be suitable for other tasks as discussed in Yosinski et al. (2014).

In case a DNN is trained in unsupervised way for feature extraction as discussed in Section 2.6, the acquired features way may be used for classification by another DNN or any other classification function. In case of generatively trained DBN, the network is usually trained further for the classification task (Hinton et al. (2006)). One layer of new output nodes is added on top of a DBN for performing the classification function. Then, another training passage is performed using a back-propagation type -algorithm for gradient descent to minimize the error to the target class labels. In addition to learning the parameters of the newly added output elements, also other weights are allowed to change a bit – fine tuned – for better discriminative power.

I have used single layer feed forward neural networks in my Publications V and VI. For the laughter detection task from videos in Publication VI I have trained two NNs for laughter vs. speech/non-laughter discrimination. One of the NNs operates on audio stream of the video, and the other one on video frames. MFCC features form an audio frame are given as an input to the audio related NN, which provides a likelihood of laughter $\in [0, 1]$ within the input frame at its single output. The visual modality based NN operates on face point features, explained in Section 5.4, from images of the video. A single output provides a likelihood of laughter similarly to the other NN. For the ASR task in Publication V I have used an NN which takes MFCC features from 3 consecutive audio frames as an input, and provides likelihoods for the 250 speech states as an output.

4.3 Utilizing multiple classifiers or detectors

Using class likelihood information form multiple models of a phenomenon for classification suggests better classification accuracy, as different models likely provide some uncorrelated information. Similarly, combining decisions made by multiple classifiers for the same task proposes better decisions. Here, methods for combining information from several independent classifiers or detectors are discussed. First shared classification decision making, when multiple different classifiers are available, is addressed. Then decision trees as a form of classifier ensembles is discussed. Finally the attention is focused to utilization of Boolean operators for combining multiple detectors.

Shared decision making

When having a class hypothesis from multiple classifiers, a simple way to combine them is voting. In the next section I discuss about Boolean combination, which may be seen as generalization of voting. In case of using the class probability values from the multiple models, the most straightforward way to combine them is to compute their average.h In case of having likelihood values rather than probabilities, the likelihood values must be normalized to similar ranges before combining. Different combination methods: sum rule, product rule, max rule, min rule, median rule and majority vote for class likelihood values are compared by Kittler et al. (1998).

Bagging and boosting are principles to build multiple classifiers for a same task to perform shared decision making. Boosting is rooted in the idea of building a strong classifier from multiple weak classifiers. Each weak classifier may be only slightly better than random guessing, while a combination of a multitude of them together should be able to make arbitrarily accurate classification. The classifier function to be used for each individual classifier within the boosting combination may be selected freely. The multitude of different classifiers with the same training data is obtained by setting different weights for data samples when training each of the classifiers. The most widely known boosting algorithm is AdaBoost introduced in Freund and Schapire (1997b). AdaBoost concurrently trains multiple classifiers and a weighted average function for combining their decisions. When training each classifier for the combination, AdaBoost algorithm weighs the training samples such that each new classifier emphasizes on not-yet-so-well-classifying area of the input space.

Bagging, i.e. bootstrap aggregating, is a principle to pick different data samples for training each of the models or classifiers. For training one classifier for an ensemble, a subset of size $N^{\#} \leq N$ is picked randomly with replacement from the entire training data set of N samples. Thus bagging is a special case of boosting, where the data weights are integers, usually 0 or 1. For the final decision making, outputs of the multiple models are averaged, or the multiple classifiers vote for the category. Theoretical analysis of the effect of the sample size $N^{\#}$ and the number of classifiers trained for the combination are discussed by Fumera et al. (2008). A popular Random Forest (Breiman (2001)) classifier uses bagging principle to pick randomly both the attributes and the samples to use for training multiple decision trees. The final classification of a random forest is obtained by voting.

Boolean combinations

Boolean operators are used for combining multiple detectors, each of which outputs either *true* or *false* denoting whether the quested phenomenon is detected or not. A Boolean combination of detectors may utilize the Boolean operators for intersection (**AND**, $\wedge$), union (**OR**, $\vee$) and negation (**NOT** , $\neg$) on detector outputs. Boolean combinations have a close relationship to binary decision trees, as any Boolean function may be expressed as a decision tree and vice versa.

At simplest, classification trees – deducible to Boolean functions – are constructed based on nominal attributes, i.e. features x. In this case the detectors of the resulting Boolean function are of the form

$$\mathrm{d}(x) = \begin{cases} \textit{true} & \text{if } x \text{ equals the selected nominal value} \\ \textit{false} & \text{else.} \end{cases} \tag{4.13}$$

However, continuous valued features may be used by the aid of a threshold θ, which is used to binarize the feature. A continuous valued feature x results in multiple detectors of the form

$$\mathrm{d}(x;\theta) = \begin{cases} \textit{true} & \text{if } x \geq \theta \\ \textit{false} & \text{else.} \end{cases} \tag{4.14}$$

This kind of binarization scheme proposed in Boros et al. (1997) was used for class likelihood evaluation and Boolean function learning in Publications IV and VI.

Finding the best possible Boolean function for a space partition is an NP-complete problem, thus iterative algorithms based on different heuristics are utilized. Related, but slightly easier, is the problem of finding thresholds for continuous valued attributes, when the form of the Boolean function is predefined. Even in this case, the search space is unfeasibly large and gradient descent type algorithms likely fail due to the discontinuous nature of the error surface of a non-differentiable Boolean function.

Among the classification tree learning algorithms, the ID3 -algorithm has been developed for utilizing nominal valued data, whereas the C4.5 and CART algorithms have the capability of handling continuous valued attributes also. Algorithms finding an input space partition which perfectly classifies the training data are e.g. A^q (Michalski (1983)) and LAD (Hammer (1986)) -based algorithms and the OCAT-RA1 -algorithm (Deshpande and Triantaphyllou (1998)). The A^q and LAD-based algorithms iteratively find conjunctions of the form $\bigwedge_i \mathrm{d}_i(x_i)$ to be combined disjunctively, for building a Boolean function in disjunctive normal form (DNF) $B = \bigvee_j \bigwedge_i \mathrm{d}_{ij}(x_{ij})$. The OCAT-RA1 algorithms iteratively finds disjunctions $\bigvee_i \mathrm{d}_i(x_i)$ of a CNF (conjunctive normal form) Boolean function $B = \bigwedge_j \bigvee_i \mathrm{d}_{ij}(x_{ij})$.

Algorithms specifically designed for combining threshold tunable detectors and utilized as reference algorithms in Publication VI are the Boolean Algebra of ROC (receiver operating characteristic) curves, proposed in Oxley et al. (2007), and iterative exhaustive search algorithm, proposed in Tao and Veldhuis (2009). These algorithms aim at finding thresholds for available tunable detectors $\mathrm{d}_i(x_i;\theta_i),\ i=1...I$ when combined within a predefined Boolean function, which in the original setting is either disjunctive $B_I = \bigvee_{i=1}^{I} \mathrm{d}_i(x_i;\theta_i)$ or conjunctive $B_I = \bigwedge_{i=1}^{I} \mathrm{d}_i(x_i;\theta_i)$. At each iteration i of the Boolean algebra of ROC curves -algorithm, thresholds for the new combination B_i is selected based on the true positive and false positive rates (*tpr* and *fpr*) of the ROC curves of B_{i-1} and d_i. The method is computationally very efficient, but the drawback is its implicit assumption that all the component detectors are conditionally independent of each other. The iterative exhaustive search -algorithm similarly, using a predefined Boolean operator, iteratively adds new component detectors into B. For combining d_{i+1} into B_i an exhaustive search over combinations of all the possible operating points on the ROC curves of B_i and d_{i+1} is performed.

In the Publication VI a BOATS algorithm has been proposed for finding thresholds for a DNF Boolean function. The proposed algorithm also tunes the predefined form of the function for best performance. The algorithm is shortly described in the next subsection. For evaluation of the BOATS algorithm in Publication VI, the two above mentioned algorithms are implemented for comparison such that they facilitate learning of a DNF Boolean function.

4.4 Boolean OR of ANDs detector combination

In Publication VI I present a DNF Boolean **OR** of **AND**s (BOA) detector combination and the BOATS algorithm to set its parameters. The BOA is built of unrestricted number of detectors on M different modeling functions $\mathrm{f}_m(\boldsymbol{x}_m),\ m = 1...M$ for detecting a target phenomenon. Each function utilizes its own input feature set $\boldsymbol{x}_m$ and outputs a target likelihood score l_m, that is $\mathrm{f}_m(\boldsymbol{x}_m) = l_m$, which represents target probability as its monotone transformation. The target likelihood score l_m is compared to a threshold value θ_m, which defines a detector

$$\mathrm{d}_m(\boldsymbol{x};\theta_m) = \begin{cases} \textit{true}, & \text{if } \mathrm{f}_m(\boldsymbol{x}) \geq \theta_m \\ \textit{false}, & \text{otherwise.} \end{cases} \tag{4.15}$$

This detector is sensitivity tunable, that is, by changing the value of threshold θ, the detector operates at different operation points. Since the BOA combination is a Boolean function in disjunctive normal form, individual detectors are evaluated within *conjunctions* of the BOA function, which are then joined together as a disjunction. That is, within a BOA combination, multiple *conjunctions* of the form $\bigwedge_{m \in z} \mathrm{d}_m(\boldsymbol{x};\theta_m)$, where some detectors $\mathrm{d}_m(\boldsymbol{x};\theta_m)$, $m \in z \subseteq \{1,...,M\}$ are combined with **AND** operators, are primarily utilized. The conjunctions are joined together disjunctively with **OR** operators to form the BOA combination as

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{q=1}^{Q} \bigvee_{n=1}^{N_q} \left[\bigwedge_{i=1}^{M_q} \mathrm{d}_{z_q(i)}\left(\boldsymbol{x};\theta^{q,n}_{z_q(i)}\right) \right], \tag{4.16}$$

where the detector function identifiers m of d_m are given within lists $z_q,\ q = 1...Q$ of M_q identifiers each, e.g. if $z_1 = (1), z_2 = (1,2)$ and $z_3 = (1,3)$, $M_1 = 1, M_2 = 2$ and $M_3 = 2$. The number of conjunctions over the same list z_q is denoted with N_q. Every conjunction, enumerated by $q{=}1...Q$ and $n{=}1...N_q$, operates with a distinct set of M_q thresholds $\{\theta^{q,n}_m \mid i{=}1...M_q,\ m = z_q(i)\}$. Thus the operating point $\boldsymbol{\theta}$ of the BOA function (4.16) stands for $\sum_{q=1}^{Q} N_q \cdot M_q$ thresholds, one for each detector instance within the BOA.

Ideally, a – possible infinite – BOA combination has capability of producing any monotonic decision boundary for the two classes, the target and the clutter class. The decision boundary of a finite BOA combination is monotonic and piecewise linear in the space $(l_1, l_2, ..., l_M)$ of target class likelihoods. An example of a BOA decision boundary is shown in Figure 4.5.

The BOA combination is sensitivity tunable, as values of the thresholds $\theta^{q,n}_m$, $q{=}1...Q,\ n{=}1...N_q,\ m{=}z_q(i), i{=}1...M_q$ in a set $\boldsymbol{\theta}$ may be changed to vary the detection behaviour of the BOA combination. Below, an algorithm is presented to find sets $\boldsymbol{\theta}_\alpha$ of threshold values for a BOA combination to account for different sensitivity levels α. That is, the training provides the user with one sensitivity parameter α such that the BOA combination $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}_\alpha)$ may be denoted as $\mathrm{B}(\boldsymbol{x};\alpha)$.

For the cascade interpretation of the BOA function, which will be explained in Section 5.3, the negation $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ of the BOA function in disjunctive normal form is needed. The $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ in the disjunctive normal form has $K = \prod_{q=1}^{Q} M_q^{N_q}$ conjunctions, each of which contains $\sum_{q=1}^{Q} N_q$ detectors. That is, for each conjunction of $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ in the disjunctive normal form, one detector is picked from each conjunction (q,n) of the corresponding BOA $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$. Using a detector indexing function

$$\mathcal{I}(k,q,n) = \left\lfloor \frac{\left\lfloor \dfrac{k-1}{\prod_{i=q+1}^{Q} M_i^{N_i}} \right\rfloor}{M_q^{N_q-n}} \right\rfloor \bmod M_q \quad + 1, \tag{4.17}$$

$\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ in disjunctive normal form is given by

$$\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{k=1}^{K} \left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \neg\mathrm{d}_{z_q(\mathcal{I}(k,q,n))}\left(\boldsymbol{x};\theta^{q,n}_{z_q(\mathcal{I}(k,q,n))}\right) \right]. \tag{4.18}$$

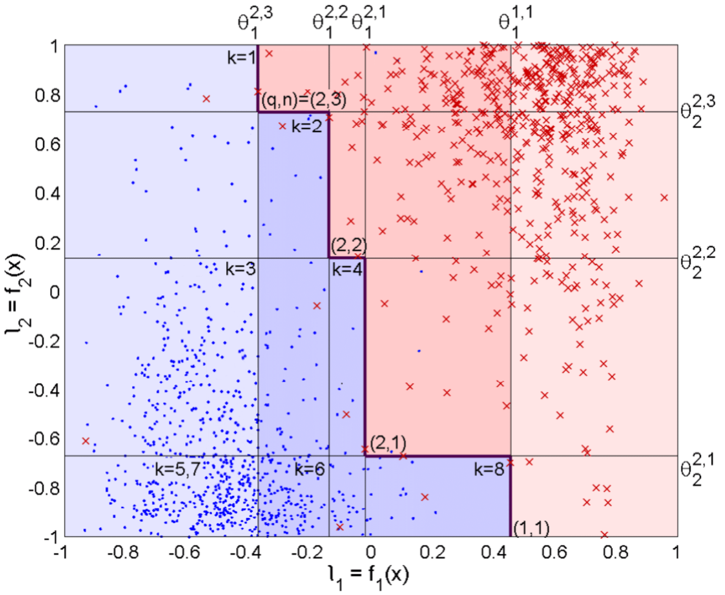


Figure 4.5: Example of data partition with a BOA detector. Data is represented in terms of two scores l_1 and l_2, which represent the likelihood of target class. Red crosses denote data samples from the target class, and samples representing the non-target class are shown with blue dots. The angular decision boundary of BOA combination $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathrm{d}_1(x_1;\theta_1^{1,1}) \vee \bigvee_{n=1}^{3}\left[\mathrm{d}_1(x_1;\theta_1^{2,n}) \wedge \mathrm{d}_2(x_2;\theta_2^{2,n})\right]$ is shown with the bold line. Each individual detector threshold $\theta_m^{q,n}$, $m=1,2,\ q=1,2,\ n=1,2,3$ is illustrated with a thin line. The detector score space where $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = $ *true* is colored with red background, and the space where $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})) = $ *true* is colored with blue background. The pale red and pale blue backgrounds illustrate the subspaces, where the decision is done based on l_1, using the model function $\mathbf{f}_1$ only.

Possibly more comprehensible notation for the $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ in the disjunctive normal form is given by using multiple OR-operators, each of them referring to one conjunction (q,n) of B as

$$\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \underbrace{\bigvee_{i_{1,1}=1}^{M_1}\ \bigvee_{i_{1,2}=1}^{M_1}\ \bigvee_{i_{1,3}=1}^{M_1} \cdots \bigvee_{i_{1,N_1}=1}^{M_1}}_{N_1 \bigvee\text{-operators , i.e. } M_1^{N_1}\text{ conjunctions}}\ \underbrace{\bigvee_{i_{2,1}=1}^{M_2}\ \bigvee_{i_{2,2}=1}^{M_2} \cdots \bigvee_{i_{2,N_2}=1}^{M_2}}_{N_2 \bigvee\text{-operators, i.e. } M_2^{N_2}\text{ conjunctions}} \cdots$$

$$\cdots \underbrace{\bigvee_{i_{Q,1}=1}^{M_Q}\ \bigvee_{i_{Q,2}=1}^{M_Q} \cdots \bigvee_{i_{Q,N_Q}=1}^{M_Q}}_{N_Q \bigvee\text{-operators, i.e. } M_Q^{N_Q}\text{ conjunctions}} \left[\ \bigwedge_{q=1}^{Q}\ \bigwedge_{n=1}^{N_q} \neg\mathbf{d}_{z_q(i_{q,n})}\left(\boldsymbol{x},\theta^{q,n}_{z_q(i_{q,n})}\right)\right]. \tag{4.19}$$

Now the indices $i_{q,n}$, which enumerate the disjunctions of $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ directly give the detector identifier index $i_{q,n}$ from a list z_q.

An algorithm to train a BOA combination

In Publication VI I have presented an algorithm, which finds sets of thresholds to be used within any Boolean combination of BOA form. The algorithm finds unique sets

$\boldsymbol{\theta}_\alpha = \{\ \theta_m^{q;n} \mid q = 1...Q,\ \ n = 1...N_q,\ \ m = z_q(i), i{=}1...M_q\}$ of thresholds for a set of sensitivity values $\alpha \in [0...1]$ of BOA. The trained BOA combination may now be referred to as $\mathrm{B}(\boldsymbol{x}, \alpha) \equiv \mathrm{B}(\boldsymbol{x}, \boldsymbol{\theta}_\alpha)$. The details of the algorithm can be found in the Publication VI, but here I give a brief description of it.

The algorithm starts by setting all the BOA thresholds in $\boldsymbol{\theta}$ of (4.16) to infinity, to account for the setting $\mathrm{B}(\boldsymbol{x}, 0)$ where $\alpha = 0$, and the BOA function does not *accept* any samples, that is $\mathrm{B}(\boldsymbol{x}, 0) = \textit{false}\ \ \forall \boldsymbol{x}$. Then the thresholds are mitigated such that only one (if possible) training sample which represents the target class will be accepted, and as few false positives as possible will be accepted along. The newly obtained set of thresholds $\boldsymbol{\theta}_\alpha$ is saved to account for $\alpha = t/T$, where t, is the number of accepted samples out of T training samples representing the target category. The algorithm continues diminishing the thresholds of the BOA step-wise, such that at every iteration exactly one, if possible, new training sample from target category will be accepted by the BOA, until the thresholds are at a level accepting all the target category samples of the training set. At each iteration, the found set $\boldsymbol{\theta}_\alpha$ of BOA thresholds is saved to account for the BOA operating point $\alpha = t/T$, where t is the number of accepted training samples from the target category at that operating point. The training algorithm thus produces parameters for a BOA to form a continuum $\alpha \in [0, 1]0, 1$ of operating points which account for true positive rates from *tpr* $= 0$ to *tpr* $= 1$ on the training data.

5 Sequential classification

Sequential decision making means that the decision making process is step-wise. The idea is that after each step of the process there are multiple possible ways to continue, and the operation done at next processing step is defined by the result of an operation at the current step. A sequential decision making process may be illustrated as a flowchart similar to a decision tree. The blocks of the flowchart, corresponding the nodes of a decision tree, consist of functions, which make the decision about the next step or provide the output, i.e. final conclusion, which is denoted as a leaf in terms of a decision tree. However, while the classification trees commonly use univariate functions at each node, within the sequential decision making process any kind of decision making functions may be used.

The computational advantage of sequential classification strategy over the arithmetic classifier ensembles is, that all the features (attributes) might not always be needed for the classification. That is, for many input samples the sequential classification process may appear such that the classification is done by evaluating only few decision making functions based on only few attributes. The evaluation of the rest of the decision making functions in the system is thus avoided, and computation of some features (attributes) may be omitted.

Boolean classifier combinations utilize the Boolean operators **OR** ($\vee$), **AND** ($\wedge$), and **NOT** ($\neg$) for a combination function of multiple binary classifiers, i.e. detectors $\mathrm{d}(\boldsymbol{x})$. Every Boolean function may be expressed as a binary decision tree, whose step-wise evaluation possibly ends up with the classification before all the component detectors of the Boolean function have been evaluated. For example, with a Boolean combination $B = \mathrm{d}_1(\boldsymbol{x}) \vee \mathrm{d}_2(\boldsymbol{x})$ of two binary classifiers $\mathrm{d}_1(\boldsymbol{x})$ and $\mathrm{d}_2(\boldsymbol{x})$, algorithmically the result $B =$ *true* will be released as soon as the detector $\mathrm{d}_1(\boldsymbol{x})$ outputs *true*. Similarly, the Boolean combination $B = \mathrm{d}_1(\boldsymbol{x}) \wedge \mathrm{d}_2(\boldsymbol{x})$ is definite without evaluating $\mathrm{d}_2(\boldsymbol{x})$, if $\mathrm{d}_1(\boldsymbol{x}) =$ *false*.

In the following, the experiments and results of Publication IV on sequential classification process using Boolean combinations are discussed in Section 5.1. Then, in Section 5.2, the principle of a classification cascade as a sequential classification strategy is explained and literature on detection cascades is addressed. The cascade classification process of a BOA combination is explained in Section 5.3, and in Section 5.4, the experiments and results of Publication VI using a BOA cascade on laughter vs speech classification task are discussed.

5.1 Experiments on Boolean combinations for sequential decision making

In the Publication IV I have demonstrated how a sequential classification process with Boolean combinations reduce computational load of classification and improve the classification accuracy over the individual detectors. The sequential classification process is experimented on lifelog video material of our CASA2 dataset for video context change detection task. It contains over 7 hours of video data from 23 different types of environments. The videos have been shot with a small spy camera, and the stereo sound tracks are recorded by a pair of in-ear microphones. Video context change detection is a well studied problem having matured solutions. However, our experiments focused on examining the computational efficacy of sequential processing using two signal modalities rather than improving the state-of-the art solutions for video context change detection.

In the work I have utilized three methods, $\mathrm{f}_A, \mathrm{f}_{V1}$ and f_{V2}, to provide context change likelihood information from videos. The context change likelihoods are thresholded to obtain three sensitivity tunable detectors $\mathrm{d}_A(\boldsymbol{x};\theta), \mathrm{d}_{V1}(\boldsymbol{x};\theta)$ and $\mathrm{d}_{V2}(\boldsymbol{x};\theta)$, as in (4.15). One or multiple instances of the three sensitivity tunable detectors at different operating points θ are combined as a BOA combination. It should be noted that evaluating multiple detectors d_m based on the same context change likelihood method $\mathrm{f}_m(\boldsymbol{x})$ within a BOA combination is virtually free. This is due to that the heavy part of a detector evaluation is the computation of the likelihood $\mathrm{f}_m(\boldsymbol{x}) = l_m$, which is done when the first one of the detectors d_m is run, and may be then saved for late use.

One of the context change likelihood extraction methods operates on audio stream of the video, and two others operate on the image frames of the video. The method f_A based on the audio stream of the video is the fastest of the three, and it provides a context change likelihood $\mathrm{f}_A(\boldsymbol{x}_A) = l_A$ for all the moments of the video in 0.01 times real time on a desktop pc. The second method f_{V1}, providing context change likelihood $\mathrm{f}_{V1}(\boldsymbol{x}_V) = l_{V1}$ for each video frame, utilizes RGB histograms of video image frames. It takes 29 ms to extract the RGB-histogram from one video frame, that is 0.43 times real time for the videos with 15 frames per second, on the used desktop pc. The heaviest one of the used methods f_{V2}, which is based on SIFT BoW features, provides a context change likelihood $\mathrm{f}_{V2}(\boldsymbol{x}_V) = l_{V2}$ for each video frame. It takes 12.3 s, that is 184 times real time, to compute the SIFT BoW features for one video frame on the same desktop pc. Detailed descriptions of the methods can be found in Publication IV. The operating points, defined by the thresholds, of all the detectors within each of the tested Boolean combination are found utilizing a preliminary version of the BOATS algorithm, which was introduced in Section 4.4.

The most important results of Publication IV for video context change detection are presented in Table 5.1. The results are presented in terms of the F_1 score and the computation time in respect to real time, both averaged over the used videos. The achieved best performance is reported with each of the three detectors individually, as well as with some Boolean combinations of them.

Comparing results with Boolean **OR** ($\vee$) and **AND** ($\wedge$) combinations, we see that conjunctive combinations with $\wedge$ are far more computationally efficient than disjunctive combinations with $\vee$. With pairs $(\mathrm{d}_A, \mathrm{d}_{V1})$ and $(\mathrm{d}_A, \mathrm{d}_{V2})$ of detectors, the difference in classification speed with disjunctive and conjunctive combination is $0.44/0.02 = 22$ -fold and $184/0.3 = 613$ -fold, respectively. The difference in computational speed is due to that the class distribution is heavily unbalanced, the 'no context change' -class being

Table 5.1: The main results of Publication IV on efficacy of Boolean detector combinations in a video context change detection task. Detection results are given with the three used sensitivity tunable detectors alone and within different Boolean combinations. The results are expressed in terms of the best F_1 score and the computation time (CT) of detection in respect to real time of videos. Note that the thresholds θ^n_m are different for each BOA combination.

the Boolean combination	F_1	CT s/s
$\mathrm{B} = \mathrm{d_A}(\boldsymbol{x};\theta_A)$	.84	0.01
$\mathrm{B} = \mathrm{d_{V1}}(\boldsymbol{x};\theta_{V1})$	.40	0.43
$\mathrm{B} = \mathrm{d_{V2}}(\boldsymbol{x};\theta_{V2})$	.52	184.0
$\mathrm{B} = \mathrm{d_A}(\boldsymbol{x};\theta_A) \vee \mathrm{d_{V1}}(\boldsymbol{x};\theta_{V1})$	.85	0.44
$\mathrm{B} = \mathrm{d_A}(\boldsymbol{x};\theta_A) \vee \mathrm{d_{V2}}(\boldsymbol{x};\theta_{V2})$	.84	184.0
$\mathrm{B} = \mathrm{d_A}(\boldsymbol{x};\theta_A) \wedge \mathrm{d_{V1}}(\boldsymbol{x};\theta_{V1})$	.90	0.02
$\mathrm{B} = \mathrm{d_A}(\boldsymbol{x};\theta_A) \wedge \mathrm{d_{V2}}(\boldsymbol{x};\theta_{V2})$	.95	0.30
$\mathrm{B} = \mathrm{d_{V1}}(\boldsymbol{x};\theta_A) \wedge \mathrm{d_{V2}}(\boldsymbol{x};\theta_{V2})$	.68	1.30
$\mathrm{B} = \left[\mathrm{d_A}(\boldsymbol{x};\theta^1_A) \wedge \mathrm{d_{V1}}(\boldsymbol{x};\theta_{V1})\right] \vee \left[\mathrm{d_A}(\boldsymbol{x};\theta^2_A) \wedge \mathrm{d_{V2}}(\boldsymbol{x};\theta_{V2})\right]$	.96	0.30

the prevalent one. The conjunctive combination with $\wedge$ needs not evaluate the second detector if the first one already outputs *false*, while in the same situation the disjunctive combination with $\vee$ must run both the detectors. The notably higher F_1-scores with conjunctive combinations than disjunctive combinations suggest that the audio and visual modality provide complementary information especially about the 'no context change' -class. The complementary information comes out of the detectors at operating points with low specificity level, which is used for detectors within a conjunctive combination. That is, within a conjunctive conjunction the detectors are tuned for higher sensitivity and lower specificity than if used alone. The low specificity level of detectors within a combination is compensated by the AND ($\wedge$) -operator, if the incorrect decisions of the detectors to the 'context change' -class do not correlate.

The best performance of $F_1 = 0.96$ in our experiments was achieved with combination $(\mathrm{d_A} \wedge \mathrm{d_{V1}}) \vee (\mathrm{d_A} \wedge \mathrm{d_{V2}})$. Combining the terms $\mathrm{d_A} \wedge \mathrm{d_{V1}}$ and $\mathrm{d_A} \wedge \mathrm{d_{V2}}$ disjunctively allows setting their operating points such that they are slightly more reluctant to output *true* (for 'context change' detected) than at the best operating point for each term individually. That is, within the combination the two terms may operate with higher specificity than alone. Combining their uncorrelated decisions disjunctively gives a chance to exploit the diverse information given by l_{V_1} and l_{V2} about the 'context change' -class and compensate for the reduced sensitivity of the terms. The computational speed of the Boolean detector combination $(\mathrm{d_A} \wedge \mathrm{d_{V1}}) \vee (\mathrm{d_A} \wedge \mathrm{d_{V2}})$, when evaluated sequentially is $(184.0 + 0.43 + 0.01)/0.3 \approx 615$ times faster than by evaluating all the methods for every moment of the video.

5.2 Decision cascades for classification

Decision cascade is a sequential decision process, which may be represented as a decision tree consisting of a single branch with leaves. The difference between a cascade and sequential decision making process in general is, that within a cascade the decision maker functions responsible for the process are evaluated always in the same order, independent

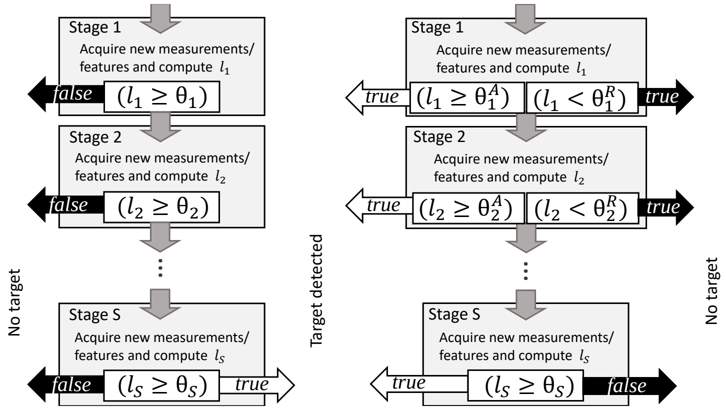

Stage 1
Acquire new measurements/ features and compute l_1
false
$(l_1 \geq \theta_1)$
Stage 2
Acquire new measurements/ features and compute l_2
false
$(l_2 \geq \theta_2)$
Stage S
Acquire new measurements/ features and compute l_S
false
$(l_S \geq \theta_S)$
true
No target
Target detected
Stage 1
Acquire new measurements/ features and compute l_1
true
$(l_1 \geq \theta_1^A)$
$(l_1 < \theta_1^R)$
true
Stage 2
Acquire new measurements/ features and compute l_2
true
$(l_2 \geq \theta_2^A)$
$(l_2 < \theta_2^R)$
true
Stage S
Acquire new measurements/ features and compute l_S
true
$(l_S \geq \theta_S)$
false
No target

Figure 5.1: A one sided (left) and symmetrical (right) cascade structures for efficient subfigure classification in computer vision applications.

of the input characteristics. A classification cascade thus consists of multiple processing steps, i.e. *stages*, where at each entered stage a decision about releasing a classification or entering the next stage is made.

Detection cascades have been investigated mostly in the field of machine vision starting from the work of Feraund et al. (2001) and Viola and Jones (2001b). Here, a term detection cascade is utilized in the place of a classification cascade to emphasize the task being binary classification into a 'target' class and a 'clutter' class with highly imbalanced class distribution. The primary goal in cascaded decision making for computer vision applications is in reducing computational load. The heavily imbalanced class distributions in object detection from images, as most of the search windows of different sizes and positions do not contain the target object, offers great possibilities to make decisions with minor examination. The cascading schemes are designed such that gradually more and more rigorous classifiers are imposed on each search window. The most popular application areas of detection cascades have been face detection (e.g. Li et al. (2015)) and pedestrian detection (e.g. Shen et al. (2013)).

To design an object detection cascade for computer vision applications there is close to infinite pool of image features available, e.g. Haar, HOG, etc. to utilize. The dominant approach is to utilize Haar features selected by the AdaBoost algorithm. Usually at each stage s of the cascade a linear function over some selected features is utilized to provide a class likelihood l_s for the 'target' class, and a threshold value is used to make the decisions, e.g. as in Viola and Jones (2001a).

The first object detection cascades proposed were one-sided, as shown in Figure 5.1 (left). At each stage they make the decision either to classify the figure patch as 'non-target' if $(l_s \geq \theta_s) =$ *false*, or else to enter the next stage. Thus early classification is possible for the 'non-target' class only,. Classification as 'target' is possible to be made only at the last cascade stage if $(l_S \geq \theta_S) =$ *true*.

A symmetrical object detection cascade, shown in Figure 5.1 (right), has an option to make the classification to both the categories at every stage. This kind of decision cascade was first introduced by Sochman and Matas (2005). Symmetrical cascade offers more computational reduction than the one-sided cascade and it is suitable also for classification problems with balanced class distributions. The computer vision applications presented in the literature utilize a single target likelihood value l_s similarly to the one-sided cascades. Decisions at each stage are made using two threshold values, one for classification to the 'target' class, i.e. for accepting, if $(l_s \geq \theta_s^A) = \mathit{true}$ and another for classification to the 'non-target' class, i.e. rejecting, if $(l_s < \theta_s^R) = \mathit{true}$. At the last cascade stage, only one threshold θ_S is used to enforce classification.

For training an object detection cascade, approaches with concurrent design of the cascade length S, selection of features $\boldsymbol{x}_s$ and linear functions $\mathrm{f}_s(\boldsymbol{x}) = l_s,\ s = 1...S$ on them as well as the thresholds θ_s^R and θ_s^A are proposed by Dundar and Bi (2007) and Saberian and Vasconcelos (2012).

5.3 BOA as a cascade of Boolean combinations

The BOA detector defined by Equations (4.16) and (4.18) may be formulated as a decision cascade, which offers computational advantage. As already discussed in the beginning of this chapter, a Boolean function of different detectors may be evaluated sequentially. However, if a certain detector $\mathrm{d}_m(\boldsymbol{x};\theta)$, based on a certain target likelihood function $\mathrm{f}_m(\boldsymbol{x}) = l_m$, is utilized in many of the BOA conjunctions, this principle is not enough to evaluate the computational efficiency of a BOA. This is due to that within a detector $\mathrm{d}_m(\boldsymbol{x};\theta)$ it is costly to compute the target class likelihood l_m with a function $\mathrm{f}_m(\boldsymbol{x}) = l_m$, but the cost of comparing this value to a threshold θ is marginal. The computational efficiency of a BOA combination is thus affected by the fully dependent, concurrent decisions of all the detectors $\mathrm{d}_m(\boldsymbol{x};\theta)$ with different values of θ. For a BOA evaluation, the computationally heavy part is the target likelihood estimation via M utilized functions $\mathrm{f}_1, \mathrm{f}_2, ..., \mathrm{f}_M$, while the computational load of a threshold comparisons within detectors may be considered negligible. The interpretation of a BOA function as a cascade is thus founded on step-wise target likelihood computation, and the number S of stages in the BOA cascade equals the number of target likelihood extraction functions, that is $S = M$. To simplify notation in the following, we enumerate the functions $\mathrm{f}_1, \mathrm{f}_2, ..., \mathrm{f}_M$ used within the BOA such that the function run at stage s is denoted as f_s.

To formulate a BOA cascade mathematically, we utilize binary functions B_s^1 and B_s^0 for stages $s = 1...S$. The functions B_s^1 account for classification to the 'target' class and the functions B_s^0 are responsible of classification to 'non-target' class at each stage of the cascade. The functions $B_s^1,\ s = 1...S$ are partitions of the BOA function (4.16) such that $\mathrm{B} = B_1^1 \vee B_2^1 \vee ... \vee B_S^1$. Similarly the functions $B_s^0,\ s = 1...S$ are partitions of the negation (4.18) of the BOA function such that $\neg\mathrm{B} = B_1^0 \vee B_2^0 \vee ... \vee B_S^0$. Since the BOA functions (4.16) and (4.18) give a non-overlapping partition of the target likelihood space $[l_1, l_2, ..., l_S]$, at each stage s it is possible for only another one of the functions B_s^1 and B_s^0 to output *true*. This means that the decision at stage s of a BOA cascade can be made as

$$\begin{cases} \text{if} & B_s^1(\boldsymbol{x}) = \mathit{true}, & \text{classify } \boldsymbol{x} \text{ as 'target' and stop,} \\ \text{else if} & B_s^0(\boldsymbol{x}) = \mathit{true}, & \text{classify } \boldsymbol{x} \text{ as 'non-target' and stop,} \\ \text{else} & & \text{enter stage } s+1. \end{cases} \tag{5.1}$$

It also means that the BOA cascade classification is consistent. That is, if the classification is made by B_s^1 or B_s^0 at a cascade stage s, the decision makers B_r^1 and B_r^0 of the other stages $r = 1...S,\ r \neq s$ would not make contradicting classifications.

The functions B_s^1 and B_s^2 at stage s of a BOA cascade utilize all the target likelihoods $l_1, l_2, ..., l_s$ computed so far, thus the functions B_s^1 and B_s^2 being partitions of (4.16) and

(4.18), are defined by the target likelihood function identifier lists $z_q,\ q=1...Q$ of the BOA. The decision makers $B_s^1,\ s=1...S$ for the 'target' class are partitions of (4.16) as

$$B_s^1(\boldsymbol{x};\alpha) = \bigvee_{z_q \text{ such that } \substack{\exists j \text{ s.t. } s = z_q(j) \\ \nexists j \text{ s.t. } r = z_q(j),\, r>s}} \bigvee_{n=1}^{N_q} \bigwedge_{i=1}^{M_q} \mathrm{d}_{z_q(i)}\left(\boldsymbol{x};\theta^{q,n}_{z_q(i)}\right). \tag{5.2}$$

That is, B_s^1 contains those conjunctions (q,n) of the BOA (4.16) which utilize the newly computed likelihood score l_s and possibly the likelihoods $l_1,...l_{s-1}$ computed at earlier stages, but none of the target likelihoods $l_r,\ r>s$.

Similarly, the internal decision makers $B_s^0,\ s=1...S$ of the BOA cascade for the 'non-target' class are partitioned from the disjunctive normal form (4.18) of the negated BOA. That is, each B_s^0 contains the conjunctions k of the BOA (4.18) that utilize the newly computed likelihood score l_s and possibly those computed at earlier stages, but none of the scores $l_m,\ m>s$ as

$$B_s^0 = \bigvee_{k \text{ such that } \substack{k\in\{1...K\}\\ I_s(k)=\textit{true}}} \left[\bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q} \neg\mathrm{d}_{z_q(\mathcal{I}(k,q,n))}\left(\boldsymbol{x};\theta^{q,n}_{z_q(\mathcal{I}(k,q,n))}\right)\right], \tag{5.3}$$

where the included conjunctions of (4.18) are given via a recursively defined Boolean function $I_s(k)$

$$\begin{aligned} I_0(k) &= \textit{false} \quad \forall\, k=1...K \\ I_s(k) &= \bigwedge_{r=0}^{s-1} \neg I_r(k) \wedge \bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q}\bigwedge_{m=s+1}^{S} \left[\, z_q(\mathcal{I}(k,q,n)) \neq m\,\right]. \end{aligned} \tag{5.4}$$

The first part of the equation for $I_s(k)$ makes sure that the conjunction k has not been used for $B_r^0,\ r<s$. The rest of the equation sets $I_s(k)=\textit{false}$ if any of the target likelihoods $l_{s+1}, l_{s+2}, ..., l_S$, are utilized in the conjunction k of (4.18). Using the alternative notation (4.19) of $\neg\mathrm{B}$, the decision makers $B_s^0,\ s=1...S$ for the 'non-target' class may be written as

$$\begin{aligned} B_s^0 = &\bigvee_{\substack{i_{1,1}=1...M_1\\ z_1(i_{1,1})\le s}} \bigvee_{\substack{i_{1,2}=1...M_1\\ z_1(i_{1,2})\le s}} \cdots \bigvee_{\substack{i_{1,N_1}=1...M_1\\ z_1(i_{1,N_1})\le s}} \bigvee_{\substack{i_{2,1}=1...M_2\\ z_2(i_{2,1})\le s}} \cdots \bigvee_{\substack{i_{2,N_2}=1...M_2\\ z_2(i_{2,N_2})\le s}} \cdots \\ &\bigvee_{\substack{i_{Q,1}=1...M_Q\\ z_Q(i_{Q,1})\le s}} \cdots \bigvee_{\substack{i_{Q,N_Q}=1...M_Q\\ z_Q(i_{Q,N_Q})\le s}} \left[\bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q} \neg\mathrm{d}_{z_q(i_{q,n})}\left(\boldsymbol{x};\theta^{q,n}_{z_q(i_{q,n})}\right)\right]. \end{aligned} \tag{5.5}$$

This notation, while possibly being more comprehensible, includes all the decision makers $B_r^0,\ r<s$ in B_s^0, however this redundancy does not affect the functionality. Examples of partitions of different BOA functions are given in the next section.

To achieve a specific type of cascade from a BOA combination, e.g. one-sided or symmetrical cascade, the target likelihood extraction method indices listed in $z_q,\ q=1...Q$ for the BOA must be selected appropriately. In detection problems, it is often the case that the class distribution is clearly unbalanced, the 'clutter' class being the prevalent one and 'target' class samples being rare. In this case, for a decision cascade to be computationally efficient, it should specifically be able to make early decisions to the prevalent 'clutter' class. A one-sided BOA cascade, which is able to make early detections to the 'clutter' class, actualizes by utilizing only single conjunction which combines functions on all the target likelihoods according to conjunction list $z_1 = (1, 2, 3, ..., S)$. On the other hand, a one-sided BOA cascade capable of making early detections to 'target' class actualizes if conjunction lists of every single target class model are included in the set of conjunction lists of the BOA, i.e. $\{(1), (2), (3), ..., (S)\} \subseteq \{\, z_q \,|\, q = 1...Q\}$. Symmetrical BOA cascade actualizes using $Q = S$ conjunction lists $z_1 = (1), z_2 = (1, 2), z_3 = (1, 2, 3), ..., z_Q = (1, 2, 3, ..., S)$.

5.4 Laughter detection with a BOA cascade

In Publication VI I have utilized a BOA cascade framework for classifying video clips of MAHNOB Laughter dataset Petridis et al. (2015) to those which contain laughter, and to those that do not contain laughter. I have used two models – or detectors – of laughter for the task. The detectors are built to mimic those used in Petridis et al. (2015) for the same task as closely as possible. The first detector operates on audio stream of the video and provides a laughter likelihood score l_1 for the video clip. The detector computes MFCC features from the audio stream of a video and evaluates a single-output feed-forward NN. The second detector provides the laughter likelihood score l_2 based on the image stream of the video. It finds 20 face points with the algorithm of Zhu and Ramanan (2012), reduces the feature dimensionality with PCA and provides a laughter likelihood score l_2 using a feed-forward NN. Details of the two detectors can be found in Publication VI.

The class distribution in this task is nearly balanced, so any cascade type likely results in reduction of computational load. Thus BOA combinations with all the possible conjunction list configurations have been evaluated in the experiments for Publication VI.

The best results were obtained with a BOA cascade

$$\mathrm{B}_{\mathfrak{C}}(\boldsymbol{x}; \alpha) = \mathrm{d}_1(\boldsymbol{x}; \theta_1^1) \vee \bigvee_{n=1}^{N} \Big[\, \mathrm{d}_1(\boldsymbol{x}; \theta_1^{2,n}) \wedge \mathrm{d}_2(\boldsymbol{x}; \theta_2^{2,n}) \, \Big] \tag{5.6}$$

which is built with $z_1 = [1]$ and $z_2 = [1, 2]$ and N selected by the BOATS algorithm. The

Table 5.2: Results obtained with a BOA cascade $\mathrm{B}_{\mathfrak{C}}$ of (5.6) in comparison to results found in the literature in laughter vs speech classification on MAHNOB laughter data. The used measures of classifier performance are the overall accuracy, F_1 -scores for both speech (F_1^{sp}) and laughter (F_1^{lg}), and percentage of computed visual features (**v.f.**). The BOA detectors are used at the operating point α with the highest accuracy. *) The classifier of Petridis et al. (2015) has been trained with another dataset. **) Results of Rao et al. (2015) are with 15 speakers while the other authors use 22 speakers in their tests.

	acc.	$F_1^{\mathbf{sp}}$	$F_1^{\mathbf{lg}}$	**v.f. %**
BOA cascade of $\mathrm{B}_{\mathfrak{C}}$, (5.6) $N = 1$	96.0	.966	.958	11%
BOA cascade of $\mathrm{B}_{\mathfrak{C}}$, (5.6) N by BOATS	96.9	.972	.955	33%
Rudovic et al. (2013)	92.7	.943	.905	100%
Petridis et al. (2015)*	91.7	.932	.893	100%
Rao et al. (2015)**	96.9	.973	.963	100%

decisions at the two stages of the resulting cascade are made according to

$$\begin{aligned}
B_1^{\text{laughter}} &= \mathrm{d}_1(\boldsymbol{x};\theta_1^1) \\
B_1^{\text{speech}} &= \neg\mathrm{d}_1(\boldsymbol{x};\theta_1^1) \wedge \bigwedge_{n=1}^{N} \neg\mathrm{d}_1(\boldsymbol{x};\theta_1^{2,n}) \qquad (5.7) \\
B_2^{\text{laughter}} &= \bigvee_{n=1}^{N} \Big[\, \mathrm{d}_1(\boldsymbol{x};\theta_1^{2,n}) \wedge \mathrm{d}_2(\boldsymbol{x};\theta_2^{2,n})\Big] \\
B_2^{\text{speech}} &= \bigvee_{k=2}^{2^N} \left[\; \neg\mathrm{d}_1(\boldsymbol{x};\theta_1^1) \wedge \bigwedge_{n=1}^{N} \neg\mathrm{d}_{z_2(j)}(\boldsymbol{x};\theta_{z_2(j)}^{q,n}\ ) \;\right] \\
&= \bigvee_{i_1=1}^{2} \bigvee_{i_2=1}^{2} \cdots \bigvee_{i_N=1}^{2} \Big[\neg\mathrm{d}_1(\boldsymbol{x};\theta_1^1) \wedge \neg\mathrm{d}_{z_2(i_1)}(\boldsymbol{x};\theta_{z_2(i_1)}^{2,1}) \wedge \neg\mathrm{d}_1(\boldsymbol{x};\theta_{z_2(i_2)}^{2,2}) \cdots \\
&\qquad \cdots \wedge \neg\mathrm{d}_{z_2(i_3)}(\boldsymbol{x};\theta_{z_2(i_3)}^{2,2}) \wedge \cdots \wedge \neg\mathrm{d}_{z_2(i_N)}(\boldsymbol{x};\theta_{z_2(i_N)}^{2,N})\Big]
\end{aligned}$$

The classification performance of the BOA $\mathrm{B}_{\mathfrak{C}}$ cascade was evaluated in terms of resulting classification accuracy and computational load. The main results of Publication VI are reproduced in Table 5.2. The results show that a BOA cascade $\mathrm{B}_{\mathfrak{C}}$, trained with the developed BOATS algorithm, described in Section 4.4, outperforms the reference classifiers of Rudovic et al. (2013) and Petridis et al. (2015), while utilizing far less computational resources. The best classification accuracy was achieved with $\mathrm{B}_{\mathfrak{C}}$ with N selected by the BOATS algorithm.

6 Automatic Speech Recognition

Automatic speech recognition (ASR) has appeared to be extremely difficult task to perform. This is shown by the long history of research on the problem. Only during recent years, the capability of algorithms have reached the performance satisfactory for general use. In case of low background noise, small vocabulary systems like digit recognition frameworks, or user interfaces with few command words have performed satisfactorily for a while. Also dictation software, which transcribes the spoken sentences into text, have been successfully used in noiseless environments, e.g. medical doctor appointment room. Automatic speech recognition performance reaching human level has been recently reported by Xiong et al. (2017) using DNNs.

The difficulty of the ASR -task is due to large variability inherent in speech signal. There are multiple sources of uncertainty for the ASR task of decoding the spoken words from an acoustic signal. Different languages form totally different probability distributions of phone sequences that the speech may be formed of. Every person has a unique voice and there is great variation in the ways a certain sentence may be pronounced. The acoustical characteristics of the speaking environment varies hugely and there is often some background noise on top of the speech to be recognized.

In terms of Bayesian decision making, the estimated word sequence $\hat{w}$ from among all the possible word sequences W is given by a conditional probability

$$\hat{w} = \arg\max_{w \in W} P(w|X) = \arg\max_{w \in W} P(w)P(X|w) \tag{6.1}$$

in respect to the speech signal X. The standard procedure is to represent the probability $P(w|X)$ in terms of two models representing the speech phenomenon, producing the probability of the word sequence $P(w)$ and the probability of the signal in respect to the given sequence $P(X|w)$. The success of an ASR framework thus depends on how well the models manage in their function. However, since the probability distributions of different word sequences are broad and overlapping, even the smallest possible Bayes error is notable for ASR.

In the core of an ASR framework, there is a *language model* producing $P(w)$. The language model incorporates information about the vocabulary and grammar of the language that the ASR framework is built for. It also includes knowledge about different possibilities for pronunciation of words and phrases in the language. The language model, usually an n-gram model, mostly an HMM, is trained using written texts and pronunciation information in the language. Errors due to this model are of two kinds. First, it is very unlikely to learn an n-gram model incorporating all the possible use cases of the language. On the other hand, the more flexible the model is, the smaller discrimination capability among different word sequences it has.

The probability $P(X|w)$ within an ASR framework is given by an *acoustical model*. It informs the system about acoustic characteristics of each phone of speech. The acoustic model should handle the variability of the signal due to different voices, speech emphasis and acoustic conditions. The model is trained using recorded speech audio, and it has been shown by Raj et al. (2012) that if the training audio matches the acoustic conditions of the ASR-system use cases, the system performs notably better than if there is a mismatch between the two.

ASR systems are usually evaluated in terms of the word error rate (WER)

$$\text{WER} = \frac{S + D + I}{N}, \tag{6.2}$$

where N is the number of uttered words within the audio material, S is the number of misinterpreted words, D is the number of words unnoticed by the system, and I is the number of extra words within the transcription. The recognition accuracy acc. $= 1 - \text{WER}$ is also an often used performance indicator.

In case of some words being more important to recognize than others, weighted error rates may be used. Binary weighting of words is used for reporting *key word* error rate or accuracy. In the Publications II,I and V the performance of the proposed ASR frameworks are reported in terms of the key word accuracy.

6.1 The traditional ASR framework

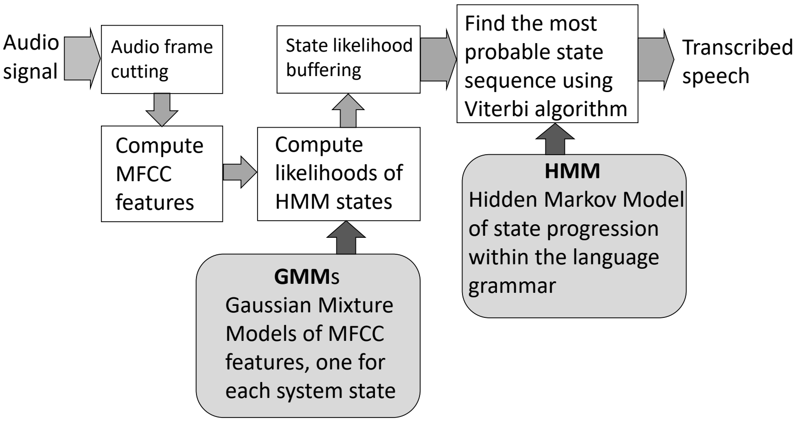

Audio signal
Audio frame cutting
Compute MFCC features
Compute likelihoods of HMM states
State likelihood buffering
Find the most probable state sequence using Viterbi algorithm
Transcribed speech
GMMs
Gaussian Mixture Models of MFCC features, one for each system state
HMM
Hidden Markov Model of state progression within the language grammar

Figure 6.1: A general framework, which is the basis of traditional style implementations for ASR. The framework operates on MFCC feature vectors from audio frames. It utilizes GMMs to model the distributions of feature vectors representative to each HMM state. The vocabulary and grammar are modeled with the transition probabilities between HMM states, and the best hypothesis for the speech transcription is found using the Viterbi algorithm.

A traditional ASR system, which is based on MFCC features, Gaussian mixture models of phonemes and hidden Markov n-gram models modeling the language structure, i.e. capturing the vocabulary and grammar, is depicted in Figure 6.1.

The audio signal is processed in frames, as explained in Chapter 2. The length of a smoothed frame is usually around 5-20 ms, the consecutive frames generally overlapping

50 %. Traditional GMM-HMM ASR frameworks mostly operate on MFCC features of audio frames. Usually only 13 low order MFCC coefficients are preserved to model the gross shape of the frequency spectral envelope. The MFCC coefficients and the MFCC delta and acceleration features are concatenated for the frame representation.

For ASR systems, speech is assumed to be based on phonemes. This means that automatic speech recognition implicitly contains a task of classifying the audio frames into phoneme based categories. Within ASR frameworks these categories are called states, which serve the similar function as classes in classification frameworks. The traditional ASR framework utilizes GMMs of MFCC feature distributions for this acoustic modeling, i.e. modeling each state. Combinations of 4-10 Gaussian probability distribution functions (PDF) are used to model the MFCC-feature variation among audio frames representing each phoneme based state. The GMM of each state is trained using training data representing the respective phoneme. For every new input audio frame, the trained GMMs are used to provide a state likelihood in respect of each phoneme based state of the system. The system state likelihoods from all the audio frames of the recording, or a batch of frames corresponding to a few seconds of the input signal, are buffered for language analysis.

In a traditional ASR framework the vocabulary and grammar of the language are modeled using HMMs. At the lowest level, phoneme, biphone (two consecutive phonemes) or triphone (three consecutive phonemes) HMMs are defined and trained to capture the probabilities of transitions between system states according to pronunciation of the used language. At the next level, the phoneme level HMMs are concatenated to form word level HMMs and the transition probabilities between phonemes within the word level HMM models are learned from the ways of pronunciation within the training material. Finally the word level HMMs are concatenated to form the language level HMM. The transition probabilities between all the pairs of words and established expressions are defined according to the language grammar, e.g. using representative text data of the language. It is noteworthy that the language model plays crucial part in the recognition capability of the ASR -framework.

To solve the utterance which the input signal most likely represents, the buffered state likelihoods are analyzed in respect to the generated hidden Markov model of the language. The most probable utterance is usually found using the Viterbi algorithm.

The performance of an ASR system is more or less equally defined by the success of the language model and the acoustic models. The quality and appropriateness of the language model builds a foundation for the recognition apparatus, and the fit and robustness of the acoustic model enables then accurate recognition. In Publications I, II and V the language model defined in The PASCAL 'CHiME' Speech Separation and Recognition Challenge Barker et al. (2013) baseline system is used unmodified, and the research has concentrated on improving the acoustical modeling part of the ASR framework. Thus in the following sections different methods for acoustic modeling of ASR are discussed.

6.2 Methods for state likelihood estimation for ASR

Within the above explained traditional ASR system, the acoustic models utilize MFCC features of the input audio. For each input feature vector, the system produces likelihoods for all the states within the HMM -model. This likelihood for each HMM state is obtained according to the expected feature distribution of the state, where the feature distribution

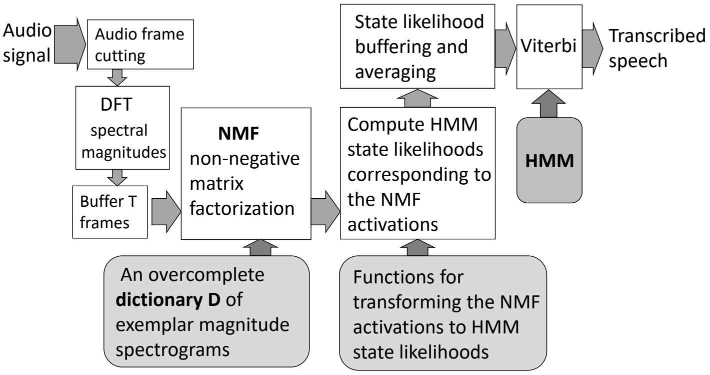

Audio signal
Audio frame cutting
DFT spectral magnitudes
Buffer T frames
NMF non-negative matrix factorization
An overcomplete dictionary D of exemplar magnitude spectrograms
Compute HMM state likelihoods corresponding to the NMF activations
Functions for transforming the NMF activations to HMM state likelihoods
State likelihood buffering and averaging
Viterbi
HMM
Transcribed speech

Figure 6.2: An ASR framework, which utilizes overdetermined dictionary based sparse feature vectors for acoustic modeling. DFT spectral magnitudes are often used as features, and the dictionary exemplars represent events in features spanning over T consecutive frames.

is expressed as a GMM -model. Despite the great expressive power and performance of GMM-models in studio conditions, the relentless problem is their lack of robustness against noise and mismatch in acoustic conditions of training data and the real use. Lots of methods have been proposed for adapting GMM-models for different speakers, different room acoustic conditions, different kind of background noises etc. . However, the problem has prevailed over the 30 years of ASR research, and thus some alternatives for MFCC and GMM based acoustic modeling have been tried out. It has not been easy to come up with a new paradigm to the matured field of ASR, but at least couple of new alternatives have broken their way through. The two alternative *hybrid* approaches that I will discuss below are the non-negative matrix factorization based and deep neural network based HMM state likelihood estimation.

6.2.1 Acoustic modeling based on non-negative matrix factorization

One alternative for GMM based HMM state likelihood estimation is NMF-based acoustic modeling. This approach has been utilized in Publications II and I, and the general framework structure is illustrated in Figure 6.2. The approach provides noise robustness and it has been successfully utilized for limited vocabulary tasks in noisy conditions in Publications II, I and V among many others. The idea in NMF processing is to represent input signal features $\boldsymbol{x}$ in terms of *exemplars* within a dictionary $\mathbf{D}$ as

$$\boldsymbol{x} \approx \mathbf{D}\mathbf{w}, \tag{6.3}$$

where $\mathbf{w}$ is a vector of non-negative weights. As its columns, the dictionary $\mathbf{D}$ contains exemplars, which express audio representative to different HMM states of the system. Instead of operating with MFCC features, the exemplars usually contain spectral magnitude features form a few consecutive audio frames.

The dictionary may also contain exemplars representative to different noise backgrounds, which accounts for the noise robustness of the method. In case the dictionary consists of sets of exemplars from different speakers $\mathbf{S}_1, \mathbf{S}_2, ...$ and other sounds $\mathbf{N}_1, \mathbf{N}_2, ...$ as $\mathbf{D} = [\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{N}_1, \mathbf{N}_2, ...]$, NMF processing performs sound source separation via weight vector partition $\mathbf{w} = \left[\mathbf{w}_{S1}^T, \mathbf{w}_{S2}^T, ..., \mathbf{w}_{N1}^T, \mathbf{w}_{N2}^T, ...\right]^T$ with

$$\boldsymbol{x} \approx \mathbf{D}\mathbf{w} = \mathbf{S}_1\mathbf{w}_{S1} + \mathbf{S}_2\mathbf{w}_{S2} + ... + \mathbf{N}_1\mathbf{w}_{N1} + \mathbf{N}_2\mathbf{w}_{N2} + ... \tag{6.4}$$

Due to this capability for source separation, NMF processing is also often utilized as preprocessing step for other kinds of ASR frameworks (Geiger et al. (2014)).

Since the dictionary $\mathbf{D}$ is multiple times overdetermined, the weight vector $\mathbf{w}$ is not unique and some optimization algorithm with additional constraints must be utilized to solve it. Then, some means to obtain HMM state likelihoods based on the weight vector $\mathbf{w}$ are needed. These aspects, in addition to dictionary training, are discussed in more detail below.

Dictionary training

In some early implementations of NMF for ASR, dictionary training has been done using algorithms, which factorize a matrix $\mathbf{X}$ of training data features into two non-negative matrices $\mathbf{D}$ and $\mathbf{W}$ such that the reconstruction $\mathbf{R} = \mathbf{D}\mathbf{W}$ error of $\mathbf{X} \approx \mathbf{R}$ is minimized in terms of Euclidean error or divergence

$$\mathrm{D}(\mathbf{X}, \mathbf{R}) = \sum_{f,n} \left(X(f,n) \log \frac{X(f,n)}{R(f,n)} - X(f,n) + R(f,n) \right) \tag{6.5}$$

(Lee and Seung (2001)), where f and n denote indices of the elements within each data matrix. In addition to minimizing the reconstruction error, sparsity of the weight matrix $\mathbf{W}$ is characteristics, which is desired for sparse classification. An algorithm, which promotes sparsity of the reconstruction matrix $\mathbf{W}$ while minimizing the Euclidean reconstruction error is presented by Hoyer (2004).

In the above mentioned non-negative matrix factorization methods, the size of dictionary atoms is the same as the input feature vector from one audio frame. To model the continuity of audio signal it has been found out that instead of atom vectors for one audio frame, using exemplars that represent multiple consecutive audio frames is beneficial. In this case, the representation of Equation (6.3) accounts for a sequence (window) of input frames and the above mentioned dictionary learning algorithms are not applicable. Thus a popular approach of building a dictionary $\mathbf{D}$ for ASR has been to collect the exemplars into it by sampling from training data (Raj et al. (2010), Schmidt and Olsson (2006)). The sampling may be done randomly, or by selecting good representatives of all the acoustic events that are desired to be explicitly modeled. In Publications II,I and V, an overly large dictionary is first built by random sampling. It is then pruned to the desired size such that the dictionary exemplars cover the different acoustic events more or less evenly.

Optimization for feature representation in terms of the dictionary

When the dictionary $\mathbf{D}$ for factorization of input vectors in $\mathbf{X}$ is given, the optimization of weights $\mathbf{W}$ is done using constraints for non-negativity and sparsity. In audio processing, when spectral magnitudes or power spectral densities are used as raw features in input vectors $\boldsymbol{x}$, the measure mostly used for the reconstruction error is the Kullback-Leibler divergence

$$\mathrm{D}_{KL}(\boldsymbol{x}, \mathbf{D}\mathbf{w}) = \sum_{f} \boldsymbol{x}(f) \log \frac{\boldsymbol{x}(f)}{\mathbf{D}\mathbf{w}(f)}. \tag{6.6}$$

The standard algorithm for finding weights $\mathbf{w}$ which minimize $\mathrm{D}_{KL}(\boldsymbol{x}, \mathbf{D}\mathbf{w})$ is a gradient descent type EM-algorithm presented in Lee and Seung (2001), which is used also in publications II, I and V. A fast active-set Newton algorithm (ASNA) algorithm for the task has been proposed by Virtanen et al. (2013). An algorithm which, in addition to sparsity, promotes also temporal continuity of weights in $\mathbf{W}$ while minimizing the divergence D_{KL} is presented by Virtanen (2007).

Non-negative matrix deconvolution (NMD) -algorithms , e.g. by Smaragdis (2004), have been specifically developed for implementations, where the dictionary exemplars represent episodes of multiple input frames. In this kind of implementations, the above discussed NMF algorithms have to be utilized for each input vector separately, since the utilized vectors consist of concatenated features from multiple audio frames. The NMD algorithms factorize the whole sequence of audio feature vectors at once. NMD -algorithms optimize the usage of multi-frame dictionary exemplars in $\mathbf{D}$ for representation of the audio frame sequence $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T]$ in such a way, that the audio frame sequence will be represented by the combined effort of obtained weight vectors $\mathbf{w}_t,\ t \in \{1...T\}$, which are optimized to be sparse also in respect to time dimension.

Transforming exemplar weights to ASR -system state likelihoods

It would be possible to utilize the sparse weights $\mathbf{w}$ as features for any type of ASR -framework. For signal enhancement, a subset $\mathbf{w}_s$ of weights in $\mathbf{w}$ corresponding to a clean subset $\mathbf{D}_s$ of exemplars in $\mathbf{D}$ is used for clean speech feature reconstruction as $\hat{\boldsymbol{x}}_s = \mathbf{D}_s \mathbf{w}_s$. Within plain NMF -based ASR-frameworks of Publications II,I and V, the weights are converted into speech state likelihoods l_{state} using a trained linear transform.

When the dictionary of multi-frame exemplars is sampled from training data, each exemplar has associated with it a certain speech state sequence, which is obtained from labels of the training data. The speech state likelihoods according to $\mathbf{w}$ are now obtained as $\mathbf{l} = \mathbf{L}\mathbf{w}$, where $\mathbf{L}$ is a binary matrix of dictionary speech state labels and each column of $\mathbf{L}$ corresponds to labels of one dictionary exemplar. This approach has been utilized in my Publication V, and as a reference algorithm in Publications II and I.

Optionally, a transformation matrix may be trained for the conversion from $\mathbf{w}$ to $\mathbf{l}$ such that $\mathbf{l} = \mathbf{B}\mathbf{w}$. This approach has been used in my Publications II and I, where different transformation matrices $\mathbf{B}$ have been trained using OLS and PLS -algorithms.

6.2.2 Acoustic modeling using deep neural networks

The artificial neural networks (ANN), invented already in 1943 by Warren McCulloch and Walter Pitts, have turned effective for ASR task only from the beginning of the 21st century. In 1990s, one layer neural networks were tried out for ASR task, but the results of these early works showed that a shallow one-layer NN is too simple structure to be able to charactrize the complexity inherent in large vocabulary speech signal. Once many problems in learning parameters for a DNN had been solved and the necessary computational power for DNN parameter learning had become available by the beginning of the 21st century, DNNs became a new dominant methodology in many fields if machine intelligence, including ASR.

For ASR, DNNs have been utilized mainly in two different ways, either for directly providing state likelihoods l_{state} for HMM analysis, or for providing better features to be utilized in place of MFCCs for a GMM-HMM based framework. The so called *hybrid* ASR systems utilize DNN for providing state likelihoods l_{state}, which are interpreted as conditional input probabilities of the HMM as

$$P(\,\boldsymbol{x}\,|\,\text{state}\,) \propto l_{\text{state}}/P(\,\text{state}\,), \tag{6.7}$$

where $P(\text{ state })$ is the prior probability of a HMM state (Virtanen et al. (2018), page 399). The DNN is trained discriminatively with state labels of training data. A so called *tandem* ASR framework utilizes these DNN based features in place or aside of MFCCs within a GMM-HMM based framework. The DNN features by a deep autoencoder were demonstrated to excel the MFCC features first time for ASR by Heck et al. (2000).

Many different input formats of audio frames for DNNs have been experimented with. In general it has been found out, that DNNs are able to learn excellently performing features, and thus hand-crafted feature extraction is unnecessary or even degenerative for performance. The best results have been obtained using band-pass-filter output energies, e.g. STFT-magnitudes or Mel frequency scale coefficients, and even raw time-domain audio signal has been succesfully utilized as DNN input by Tüske et al. (2014).

DNNs pre-trained as a stack of RBMs

The renaissance of NNs in the form of DNNs truly appeared, when the research group of Geoffrey Hinton (2007) published their inventions on generative training of RBMs, specifically their discovery on how to stack multiple one-layer RBMs on top of each other. They showed how arbitrarily many RBM layers may be trained by contrastive divergence (CD) algorithm, one at a time, to maximize the probability of the input data, and stacked as a generative DBN. This generative DBN is then turned into a discriminative DNN by adding one more layer of neurons for producing the outputs, i.e. outputting likelihoods of each speech state, and fine-tune-training the DNN with backpropagation algorithm. The backpropagation training of a DBN based DNN is optimized in terms for maximal mutual information (MMI) criterion among the true and estimated state sequences. This kind of hybrid DBN-DNN has been shown to outperform a GMM-HMM framework in multiple large vocabulary continuous speech recognition (LVCSR) tasks by Hinton et al. (2012).

Deep Convolutive Neural Networks

Convolutive neural networks (CNN) have been utilized in image processing with great success (LeCun and Bengio (1995)). For images, CNNs have very crucial functionality to provide invariance in position, rotation and scale for feature computation. However, it is not obvious how this capability would best leverage ASR. A spectrogram of STFT-magnitudes of an audio sequence may be considered as an image, but for example similar phenomema at low or high frequencies should likely have different interpretations, and total position or scale invariance in frequency dimension is not what is desired for ASR. On the other hand, it has been noticed, that the time varying nature of speech is better modeled with HMM than with CNN. These concerns have been solved by Abdel-Hamid et al. (2014) applying the convolutive operation only in frequency dimension. In addition, limited weight sharing is utilized, such that only the units that are attached to the same pooling unit share the same convolution weights. This restricted position invariance compared to puristic CNN is shown to improve recognition of acoustic events.

Deng et al. (2013) demonstrate that using one or more convolutional layers before the full connectivity layers within a DNN improves LVCSR performance. They show how invariance to vocal tract differences between different speakers is obtained by using one or more convolutional layers with weight-sharing across nearby frequencies and then pooling the convolution filter responses to similar frequencies.

In Sainath et al. (2013) the authors show that CNN performs better as feature extractor within a CNN-GMM-HMM system than as a state likelihood estimator within a hybrid CNN-HMM speech recognizer. They train the feature extractor CNN as an autoencoder with a couple of convolutive layers first and a bottle-neck of 512 units to be turned into the feature extractor output layer.

Deep Recurrent neural networks

Recurrent neural networks utilize feed-back loops, which enable them to model dependencies among consecutive inputs (Haykin (1998)). Tandem systems combining RNN outputs with GMM-HMM models have not been particularly successfull. This is probably due to that the additional time-domain information given by deep RNN outputs over feed forward DNN outputs causes confusion for learning the GMM models and thus the combined RNN-GMM-HMM systems fails being efficient. Instead, RNN architectures are at best when trained to directly output phoneme likelihoods against an error function in respect to longer speech sequences than individual phonemes, where the best frame-phoneme alignment of the sequence is left for the neural network to decide about. When decoding a sequence of input frames based on RNN phoneme likelihood outputs e.g. beam search is utilized to yield a list of best transcription candidates.

Again, one-layer RNNs have appeared to be too simple models to handle large vocabulary speech recognition (LVSR) task, but once the problems in learning deep RNNs had been solved, they have become the most successful of all DNNs for LVSR (Graves et al. (2013)). Specifically, Long-Short-Term Memory RNNs (LSTM) networks (Hochreiter and Schmidhuber (1997)) have shown to provide excellent recognition accuracy (Sak et al. (2014)). Human level conversational speech recognition accuracy has been achieved using deep LSTM networks by Xiong et al. (2017).

6.3 A cascaded state classifier for ASR

Within an automatic speech recognition task, system state likelihoods/probabilities are used as an intermediate representation of the input data. This state likelihood estimation task is similar to the assignment of multi-class classification problems, while the final ASR transcription is made using the language model. In Publication V on ASR task I have utilized cascade processing, which was discussed in Chapter 5, for efficient state likelihood estimation. The idea in the cascade processing is that where the speech is clear and easy to recognize, the state likelihood estimation is performed with a computationally

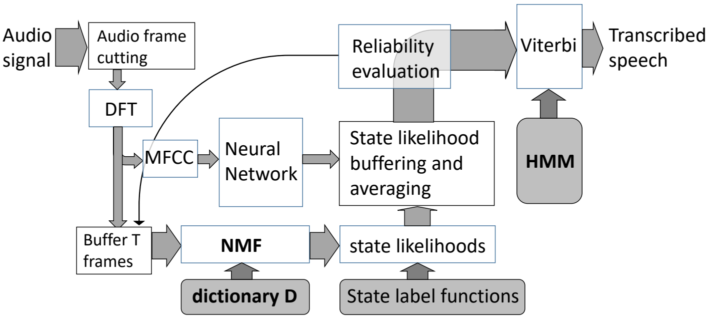
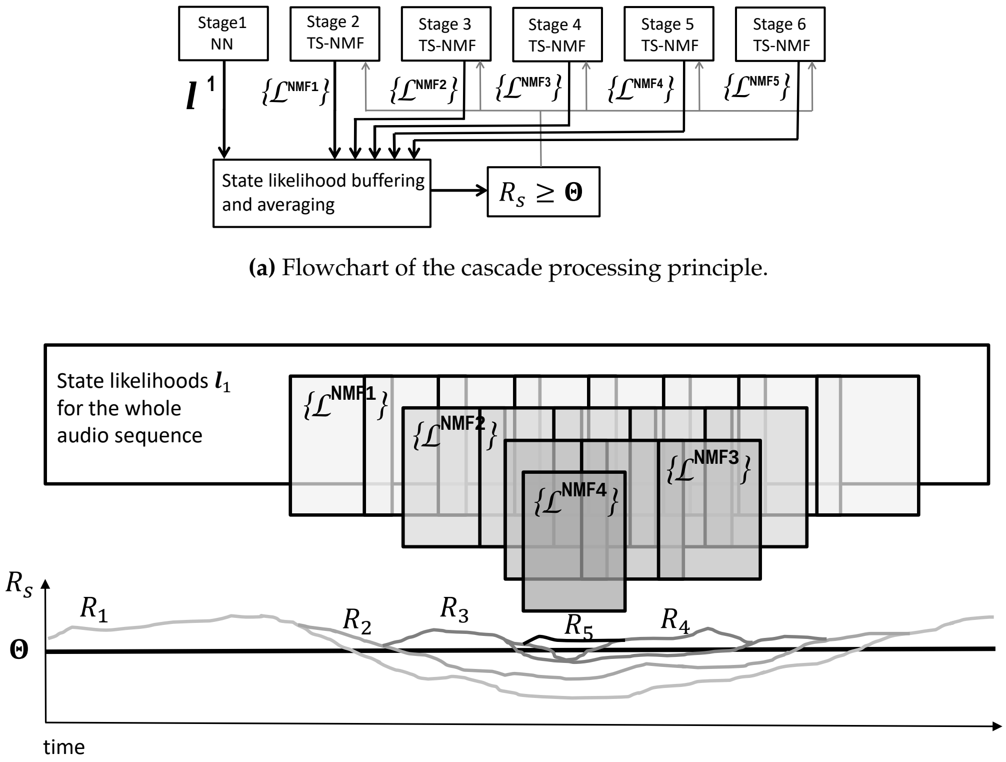


Figure 6.3: An ASR framework utilized in Publication V, which utilizes a neural network and NMF based acoustic modeling for HMM state likelihood estimation. Efficient cascade processing for state likelihood estimation is obtained with the state likelihood information reliability evaluation, which guides the amount of NMF processing.


Stage1
NN
Stage 2
TS-NMF
Stage 3
TS-NMF
Stage 4
TS-NMF
Stage 5
TS-NMF
Stage 6
TS-NMF
$\boldsymbol{l}^1$
$\{\mathcal{L}^{\text{NMF1}}\}$
$\{\mathcal{L}^{\text{NMF2}}\}$
$\{\mathcal{L}^{\text{NMF3}}\}$
$\{\mathcal{L}^{\text{NMF4}}\}$
$\{\mathcal{L}^{\text{NMF5}}\}$
State likelihood buffering and averaging
$R_s \geq \boldsymbol{\Theta}$
State likelihoods $\boldsymbol{l}_1$ for the whole audio sequence
$\{\mathcal{L}^{\text{NMF1}}\}$
$\{\mathcal{L}^{\text{NMF2}}\}$
$\{\mathcal{L}^{\text{NMF3}}\}$
$\{\mathcal{L}^{\text{NMF4}}\}$
R_s
R_1
R_2
R_3
R_5
R_4
$\boldsymbol{\Theta}$
time

(a) Flowchart of the cascade processing principle.

(b) Cumulation of state likelihood certainty R_s.

Figure 6.4: Cascade processing principle utilized in ASR state likelihood estimation cascade of Publication V. State likelihood information $\mathbf{l}^s$ is cumulated stage by stage via computing new state likelihood windows $\mathcal{L}_t^s$ (shown with shaded color) with NMF until the state likelihood reliability R_s exceeds the threshold Θ.

fast neural network, and the more accurate, computationally heavy NMF-method is utilized only for parts that need it and only as much as it is necessary. The overall ASR framework utilizing cascade processing for speech state likelihood estimation is shown in Figure 6.3, and the cascade processing principle is illustrated in Figure 6.4.

At the first stage of the state likelihood estimation cascade a simple feed forward NN is used for fast state likelihood estimation from feature vectors. The system operates on Mel-spectrogram magnitude feature vectors of audio input frames. There are 250 outputs in the NN, and the output values are converted to Bayesian probabilities for all the system states $c = 1...250$ to be used as class likelihoods $l_1^c(t)$ for each audio frame. The details of the NN and the value conversion can be found in Publication V.

At the subsequent stages $s = 2...5$ of the ASR cascade, the computationally heavy NMF-method is used to acquire state likelihoods $l_s^c(t),\ c = 1...250$. The NMF method utilized at these stages produces windows of state likelihood information in matrices $\mathcal{L}_t$, each referring to a sequence of 20 input frames with indexes $t...t + 19$ and containing likelihoods of all the states as its columns. The Figure 6.4 illustrates the process. The state likelihood $l_s^c(t)$ at stage s is then computed as a weighted average of the likelihood $l_1^c(t)$ from the first cascade stage and the c:th row of all state likelihood windows $\mathcal{L}_t$ acquired up to the stage s as

$$l_s^c(t) = (1-\lambda)\, l_1^c(t) + \lambda \left[\sum_{\tau=0}^{T-1} \mathcal{L}_{t-\tau}(c, \tau+1) \right]_{\mathbf{1}} , \tag{6.8}$$

where $\lambda \in [0,1]$ is a balancing factor for the weighted average and $[\circ]_{\mathbf{1}}$ denotes vector normalization to ℓ_1 norm 1.

To decide about whether the next cascade stage should be entered to acquire more state likelihood information or not, we utilize a value of average state likelihood reliability as

$$R_s(t) = \frac{1}{2\,T_{\text{ave}}} \sum_{\tau=t-T_{\text{ave}}}^{t+T_{\text{ave}}-1} \max\left(\, l_s^1(\tau), l_s^2(\tau), ..., l_s^{250}(\tau)\,\right), \tag{6.9}$$

where T_{ave} is the length of the averaging window. The reliability $R_s(t)$ is compared to a threshold value Θ and if $R_s(t) \geq \Theta$ the state likelihood information for frame $\boldsymbol{x}$ is considered reliable and the next cascade stage is not called for the frame $\boldsymbol{x}_t$. If $R_s(t) < \Theta$, new input windows covering the frame $\boldsymbol{x}_t$ are factorized with the NMF method. Figure 6.4 illustrates the process.

Performance of the proposed ASR framework in terms of speech recognition accuracy and the computational load is illustrated in Figure 6.5. The performance curve is obtained by changing the operating point, that is, the threshold Θ for the state likelihood reliability R_s, of the framework. The speech recognition accuracy of the reference NMF framework, which has been utilized for NMF processing within the cascade and is depicted in Figure 6.2, is shown with the dashed red line. Using the proposed cascade processing for state likelihood estimation, the speech recognition task is performed six times as fast as the baseline NMF framework, the processing time reducing to only 16 % of the original, without compromising the recognition accuracy. In Figure 6.5 this operating point is denoted with a cross. To our surprise, the recognition accuracy improved 2 % using the cascade framework. This is likely due to the additional information given by the neural network. The maximum accuracy of 88.5 % at its most economical operating point, obtained with $\Theta = 0.51$, is denoted in Figure 6.5 with a circle.

6.4 Experimental results in small vocabulary ASR

The Publications II, I and V deal with a small vocabulary speech recognition task of CHiME challenge 2011 (Barker et al. (2013)). The speech data of the task is based on GRID corpus (Cooke et al. (2006)), which contains voices by 34 different speakers, 18 male and 16 female. The GRID corpus contains mono recordings of spoken command sentences, e.g. "Set green with R2 now.", with a small vocabulary and strict grammar. The defined sentence structure is described in Table 6.1. The language model of this dataset consists of 250 states, 4-6 states per word. The audio material of the noisy background recordings is taken from the CHiME corpus (Christensen et al. (2010)). The recordings are done in a living room and a kitchen of a home of two adults and two children using a head and torso simulator, which captures binaural stereo audio. For mixing, the anechoic 1-channel speech recordings of the GRID corpus have been adapted into the domestic room acoustics of CHiME noise environments using binaural room impulse responses provided by the CHiME corpus. Then the acquired 2-channel signals are mixed with CHiME stereo backgrounds at 6 different signal to noise ratios (SNR = -6, -3, 0, 3, 6,

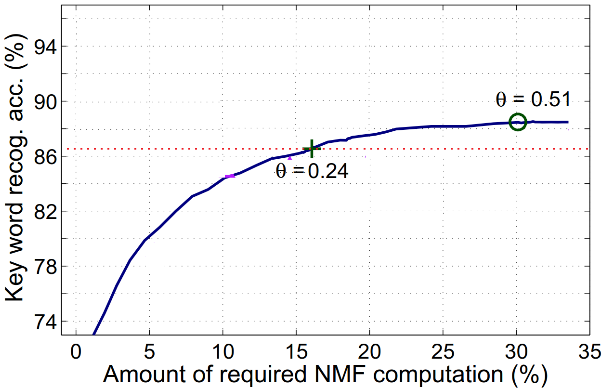


Figure 6.5: ASR results in terms of key word recognition accuracy and computational load with the ASR framework proposed in Publication V using cascade processing for state likelihood estimation. The threshold Θ accounts for required state likelihood reliability level within the framework. The dashed line marks the key word recognition accuracy of the baseline NMF framework, which requires 100% NMF computation.

9 dB to produce in total 3600 noisy speech audio files. The dataset has been used in the PASCAL CHiME speech separation and recognition challenge (Barker et al. (2013)) held in 2011, and in the second CHiME speech separation and recognition challenge (Vincent et al. (2013b,a)) in 2013. A set of 600 utterances from this dataset is used in the experiments of Publications II, I and V.

The NMF based ASR framework experimented with in all these publications is illustrated in Figure 6.2. The NMF based framework has been selected due to its noise robustness.

In the framework, a feature vector X_t used for NMF factorization contains 20 DFT magnitude vectors $\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, ..., \boldsymbol{x}_{t+19}$ corresponding to 20 contiguous audio frames. A DFT magnitude vector $\boldsymbol{x}_t$ from one audio frame is thus utilized within 20 consecutive audio window feature vectors $X_{t-19}, ...X_t$. Different NMF window lengths were tested in the Publications II and I, and the length 20 frames, representing an audio snippet of ca. 0.2 s, was found to perform best on average. This setting was thus used also in Publication V.

An overcomplete dictionary **D** of 5000 example spectrograms, i.e. *exemplars* is learned from training data using regular sampling and post pruning for even distribution of exemplars of audio window features. The exemplar spectrograms are vectorized similarly to the input spectrograms to form columns of matrix **D**. Each exemplar is also associated

Table 6.1: Vocabulary and grammar of the GRID corpus Cooke et al. (2006).

command	color	preposition	letter	digit	adverb
bin	blue	at	A - Z	zero, 1 - 9	again
lay	green	by	excluding W		now
place	red	in			please
set	white	with			soon

Table 6.2: Key word recognition accuracies obtained using the NMF based ASR framework of Publication II with noisy speech audio with different SNR levels. Three different methods, namely dictionary labels, OLS and PLS regression, for converting sparse NMF activation vectors into state likelihoods within the framework are compared. The best key word accuracy for each SNR level is written in bold. The learned transformations can be seen to outperform the performance with dictionary labels.

SNR (dB)	9	6	3	0	-3	-6
Speaker independent case						
labels	77.3	72.8	68.2	62.7	51.1	44.0
OLS	**85.2**	**80.5**	**78.7**	**71.1**	**60.2**	**51.5**
PLS	82.9	78.8	74.8	70.1	59.5	50.6
Speaker dependent case						
labels	91.6	89.2	87.6	84.2	74.7	68.0
OLS	91.1	**90.0**	**88.5**	**85.2**	77.6	69.2
PLS	**91.9**	89.3	88.2	85.0	**78.6**	**69.6**

with a sequence of 20 state labels given by training data phoneme alignment. The state labels of the dictionary exemplars are encoded in binary label matrices $\mathbf{L}_i, i = 1...5000$ of size 250 x 20. Details of the dictionary building process can be found in Publication I.

For analysis, each input feature vector X_t is factorized as

$$X_t \approx \mathbf{D} \cdot \mathbf{w}_t, \tag{6.10}$$

to weights, i.e. *activations*, $\mathbf{w}_t$ based on the dictionary $\mathbf{D}$. The factorization is done using an NMF algorithm, which minimizes the Kullback-Leibler divergence between X_t and $\mathbf{D}\mathbf{w}_t$ and induces sparsity of $\mathbf{w}_t$. The weight vector $\mathbf{w}_t$, similarly to X_t, concerns the window of 20 contiguous audio frames starting at the frame indexed by t.

To convert the sparse activation vector $\mathbf{w}_t$ into a window $\mathcal{L}_t$ of state likelihoods, three different methods are tried out in Publications II and I. The first method utilizes the binary label matrices $\mathbf{L}_i, i = 1...5000$ associated with the dictionary $\mathbf{D}$. The state likelihood window $\mathcal{L}_t$ is obtained from $\mathbf{w}_t$ using the binary label matrices $\mathbf{L}_i$, as

$$\mathcal{L}_t = \sum_{i=1}^{5000} w_t(i) \cdot \mathbf{L}_i. \tag{6.11}$$

The two other methods utilize a transformation matrix $\mathbf{B}$ as

$$\mathcal{L}_t = \mathbf{B}\mathbf{w}_t. \tag{6.12}$$

The transformation $\mathbf{B}$ is trained either by OLS or by PLS -regression algorithm.

The Table 6.2 shows the main results of comparing the keyword recognition accuracies obtained by using the three state likelihood conversion methods from Publication II. The inferior results with other analysis window lengths than 20 can be found in the publication. The Table 6.2 shows that utilizing the learned transformations $\mathbf{B}$ enables clearly higher recognition accuracies than utilizing dictionary labels especially in the speaker independent case. In the speaker independent case, the variability of input spectrograms for which a certain dictionary exemplar is activated is high and thus learning the feature to state likelihood transformation via machine learning defeats manual labels of the dictionary. The equally good performance with dictionary labels in speaker dependent case is likely due to that the speaker dependent dictionaries are highly specific. Thus in this case the labels represent well the interpretation of the activations from the NMF factorization.

In the Publication I, the framework of Figure 6.2 is used as the baseline, and a non-negative matrix deconvolution, i.e. convolutive sparse coding, based system is experimented. The NMD based framework is similar to that of Figure 6.2, except that the magnitude spectra of the entire input audio is buffered, instead of buffering 20 frames, for the non-negative factorization. Then, instead of NMF factorization algorithm, an NMD algorithm is used. It produces sparse feature vectors $\mathbf{w}_t$ for some X_t, such that the entire input feature sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{\text{sentenceLength}}$ is represented by the weighted dictionary exemplars as precisely as possible. The reconstruction error is measured in terms of Kullback-Leibler divergence of the magnitude spectra. The details of the process can be found in Publication I.

With this publication we participated in the CHiME noisy speech recognition challenge 2011. To our surprise, the NMD framework did not yield superior performance to the baseline NMF framework, and thus the submitted results were those highlighted in Table 6.2. The Figure 6.6 shows the performance of our NMF framework compared to other participants of the CHiME challenge. The performance of our framework is denoted with label "Hurmalainen et al.". The performance curve over varying noise conditions show that our solution was close to the state of the art at the time of the publication.

The Publication V, the cascade operation of which was already discussed in Section 6.3, presents an experiment on improving the computational efficiency of the NMF based framework of Figure 6.2. The proposed framework, where the system state likelihoods are

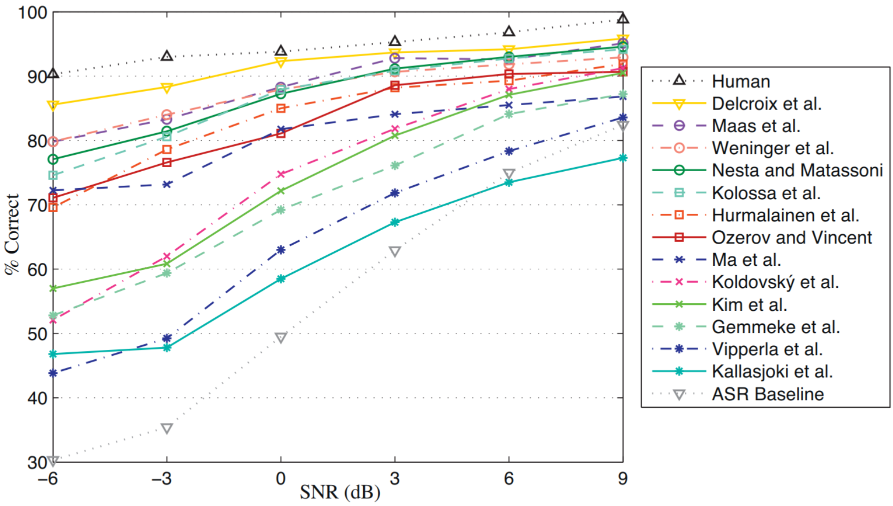


Figure 6.6: The performance of frameworks submitted to CHiME challenge 2011. The results of our framework are shown with label Hurmalainen et al. . The figure is reproduced from Barker et al. (2013) with permission.

computed using a cascade process is shown in Figure 6.3. Another change to the baseline framework, in addition to cascade processing that was experimented in Publication V, concerns the space of system states. The original set of 250 states encompassing the used vocabulary contains multiple phonetically very similar states. Unions of some of these states were formed for state likelihood sharing. The details of forming the state unions can be found in the Publication V. The state union based likelihood sharing improved the key word recognition accuracy of the baseline framework with 0.7 % units. The cascade processing was capable of reducing the computational load remarkably, as demonstrated in Section 6.3. At the same time the recognition accuracy improved by 2 % units due to the additional information given by the neural network.

7 Conclusions and discussion

In the work leading to this thesis multiple tasks within the field of signal processing has been examined. The collection of included publications concern musical audio dereverberation, video cut detection, video classification and automatic speech recognition. Within these tasks, cascade processing principle has been examined for combining multiple methodologies for a task in computationally efficient way. Following the order of presentation used in this thesis, I discuss below the research questions and the starting point of each publication and state the conclusions reached.

The work of the Publication III was an interesting mix of realism and laboratory conditions. The inspiration of the work was a highly realistic scenario to decrease excessive reverberation in ad hoc music recordings. Due to tight schedule and resources the algorithm evaluation was to be done in terms of computational SNR-improvement, instead of more reliable subjective evaluation. Thus, in the lack of reference signals for real life recordings needed for SNR computation, for the experiments we utilized audio material generated from MIDI representations and applied reverberation and dynamic compression. The dereverberation achieved on the real life recordings was only briefly evaluated by listening, and not reported in the publication. A linear prediction and spectral subtraction based dereverberation method proposed by Furuya and Kataoka (2007) for speech dereverberation was selected for use. The results showed that the algorithm may successfully be utilized in conjunction with music signals. The results also showed that dynamically compressed music can be dereverberated with the method with no increased sound quality deterioration.

In Publications IV, V and VI cascade processing principle has been examined for tasks of video cut detection, automatic speech recognition and video classification. The results of these publications have shown that combining multiple algorithms in computationally efficient way using cascade processing principle can be successfully implemented for all these tasks. That is, the results of these publications show, that remarkable speed-up can be obtained such that it does not compromise the detection, classification or ASR transcription accuracy.

The Publication IV deals with the task of video cut detection, although many researchers consider the task of video shot detection to be already solved. Furthermore, the shots and scenes of the video data used for the tests are randomly generated, thus it is controversial whether this research has potential to give any new information for the field. However, combining multiple video cut detection functions, based on the audio and visual data streams of the video, with Boolean **AND** and/or **OR** operators was tested in the work. All the tested detector combinations were shown to improve the detection accuracy over each individual detector. Then, an effect of sequential evaluation principle for solving the Boolean function on the computational load of the combined detection system was evaluated. The results showed that some Boolean functions for combining multiple detectors bring remarkable computational savings.

In the Publication VI, a framework for combining multiple sensitivity tunable classifiers with Boolean **AND** and **OR** operators as a computationally efficient cascade structure has been formulated. An algorithm for learning the operation points of the classifiers to be used within the framework has been presented. The framework has been applied to video classification task using classifiers based on audio and visual streams of videos. The results showed that the developed cascade framework is capable of combining the classifiers in computationally and classification accuracy-wise efficient way. Specifically, the framework reached classification accuracy superior to others found in the literature for the used dataset and the excellent classification accuracy was achieved with a fraction of the computational load of the other classifiers in comparison.

The work of Publications II, I and V concern automatic speech recognition with noisy background but small vocabulary. This is very realistic scenario for voice command devices. To see the value of the developed algorithms for broader purpose ASR, the experiments should be repeated with a large vocabulary data set. Within the Publications II, I and V acoustic modeling with alternative methods to Gaussian Mixture Models of MFCC-features have been examined for this scenario.

In Publications II and I acoustic modeling with a magnitude spectrogram based NMF -approach was tried out. The Publication I, participating The PASCAL CHiME Speech Separation and Recognition Challenge Barker et al. (2013) with results close to the winning team, showed that the NMF approach was indeed an effective method in comparison to other methods. The Publication II examined whether using a trained conversion function from the sparse NMF activation vectors to HMM state likelihoods provides better ASR transcription accuracy than utilization of dictionary labels for the transform. The hypothesis showed out to be true.

In Publication V a combined approach for acoustic modeling using the NMF method and a neural network as a computationally efficient cascade structure was examined. The results of the publication showed that the computational load of the NMF-based framework may be remarkably reduced with cascade processing, and that a simple neural network can be successfully used as the fast first stage HMM state likelihood estimation function. The results showed that the NN gives new information aside of NMF and thus improves the ASR transcription accuracy.

Sequential processing, including cascade processing, is a universal principle, which would likely benefit many computational frameworks. I see high potential in implementing this processing principle for virtually any signal processing task. By incorporating multiple information retrieval algorithms into one sophisticated system in computationally efficient way it would be possible to exploit the best features of each algorithm without computational overload. This is especially important for mobile and autonomous systems, which should not be too reliant on external computation capacity.

Bibliography

Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G., and Yu, D., "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

Aharon, M., Elad, M., and Bruckstein, A., "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, no. 11, pp. 4311–4322, 2006.

Allen, J., "Synthesis of pure speech from a reverberant signal," 1974, u.S. Patent No. 3786188.

Andersson, M., "A comparison of nine PLS1 algorithms," *Journal of Chemometrics*, vol. 23, no. 10, pp. 518–529, 2009.

Arnfield, S., Roach, P., Setter, J., Greasley, P., and Horton, D., "Emotional stress and speech tempo variation," in *Proceedings of ESCA-NATO workshop on Speech under stress*. ISCA, 1995.

Atkinson, K. E., *An Introduction to Numerical Analysis*. John Wiley & Sons, USA, 1979.

Barker, J., Vincent, E., Ma, N., Christensen, H., and Green, P., "The PASCAL CHiME speech separation and recognition challenge," *Computer Speech & Language*, vol. 27, no. 3, pp. 621 – 633, 2013.

Beach, A., *Real world video compression*. Pearson Education, 2010.

Bellman, R., *Dynamic Programming (DP)*. Princeton University Press. Dover paperback edition (2003), 1957.

Beranek, L., *Concert halls and opera houses: music, acoustics, and architecture*. Springer Science & Business Media, 2012.

Bishop, C. M., *Pattern recognition and machine learning*. Springer Science + Business Media, 2006.

Boros, E., Hammer, P. L., Ibaraki, T., and Kogan, A., "Logical analysis of numerical data," *Mathematical Programming*, vol. 79, no. 1, pp. 163–190, Oct 1997.

Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

Breiman, L., "Random forests," *Machine Learning*, vol. 1, no. 45, pp. 5–32, 2001.

Brownlow, K., "Silent films: What was the right speed?" *Sight & Sound*, pp. 164–167, 1980.

Canny, J., "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, no. 6, pp. 679–698, 1986.

Capon, J., "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.

Christensen, H., Barker, J., Ma, N., and Green, P. D., "The CHiME corpus: a resource and a challenge for computational hearing in multisource environments," in *Interspeech*. Citeseer, 2010, pp. 1918–1921.

Cooke, M., Barker, J., Cunningham, S., and Shao, X., "An audio-visual corpus for speech perception and automatic speech recognition," *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2421–2424, 2006.

Cox, D. R., "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, 1958.

Davis, S. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, no. 4, pp. 357–366, 1980.

de Rivaz, P. and Haughton, J., "AV1 bitstream & decoding process specification," The Alliance for Open Media, Tech. Rep., Jun. 2018.

Dempster, A., Laird, N., and Rubin, D., "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 1, no. 39, pp. 1–38, 1977.

Deng, L., Hinton, G., and Kingsbury, B., "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 2013.

Deshpande, A. and Triantaphyllou, E., "A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions," *Mathematical and Computer Modelling*, vol. 27, no. 1, pp. 75–99, Jan. 1998.

Dietzen, T., Spriet, A., Tirry, W., Doclo, S., Moonen, M., and van Waterschoot, T., "On the relation between data-dependent beamforming and multichannel linear prediction for dereverberation," in *Audio Engineering Society Conference: 60th International Conference: DREAMS (Dereverberation and Reverberation of Audio, Music, and Speech)*, Jan 2016.

Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern classification*. John Wiley & Sons, 2012.

Dundar, M. and Bi, J., "Joint optimization of cascaded classifiers for computer aided detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

Falk, T. H., Zheng, C., and Chan, W.-Y., "A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1766–1774, 2010.

Feraund, R., Bernier, O. J., Viallet, J.-E., and Collobert, M., "A fast and accurate face detector based on neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 1, pp. 42–53, 2001.

Foundation, X., “Theora specification,” Tech. Rep., Jun. 2017.

Freund, Y. and Schapire, R. E., “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

——, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, pp. 119–139, 1997.

Fumera, G., Fabio, R., and Alessandra, S., “A theoretical analysis of bagging as a linear combination of classifiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 7, pp. 1293–1299, 2008.

Furuya, K. and Kataoka, A., “Robust speech dereverberation using multichannel blind deconvolution with spectral subtraction,” *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 15, no. 5, pp. 1579–1591, 2007.

Gannot, S., Burshtein, D., and Weinstein, E., “Signal enhancement using beamforming and nonstationarity with applications to speech,” *IEEE Transactions on Signal Processing*, vol. 49, no. 8, pp. 1614–1626, Aug 2001.

Gaubitch, N. D., Naylor, P. A., and Ward, D. B., “On the use of linear prediction for dereverberation of speech,” in *Proceedings of the IEEE International Workshop on Acoustic Echo and Noise Control*, 2003, pp. 99–102.

Gaubitch, N. D., Ward, D. B., and Naylor, P. A., “Statistical analysis of the autoregressive modeling of reverberant speech,” *The Journal of the Acoustical Society of America*, vol. 6, no. 120, pp. 4031–4039, 2006.

Geiger, J. T., Gemmeke, J. F., Schuller, B., and Rigoll, G., “Investigating nmf speech enhancement for neural network based acoustic models,” in *Interspeech*, 2014.

Ghanbari, M., *Standard codecs: Image compression to advanced video coding*. Iet, 2003, no. 49.

Gonzalez, R. C. and Woods, R. E., *Digital image processing*, third edition ed. Pearson Prentice Hall, 2008.

Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

Grange, A., de Rivaz, P., and Hunt, J., “VP9 bitstream & decoding process specification,” Tech. Rep., Mar. 2016.

Graves, A., Mohamed, A., and Hinton, G. E., “Speech recognition with deep recurrent neural networks,” *Computing Research Repository (CoRR)*, vol. abs/1303.5778, 2013.

Habets, E. A. P., “Single-channel speech dereverberation based on spectral subtraction,” in *15th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2004)*, Nov 2004.

Habets, E. A. P. and Naylor, P. A., “Dereverberation,” in *Computational Analysis of Sound Scenes and Events*, Virtanen, T., Plumbley, M. D., and Ellis, D., Eds. Springer, 2018, ch. 15, pp. 331–359.

Hammer, P. L., "Partially defined boolean functions and cause-effect relationships," *Lecture in International Conference on Multi-attribute Decision Making Via OR-based Expert Systems*, April 1986.

Han, K., Wang, Y., Wang, D., Woods, W. S., Merks, I., and Zhang, T., "Learning spectral mapping for speech dereverberation and denoising," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 6, no. 23, pp. 982–992, June 2015.

Harris, F. J., "On the use of windows for harmonic analysis with the discrete fourier transform," vol. 66, no. 1, pp. 51–83, 1978.

Harris, Z., "Distributional structure," *Word*, vol. (2/3), no. 10, pp. 146–62, 1954.

Hartigan, J., *Clustering algorithms*. Wiley, New York, 1975.

Haykin, S., *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

Haykin, S. S., *Adaptive filter theory*. Pearson Education India, 2008.

Heck, L. P., Konig, Y., Sönmezc, M. K., and Weintraub, M., "Robustness to telephone handset distortion in speaker recognition by discriminative feature design," *Speech Communication*, vol. 2, no. 31, pp. 181 – 192, 2000.

Hinton, G. and Salakhutdinov, R., "Reducing the dimensionality of data with neural networks," *Science*, no. 5786, p. 504, 2006.

Hinton, G. E., "Training products of experts by minimizing contrastive divergence," *Neural Computation*, no. 14, pp. 1771–1800, aug. 2002.

Hinton, G., "Where do features come from?" *Cognitive Science*, vol. 38, no. 6, pp. 1078–1101, 2014.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., "Deep neural networks for acoustic modeling in speech recognition," *Signal Processing Magazine*, 2012.

Hinton, G. E., "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, pp. 428–434, 2007.

Hinton, G. E., Osindero, S., and Teh, Y.-W., "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

Hochreiter, S. and Schmidhuber, J., "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Hoyer, P. O., "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, no. 5, p. 1457–1469, 2004.

ITU-R Recommendation BS.1387-1, "Method for objective measurements of perceived audio quality," International Telecommunication Union, Geneva, Switzerland, Tech. Rep., Nov. 2001.

ITU-R Recommendation BS.1534-1, "Method for the subjective assessment of intermediate quality levels of coding systems," International Telecommunication Union, Geneva, Switzerland, Tech. Rep., Jan. 2003.

ITU-T Recommendation H.264, “Advanced video coding for generic audiovisual services,” International Telecommunication Union, Geneva, Switzerland, Tech. Rep., Apr. 2017.

ITU-T Recommendation H.265, “High efficiency video coding,” International Telecommunication Union, Geneva, Switzerland, Tech. Rep., Feb. 2018.

ITU-T Recommendation P.262, “Application guide for objective quality measurement based on recommendations s P.862, P.862.1 and P. 862,” International Telecommunication Union, Geneva, Switzerland, Tech. Rep., Mar. 2005.

Jackson, L. B., *Digital Filters and Signal Processing*, 2nd ed. Boston: Kluwer Academic Publishers, 1989.

Jeub, M., Schäfer, M., , and Vary, P., “A binaural room impulse response database for the evaluation of dereverberation algorithms,” in *Proceedings of the 16th International Conference on Digital Signal Processing (DSP)*, IEEE, IET, EURASIP. IEEE, 2009, pp. 1–5.

Kinoshita, K., M., D., T., N., and M., M., “Suppression of late reverberation effect on speech signal using long-term multiple-step linear prediction,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, no. 4, pp. 534–545, May 2009.

Kinoshita, K., Delcroix, M., Gannot, S., Habets, E. A., Haeb-Umbach, R., Kellermann, W., Leutnant, V., Maas, R., Nakatani, T., Raj, B. *et al.*, “A summary of the reverb challenge: state-of-the-art and remaining challenges in reverberant speech processing research,” *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–19, 2016.

Kittler, J., Hatef, M., Duin, R. P., and Matas, J., “On combining classifiers,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

Kodrasi, I. and Doclo, S., “Robust partial multichannel equalization techniques for speech dereverberation,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2012, pp. 537–540.

Krizhevsky, A., Sutskever, I., and Hinton, G. E., “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25*, Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

Kubichek, R., “Mel-cepstral distance measure for objective speech quality assessment,” in *Communications, Computers and Signal Processing, 1993., IEEE Pacific Rim Conference on*, vol. 1. IEEE, 1993, pp. 125–128.

Kumar, V. and Minz, S., “Feature selection,” *SmartCR*, vol. 4, no. 3, pp. 211–229, 2014.

Kuttruff, H., *Room acoustics*. Crc Press, 2016.

LeCun, Y. and Bengio, Y., “Convolutional networks for images, speech, and time-series,” 1995.

LeCun, Y., Bengio, Y., and Hinton, G., “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

Lee, D. D. and Seung, H. S., “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems (NIPS)*, Leen, T. K., Dietterich, T. G., and Tresp, V., Eds. MIT Press, 2001, pp. 556–562.

Li, H., Lin, Z., Shen, X., Brandt, J., and Hua, G., "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5325–5334.

Lim, F., Zhang, W., Habets, E. A. P., and Naylor, P. A., "Robust multichannel dereverberation using relaxed multichannel least squares," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 22, no. 9, pp. 1379–1390, 2014.

Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

McConnell, R., "Method of and apparatus for pattern recognition," Jan. 28 1986, US Patent 4,567,610.

Mertins, A., Mei, T., and Kallinger, M., "Room impulse response shortening/reshaping with infinity- and p -norm optimization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 249–259, Feb 2010.

Michalski, R. S., *A Theory and Methodology of Inductive Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 83–134.

Miyoshi, M. and Kaneda, Y., "Inverse filtering of room acoustics," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 2, pp. 145–152, Feb 1988.

Nakatani, T., Yoshioka, T., Kinoshita, K., Miyoshi, M., and Juang, B. H., "Speech dereverberation based on variance-normalized delayed linear prediction," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1717–1731, Sept 2010.

Nakatani, T., Kinoshita, K., and Miyoshi, M., "Harmonicity-based blind dereverberation for single-channel speech signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, pp. 80–95, 2007.

Naylor, P. A. and Gaubitch, N. D., *Speech Dereverberation*, 1st ed. Springer Publishing Company, Incorporated, 2010.

Naylor, P. A., Gaubitch, N. D., and Habets, E. A., "Signal-based performance evaluation of dereverberation algorithms," *Journal of Electrical and Computer Engineering*, vol. 2010, p. 1, 2010.

Novak, C. and Shafer, S., "Anatomy of a color histogram," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1992, pp. 599 – 605.

Nyquist, H., "Certain topics in telegraph transmission theory," *Transactions of American Institute of Electrical Engineers (AIEE)*, pp. 617–644, 1928.

Ojala, T., Pietikäinen, M., and Harwood, D., "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR)*, vol. 1, 1994, pp. 582 – 585.

Ojala, T., Pietikainen, M., and Mäenpää, T., "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 7, pp. 971–987, 2002.

Oppenheim, A. V., Schafer, R. W., , and Buck, J. R., *Discrete-time signal processing*, 1999.

O'Shaughnessy, D., *Speech communications: human and machine*. Institute of Electrical and Electronics Engineers, 2000.

Oxley, M. E., Thorsen, S. N., and Schubert, C. M., "A boolean algebra of receiver operating characteristic curves," in *Proceedings of the 10th International Conference on Information Fusion*. IEEE, 2007, pp. 1–8.

Padaki, H., Nathwani, K., and Hegde, R. M., "Single channel speech dereverberation using the LP residual cepstrum," in *National Conference on Communications (NCC)*, Feb 2013.

Petridis, S., Rajgarhia, V., and Pantic, M., "Comparison of single-model and multiple-model prediction-based audiovisual fusion," in *Proceedings of the 1st Joint Conference on Facial Analysis, Animation, and Auditory-Visual Speech Processing (FAAVSP)*. ISCA Speech Organisation, 2015.

Proakis, J. G. and Manolakis, D. G., *Digital signal processing*, third edition ed. Prentice Hall, 1996.

Quackenbush, S. R., Barnwell, T. P., and Clements, M. A., *Objective measures of speech quality*. Prentice Hall, 1988.

Quinlan, J. R., "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

——, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

Rabiner, L. R. and Juang, B.-H., *Fundamentals of speech recognition*. PTR Prentice Hall Englewood Cliffs, 1993, vol. 14.

Raj, B., Virtanen, T., Chaudhuri, S., and Singh, R., "Non-negative matrix factorization based compensation of music for automatic speech recognition," in *Proceedings of Interspeech*, 2010.

Raj, B., Virtanen, T., and Singh, R., "The problem of robustness in automatic speech recognition," in *Techniques for noise robustness in automatic speech recognition*, Virtanen, T., Singh, R., and Raj, B., Eds. John Wiley & Sons, 2012, ch. 3, pp. 33–53.

Rao, H., Ye, Z., Li, Y., Clements, M. A., Rozga, A., and Rehg, J. M., "Combining acoustic and visual features to detect laughter in adults' speech," in *Joint Conference on Facial Analysis, Animation and Audio-Visual Speech Processing (FAAVSP)*, 2015, pp. 153–156.

Read, P. and Meyer, M.-P., *Restoration of motion picture film*. Butterworth-Heinemann, 2000.

Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P., "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 2001, pp. 749–752.

Rossing, T. D., *The science of sound*, second edition ed. Addison-Wesley Publishing Company, 1990.

Rudovic, O., Petridis, S., and Pantic, M., "Bimodal log-linear regression for fusion of audio and visual features," in *Proceedings of the 21st ACM International Conference on Multimedia*. ACM, 2013, pp. 789–792.

Saberian, M. J. and Vasconcelos, N., "Learning optimal embedded cascades," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 2005–2018, 2012.

Sainath, T. N., Rahman Mohamed, A., Kingsbury, B., and Ramabhadran, B., "Deep convolutional neural networks for LVCSR," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8614–8618.

Sak, H., Senior, A. W., and Beaufays, F., "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014.

Salakhutdinov, R., Mnih, A., and Hinton, G., "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML '07. ACM, 2007, pp. 791–798.

Schmidt, M. N. and Olsson, R. K., "Single-channel speech separation using sparse non-negative matrix factorization," in *Interspeech*, September 2006.

Shen, C., Wang, P., Paisitkriangkrai, S., and van den Hengel, A., "Training effective node classifiers for cascade classification," *International Journal of Computer Vision*, vol. 103, pp. 326–347, 2013.

Smaragdis, P., "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs," in *Proceedings of International Conference on Independent Component Analysis and Blind Signal Separation (ICA)*. Springer, September 2004, pp. 494–499.

Sochman, J. and Matas, J., "Waldboost - learning for time constrained sequential detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Steinwart, I. and Christmann, A., *Support Vector Machines*. Springer-Verlag, New York, 2008.

Stevens, K., *Acoustic phonetics*, ser. Current studies in linguistics series. Cambridge, 1998.

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

Tao, Q. and Veldhuis, R., "Threshold-optimized decision-level fusion and its application to biometrics," *Pattern Recognition*, vol. 42, no. 5, pp. 823–836, 2009.

Tibshirani, R., "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B (methodological)*, vol. 58, pp. 267–88, 1996.

Tikhonov, A., Goncharsky, A., Stepanov, V., and Yagola, A., *Numerical Methods for the Solution of Ill-Posed Problems*. Kluwer Academic Publishers, 1995.

Triki, M. and Slock, D. T. M., "Blind dereverberation of quasi-periodic sources based on multichannel linear prediction," in *International Workshop on Acoustic Echo and Noise Control, (IWAENC)*, September 2005.

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

Vincent, E., Gribonval, R., and Févotte, C., "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 14, no. 4, pp. 1462–1469, 2006.

Vincent, E., Barker, J., Watanabe, S., Le Roux, J., Nesta, F., and Matassoni, M., "The second 'chime'speech separation and recognition challenge: An overview of challenge systems and outcomes," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013, pp. 162–167.

Vincent, E., on Barker, Watanabe, S., Roux, J. L., Nesta, F., and Matassoni, M., "The second 'CHiME' speech separation and recognition challenge: Datasets, tasks and baselines," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 126–130.

Viola, P. and Jones, M., "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, 2001.

——, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2001, pp. I–I.

Virtanen, T., "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, March 2007.

Virtanen, T., Gemmeke, J. F., and Raj, B., "Active-set newton algorithm for overcomplete non-negative representations of audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 11, pp. 2277–2289, Nov 2013.

Virtanen, T., Plumbley, M. D., and Ellis, D., Eds., *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

Watanabe, S., Virtanen, T., and Kolossa, D., "Application of source separation to robust speech analysis and recognition," in *Computational Analysis of Sound Scenes and Events*, Virtanen, T., Plumbley, M. D., and Ellis, D., Eds. Springer, 2018, ch. 17, pp. 393–428.

Wen, J. Y., Gaubitch, N. D., Habets, E. A., Myatt, T., and Naylor, P. A., "Evaluation of speech dereverberation algorithms using the mardy database," in *in Proceedings of International Workshop on Acoustic Echo and Noise Control (IWAENC)*. Citeseer, 2006.

Weninger, F., Geiger, J. T., Wöllmer, M., Schuller, B. W., and Rigoll, G., "Feature enhancement by deep LSTM networks for ASR in reverberant multisource environments," *Computer Speech & Language*, vol. 28, pp. 888–902, 2014.

Xiao, X., Zhao, S., Nguyen, D. H. H., Zhong, X., Jones, D. L., Chng, E.-S., and Li, H., "The ntu-adsc systems for reverberation challenge 2014," in *Proc. REVERB challenge workshop*, 2014, p. o2.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and Zweig, G., "Achieving human parity in conversational speech recognition," *Computation and Language*, 2017.

Yoshioka, T., Hikichi, T., and Miyoshi, M., "Dereverberation by using time-variant nature of speech production system," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, p. 065698, Aug 2007.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H., "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 3320–3328.

Zhang, W., Habets, E., and Naylor, P., "On the use of channel shortening in multichannel acoustic system equalization," in *Proceedings of International Workshop on Acoustic Echo and Noise Control (IWAENC)*, 2010, pp. 465–468.

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Laurent, C., Bengio, Y., and Courville, A. C., "Towards end-to-end speech recognition with deep convolutional neural networks," *CoRR*, vol. abs/1701.02720, 2017.

Zhu, X. and Ramanan, D., "Face detection, pose estimation, and landmark localization in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 2879–2886.

Zou, H. and Hastie, T., "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, p. 301–320, 2005.

Zwicker, E., "Subdivision of the audible frequency range into critical bands," *The Journal of the Acoustical Society of America*, pp. 248–248, 1961.

Zwicker, E. and Fastl, H., *Psychoacoustics*, ser. Springer Series in Information Sciences. Springer-Verlag Berlin Heidelberg, 1999.

Publications

Publication I

Exemplar-based Recognition of Speech in Highly Variable Noise

Antti Hurmalainen[1], *Katariina Mahkonen*[1], *Jort F. Gemmeke*[2], *Tuomas Virtanen*[1]

[1]Department of Signal Processing, Tampere University of Technology, Finland
[2]Department ESAT, Katholieke Universiteit Leuven, Belgium

antti.hurmalainen@tut.fi, katariina.mahkonen@tut.fi,
jgemmeke@amadana.nl, tuomas.virtanen@tut.fi

Abstract

Robustness against varying background noise is a crucial requirement for the use of automatic speech recognition in everyday situations. In previous work, we proposed an exemplar-based recognition system for tackling the issue at low SNRs. In this work, we compare several exemplar-based factorisation and decoding algorithms in pursuit of higher noise robustness. The algorithms are evaluated using the PASCAL CHiME challenge corpus, which contains multiple speakers and authentic living room noise at six SNRs ranging from 9 to -6 dB. The results show that the proposed exemplar-based techniques offer a substantial improvement in the noise robustness of speech recognition.

Index Terms: automatic speech recognition, exemplar-based, noise robustness, sparse representation

1. Introduction

While Automatic Speech Recognition (ASR) has been under intensive research for decades, its widespread adoption is still being delayed by practical issues. One of the primary problems is varying background noise. Conventional ASR systems, based on frame level Gaussian Mixture Models (GMMs), suffer significant quality degradation when spectral features become corrupted by noise. Joint modelling of the target speech and noise in the recognizer, [1], feature compensation [2], and missing data techniques [3] have been suggested to overcome this problem. Meanwhile, there are alternative routes, which no longer employ GMMs to discover the underlying speech content.

In previous work [4, 5], we have described an *exemplar-based* recognition framework, where noisy speech is represented as a combination of multi-frame speech and noise spectrogram segments, *exemplars*. The framework can be used for signal or feature enhancement, but the best results have been achieved by using exemplar labels, which directly reveal the phonetic content of an utterance via their activation weights. In this paper, we explore the effectiveness of the exemplar-based framework on highly corrupted speech using the PASCAL CHiME challenge data, in which the speech is not only reverberated, but also contains phonetically close keywords and highly variable background noise events.

Concerning our framework, the CHiME data provides a few interesting options, which were not present in the previous experiments carried out on the AURORA-2 database. First, the data is stereophonic and high quality. Second, the utterances to be recognised can be observed within their neighbouring noise context. Finally, the identity of the speaker is known at the moment of recognition, so speaker-dependent speech exemplars can be reliably employed.

The rest of the paper is organised as follows. The general concepts of our exemplar-based approach are described in Section 2. The experimental setup, including the CHiME database, feature extraction and parameter settings of the baseline system are presented in Section 3. The baseline exemplar-based recognition results are shown and discussed in Section 4. Experiments with two variants; the use of matrix deconvolution (NMD) and the use of regression to learn the mapping between words and exemplars, are described in Sections 5 and 6, respectively. The overall discussion of our findings is presented in Section 7, and the summary and conclusions in Section 8.

2. Recognition with speech and noise exemplars

Sparse representations have received increasing attention in several applications, including image and audio signal processing. The key concept is that many natural signals can be described as a linear combination of only a few atoms. Enforcing sparsity prevents overfitting with too many elements. By allowing only a small number of activations, we can expect to find the few dictionary atoms, which best explain the mixed signal.

In noise robust speech recognition, it has been proposed that speech may be described as a sparse linear combination of *exemplars*, and that noisy speech can likewise be described as a combination of noise and speech exemplars [5, 6, 7]. When a noisy utterance is represented using these components, the activations of speech exemplars, together with knowledge of the words they represent, can be used to recognise the underlying utterance.

2.1. Sparse representation of noisy speech

The base element of our sparse representation is an *exemplar*, a $B \times T$ spectrogram block of B spectral magnitudes of speech or noise in T consecutive frames, extracted from training data. The exemplars are indexed by variable e. To simplify the notation, the columns of each spectrogram matrix are stacked into vector $\mathbf{a}_e$ of length $B \cdot T$. The E exemplars are gathered into the columns of matrix $\mathbf{A}$ to form a *basis* or *dictionary*.

The utterance to be recognised is similarly converted to spectral features. A length T observation window is concatenated into vector $\mathbf{y}$. The observation window is represented as a linear combination of exemplars,

$$\mathbf{y} \approx \sum_{e=1}^{E} \mathbf{a}_e x_e, \tag{1}$$

where x_e is the weight or *activation* of each exemplar.

In the baseline exemplar-based recognition system we employ an algorithm referred as 'NMF' (*Non-negative Matrix Factorisation*) to find the non-negative and sparse activations. The vector $\mathbf{x}$ of all activations x_e in Equation 1 can be determined simultanously for multiple observation vectors stored in columns of matrix $\mathbf{Y}$, each producing its own column to the total activation matrix $\mathbf{X}$. The matrix equation to be solved thus becomes $\mathbf{Y} \approx \mathbf{AX}$.

We obtain the non-negative activation matrix $\mathbf{X}$ while minimising the Kullback-Leibler divergence and introducing an sparsity-inducing L_1 penalty for non-zero activations by using the update rule

$$\mathbf{X} \leftarrow \mathbf{X} \otimes \frac{\mathbf{A}^{\mathrm{T}}(\mathbf{Y}/(\mathbf{AX}))}{\mathbf{A}^{\mathrm{T}}\mathbf{1} + \mathbf{\Lambda}}. \quad (2)$$

Here $\otimes$ denotes elementwise multiplication. Matrix divisions are also elementwise. $\mathbf{1}$ is an utterance-sized all-ones matrix. $\mathbf{\Lambda}$ is the sparsity penalty matrix, defined for each activation entry.

For recognition of utterances of arbitrary length T_{utt}, we process the utterance in $W = T_{\text{utt}} - T + 1$ overlapping feature windows with a step of one frame between windows. Because the middle frames are estimated several times in consecutive windows, averaging is applied to the likelihoods of the next step to compensate for this. For a thorough description of this factorisation method, see [4]. An alternative method for handling temporal continuity, referred as *Non-negative Matrix Deconvolution* (NMD), is presented in Section 5.

2.2. Recognition

To decode the signal, we create a $Q \times T_{\text{utt}}$ *likelihood matrix* $\mathbf{L}$, where each entry $\mathbf{L}_{q\tau}$ denotes the probability of speech state q $(1 \ldots Q)$ in frame τ $(1 \ldots T_{\text{utt}})$. This is generated using *conversion matrices* $\mathbf{B}_t$ $(Q \times E)$, which describe the linear mapping of exemplars to states for each frame t of the exemplars. In our baseline system, we use binary labelling of dictionary exemplars for the conversion. In each exemplar frame only one state is labelled to be active. The matrices need not to be binary, though. in Section 6 we will experiment with a technique to *learn* the conversion matrices in order to take into account dependencies between exemplar activations.

After generating the whole matrix $\mathbf{L}$ as described in [4], each of its columns (representing state likelihoods in one frame) is normalised to unitary sum. The matrix is then decoded using a Viterbi algorithm and trained transition parameters.

3. Experimental setup

3.1. The CHiME database

The PASCAL 'CHiME' Speech Separation and Recognition Challenge [8] is designed to address some of the problems occurring in real world noisy speech recognition. The challenge data is based on the GRID corpus [9], where 34 speakers read simple command sentences. These sentences are of form *verb-colour-preposition-letter-digit-adverb*. There are 25 different 'letter' class words and 10 different digits. Other classes have four word options each. In the CHiME recognition task, the final score is the percentage of correctly recognised 'letter' and 'digit' keywords.

CHiME utterances simulate a scenario, where sentences are spoken in a noisy living room. The original, clean speech utterances are reverberated according to the actual room response, and then mixed to selected noise sections, which produce the desired SNR mixture level for each noisy set. The noisy sets have target SNR levels of 9, 6, 3, 0, -3 and -6 dB.

For modelling/training, there are 500 reverberated utterances per speaker (no noise), and six hours of background noise data. The development and test sets consist of 600 mixed-speaker utterances at each SNR level, Additionally, noiseless (only reverberated) development utterances are available. Development and test utterances are both given in a strictly end-pointed format, but also as embedded signals within their noise context. All data is stereophonic and has a sampling rate of 16 kHz.

3.2. Feature extraction

For the features of our framework, we used spectral magnitudes of Mel bands. These were calculated from partially overlapping 25 ms frames with a shift of 10 ms between frames. 26 bands were used for the 16 kHz signal (Nyquist frequency 8 kHz), which matches the number of bands used for the default CHiME MFCC models. Features were extracted separately for both stereo channels and concatenated, thus effectively doubling the number of feature bands.

3.3. Speech exemplars

We used 5000 speech and 5000 noise exemplars for each window length T, adding up to $E = 10000$ total entries. We created two different types of speech dictionaries: a speaker-dependent and a speaker-independent one. First, an initial speech dictionary was created for each speaker, based on a 60% subset of the noiseless speech training utterances, by extracting exemplars with a random frame shift of 4 to 8 frames. This produced approximately 10000–17000 partially overlapping exemplars per speaker and window length. For the speaker-dependent dictionaries, each initial dictionary was reduced to a fixed size of 5000 exemplars by selecting exemplars such that there is a maximally flat coverage between words. (In the original dictionaries, words from classes with fewer options are over-represented due to more frequent appearance in the training set.)

A speaker-independent dictionary was created for each window length, this time by selecting 147–148 (5000/34) exemplars from each full speaker-dependent dictionary with similar word probability flattening. These were then combined to a single 5000 exemplar dictionary per window length.

In addition to storing the spectral feature data, state labels were assigned to the speech exemplars by using transcriptions acquired by forced alignment. Alternatively, the state information was learnt by factorising the remaining 40% of training files and finding the mapping as described in Section 6.

3.4. Noise exemplars

The selection of noise exemplars has a central role in the separation quality of factorisation algorithms. If no matching noise is found, separation results become unpredictable. Initially, we created two different types of noise dictionaries. In the first, 5000 noise exemplars were randomly extracted from the provided background noise data. In the second, 5000 noise exemplars were selected by sampling the neighbourhood of embedded utterances to both directions with a shift of 4 to 7 frames, excluding locations where other test utterances were embedded.

Experiments using the development set (not shown) indicated that using the adaptive noise dictionary yields a 1–4% improvement in recognition accuracy compared to the fixed noise dictionary. In this paper, we will only report results obtained using adaptive noise.

Table 1: Results of the baseline exemplar-based recogniser on the test set. The rows refer to different exemplar sizes. CHiME GMM baseline results are also shown. The best result at each SNR level is highlighted.

SNR (dB)	9	6	3	0	-3	-6
CHiME baseline	**82.1**	70.8	61.3	52.0	39.8	34.7
$T = 10$	69.9	66.0	58.7	52.4	42.9	37.8
$T = 20$	77.3	72.8	**68.2**	**62.7**	51.1	44.0
$T = 30$	76.0	**73.5**	**68.2**	61.8	**52.7**	**44.7**

(a) Speaker-independent results

SNR (dB)	9	6	3	0	-3	-6
CHiME baseline	82.4	75.0	62.9	49.5	35.4	30.3
$T = 10$	91.3	88.3	85.8	80.8	71.4	62.3
$T = 20$	**91.6**	**89.2**	**87.6**	**84.2**	74.7	68.0
$T = 30$	88.8	88.1	86.3	82.9	**75.1**	**68.3**

(b) Speaker-dependent results

3.5. Processing test utterances

For factorisation, each utterance was read from the endpointed ('isolated') file, and converted into Mel features. After choosing the appropriate speech and noise basis for the utterance, they were reweighted together to equal Euclidean norm over Mel bands and exemplars. Band weights from the combined dictionary were then applied to the utterance features.

The NMF penalty matrix $\mathbf{\Lambda}$ used in finding a sparse representation can be set for each activation entry separately. We used two different values, one for speech exemplars and another for noise. The values were tuned by factorising a subset of development utterances with partially adaptive, speaker-dependent bases and exemplar size $T = 20$. The penalty values were set as 2.0 and 1.7 for speech and noise exemplars, respectively. Generally speaking, higher values of $\mathbf{\Lambda}$ produce better recognition rates at high SNRs, while lower ones lead to better performance at low SNRs. We selected values, which give a slight emphasis to the noisy end. The same sparsity values were used throughout all experiments.

For state representation, we used the same model as in the CHiME baseline recogniser. Each word is modelled with 4–10 successive states, and the whole system uses in total 250 states. The activations were mapped to state likelihoods as explained in section 2.2. Utterances were decoded using the HVite binary of the HTK toolkit, modified to pick its state likelihoods directly from the generated matrix $\mathbf{L}$ instead of evaluating state GMMs.

4. Baseline system results

The results of the baseline exemplar-based recogniser are presented in Table 1. Three different window lengths, $T = 10$, 20 and 30 are shown, as well as results for both speaker-dependent and speaker-independent systems. The GMM-based CHiME baseline recognition results are also shown. When comparing the results, note that the baseline system uses mono features without noise compensation other than cepstral mean normalisation.

In general, it is clear that the exemplar-based recognition system outperforms the baseline GMM system in almost all conditions, especially when using speaker-dependent speech dictionaries. The lower performance of speaker-independent dictionaries ensues because a mixed speech dictionary only has a very limited number of exemplars to match a certain speaker, while at the same time it has a larger chance of matching to speech features in the background noise, produced by people in the living room or by various entertainment appliances. Interestingly, the speaker-independent GMM-based system was more noise robust at low SNRs, possibly because the trained Gaussians have a larger variance and thus match corrupted speech features better.

Like in experiments on AURORA-2 [4, 5], using an exemplar size of $T = 10$ was found suboptimal at low SNRs, because not enough time context can be exploited. $T = 20$ generally turned out equal or superior to $T = 10$. Exemplar size $T = 30$ is the most robust against noise, but performs worse at high SNRs. As the exemplar size increases, the dimensionality of feature vectors grows, and it becomes more difficult to find a matching linear combination of speech exemplars. Using a higher number of exemplars may alleviate this effect, at the cost of increased computational complexity.

5. Non-negative matrix deconvolution

As a first variant of the baseline exemplar-based recognition system, we use *Non-negative Matrix Deconvolution* (NMD) rather than NMF to obtain sparse representations of noisy speech. NMD is a name given to an alternative method to handle temporal continuity between frames. The algorithm has also been called *convolutive sparse coding* [10].

While not a deconvolution algorithm in the traditional sense, the name reflects the principle that a reconstructed utterance is represented as a convolution between activations and exemplars. This means that all the activations jointly form the estimated utterance matrix. A few activations at specific temporal locations are typically enough to represent the utterance features. There are no independent estimates or averaging like in the sliding window NMF. For the convolutive update algorithm and comparison of behaviour, see [11].

The results for NMF and NMD algorithms are shown in Table 2. Both methods employ adaptive noise dictionaries, speaker-dependent speech dictionaries and 300 iterative updates. In NMF, the speech exemplar activations were normalised to unitary sum in each window. In NMD, no normalisation was performed. These choices have been found recommendable in earlier work [4, 11].

In these results, NMF produces slightly yet significantly better recognition rates in all conditions. This is surprising, because on AURORA-2 we observed the opposite: NMD outperformed NMF. Especially the degradation of NMD at $T = 30$ is unexpected, because on AURORA-2 it was the best performing exemplar size [11].

One possible reason is that factorisation parameters were optimised using NMF. Because the full optimisation process is computationally heavy, the same parameters were applied directly to NMD. Therefore the results may favour NMF. We can also speculate, that the closely related keywords in CHiME are prone to occasional misclassifications in sparse activations. As there is more averaging over independent estimates in NMF, the chance of errors in the final estimate is smaller than in NMD. Because a 1–2% drop was already present in the cleanest end of both keyword classes, we can suspect a problem with word recognition itself, not the noise robustness of NMD.

Table 2: Comparison of NMF and NMD factorisation algorithms in speaker-dependent recognition. The rows refer to different exemplar sizes. The best result at each SNR level is highlighted.

SNR (dB)	9	6	3	0	-3	-6
CHiME baseline	82.4	75.0	62.9	49.5	35.4	30.3
$T = 10$	91.3	88.3	85.8	80.8	71.4	62.3
$T = 20$	**91.6**	**89.2**	**87.6**	**84.2**	74.7	68.0
$T = 30$	88.8	88.1	86.3	82.9	**75.1**	**68.3**

(a) NMF

SNR (dB)	9	6	3	0	-3	-6
CHiME baseline	82.4	75.0	62.9	49.5	35.4	30.3
$T = 10$	88.3	85.9	83.3	78.8	69.1	59.8
$T = 20$	**90.5**	**88.6**	**87.0**	**81.3**	**72.1**	**65.9**
$T = 30$	87.2	86.1	84.0	79.9	70.6	63.3

(b) NMD

6. Mapping from activations to likelihoods

In our baseline system, the mapping from activations to word state likelihoods is based on labels of dictionary items, which have been obtained by forced alignment. However, in label-based mapping of word models there is the inherent problem that phonetically similar features may appear in different contexts. A factorisation algorithm (NMF or NMD) selects the exemplars with best fitting spectral features, while their labels may occasionally suggest a misleading word identity. Such an error will easily result in a misclassification.

We tested an alternative approach, where the mapping was not assigned according to dictionary labels, but *learnt* using regression algorithms on factorised training data labelled by forced alignment. Labels were assigned to a 40% subset of the training set for this purpose. Then a regression algorithm was used to discover optimal mapping matrices between activation vectors and target states.

We used two different regression algorithms, *Ordinary Least Squares* (OLS) and *Partial Least Squares* (PLS) to learn the mapping from activations to likelihoods. OLS is straightforward minimisation of the L_2 error term in mapping. PLS (also known as Projection to Latent Structures) uses an internal, usually lower dimensioned space. The original coordinates are rotated in input and output to the internal space, where the true mapping is optimised. PLS can tolerate a collinearity of input data, contrary to OLS. For details, see [12].

The outcome of the recognition with different likelihood generation methods is shown in Table 3. Results are listed for recognition with binary labels, and OLS/PLS-trained mapping. Speaker-independent results are included, because they provide interesting insight to scenarios where flaws of the original system can be countered with learning.

In speaker-independent recognition, uniform improvements of 4.3–14.1% (absolute) can be seen over the use of binary labels. In these dictionaries, very few instances of each word are present for a specific speaker. This seems to result in numerous misclassifications due to exemplars from other words being activated instead. When the conversion matrices are learnt — in this case from a large amount of training material — the actual correspondence of each exemplar can be discovered with convincing results. Possibly for the abundance of training material coming from all speakers, OLS is mostly superior to PLS.

The speaker-dependent results are more mixed. Here the dictionaries only cover one speaker at a time, and thus can include a broad representation of all words and states. In fact, the reduction algorithm did not remove any of the letter and digit exemplars gathered from the training material, because they all fit in the 5000 exemplar dictionaries. It is also worth noting, that in this scenario the regression matrices were only trained from the speaker's own training subset (200 utterances), which is quite limited regarding keyword appearance. Under this limited training data, the performance of all methods was mostly similar, unlike in the speaker-independent case.

7. Discussion

The CHiME challenge database provided some new insight to the applicability of our exemplar-based methods. Overall, the results appear very plausible. Using properly selected algorithms and parameters, our framework reduced the recognition error rates to less than half of the CHiME baseline system at all SNRs, in many cases even by significantly larger a margin. We also achieved improvements in noise robustness over our previous work on AURORA-2. These gains can be partially attributed to the characteristics of CHiME, which allow construction of accurate dictionaries for both speech and noise.

When the speaker identity is known and thus matching speech exemplars can be selected, correct phonetic features can be picked out reliably even in the presence of other voices. Our speaker-dependent results were significantly better than the speaker-independent ones. Using GMMs the difference was not so clear. Regarding noise dictionaries, we found out that adaptive noise exemplar selection can yield high separation quality under varying noise conditions. Previously there were some concerns over the feasibility of generating a generic noise dictionary using a practically manageable number of exemplars. Our CHiME experiments confirm, that adaptive selection can be used instead of a fixed dictionary. Its implementation should be feasible in practical applications as well.

One surprising and slightly disappointing aspect was the subpar performance of NMD in comparison to sliding window based NMF. It is not certain yet, whether this is a real algorithmic difference or merely a result of insufficient parameter training in NMD. Further experiments and optimisations should be carried out to find out the true capabilities of each factorisation algorithm.

More favourable results were achieved in learnt likelihood mapping. The gains over explicitly assigned labels are positive by themselves. However, in a larger context this means that well performing likelihood mappings can be learnt even for features, which are not directly derived from any specific speech sections. In other words, we can experiment with any kind of dictionary generation methods and then find out the phonetic labels even if none were originally present.

While the separation and likelihood generation algorithms of our framework have already been improved, more attention should be paid to optimising the features and state models for maximal linguistic accuracy. The CHiME data illustrates, how some closely related words can be difficult to tell apart even under favourable conditions. Although noise robustness is a crucial aspect in practical ASR systems and our framework has

Table 3: Comparison of the recognition with three different likelihood generation methods on the test set. In addition to binary labels, OLS and PLS regression are evaluated. The best result at each SNR level and for each exemplar size is highlighted.

SNR	(dB)	9	6	3	0	-3	-6
CHiME	baseline	82.1	70.8	61.3	52.0	39.8	34.7
$T = 10$	labels	69.9	66.0	58.7	52.4	42.9	37.8
	OLS	**84.3**	**77.8**	**71.4**	**65.3**	56.4	48.6
	PLS	82.1	77.1	71.0	64.0	**57.0**	**49.3**
$T = 20$	labels	77.3	72.8	68.2	62.7	51.1	44.0
	OLS	**85.2**	**80.5**	**78.7**	**71.1**	**60.2**	**51.5**
	PLS	82.9	78.8	74.8	70.1	59.5	50.6
$T = 30$	labels	76.0	73.5	68.2	61.8	52.7	44.7
	OLS	**82.8**	**80.5**	**76.3**	**70.7**	**62.1**	**54.4**
	PLS	81.1	77.8	74.3	68.8	61.1	52.4

(a) Speaker-independent recognition

SNR	(dB)	9	6	3	0	-3	-6
CHiME	baseline	82.4	75.0	62.9	49.5	35.4	30.3
$T = 10$	labels	**91.3**	**88.3**	**85.8**	**80.8**	**71.4**	62.3
	OLS	89.8	86.8	85.0	79.7	70.1	62.7
	PLS	90.5	87.8	84.5	80.2	71.3	**63.7**
$T = 20$	labels	91.6	89.2	87.6	84.2	74.7	68.0
	OLS	91.1	**90.0**	**88.5**	**85.2**	77.6	69.2
	PLS	**91.9**	89.3	88.2	85.0	**78.6**	**69.6**
$T = 30$	labels	88.8	**88.1**	86.3	82.9	75.1	68.3
	OLS	88.8	86.0	**86.4**	**83.3**	76.1	**69.2**
	PLS	**89.1**	85.7	84.8	82.4	**77.2**	68.8

(b) Speaker-dependent recognition

shown significant advances in achieving it, the ultimate goal of maximally accurate recognition of speech itself should not be forgotten or compromised. Proper phonetic state models should be introduced instead of the current word models to avoid multiple meanings between similar features, and to make large vocabulary recognition feasible.

8. Conclusions

Exemplar-based methods were presented for recognition of speech in highly variable real world noise. The main framework and its variants were evaluated using the CHiME challenge database, which covers actual living room noise at multiple SNRs. We achieved recognition rates of over 91% at 9 dB, and close to 70% at -6 dB. Long temporal context with 200 ms exemplars, speaker-dependent speech dictionaries and adaptive noise dictionary gathering were found the best options for recognition of noisy speech.

Two separation algorithms, non-negative matrix factorisation and -deconvolution were used for determining the exemplar activations from Mel-scale spectral magnitude features. In these experiments, factorisation of overlapping windows independently from each other performed better than deconvolutive separation of whole utterances at once.

Learning the mappings from exemplar activations to state likelihoods using OLS and PLS regression was proposed. These algorithms were compared to strict binary labels acquired from forced alignment. Highest gains were seen in speaker-independent recognition. The original binary labels produced unreliable results, while mappings learnt from large training data improved the recognition rates by 4–14% (absolute). In speaker-dependent recognition the differences were small.

The results surpassed significantly both the CHiME baseline results and our previous AURORA-2 recognition rates. While the noise robustness of our system is already relatively high, parameter optimisation and better speech models would help in improving the overall recognition quality even further.

9. Acknowledgements

Antti Hurmalainen has been funded by the Academy of Finland. The research of Jort F. Gemmeke was funded by IWT-SBO project ALADIN contract 100049.

10. References

[1] S. J. Rennie, J. R. Hershey, and P. A. Olsen, “Single-channel multitalker speech recognition: Graphical modeling approaches,” *IEEE Signal Processing Magazine*, vol. 27, no. 6.

[2] P. J. Moreno, B. Raj, and R. M. Stern, “A vector taylor series approach for environment-independent speech recognition,” in *Proceedings of IEEE International Conference on Audio, Speech and Signal Processing*, Atlanta, USA, 1996.

[3] B. Raj and R. M. Stern, “Missing-feature approaches in speech recognition,” *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 101–116, September 2005.

[4] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, “Exemplar-based sparse representations for noise robust automatic speech recognition,” *Accepted for publication in IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

[5] T. Virtanen, J. F. Gemmeke, and A. Hurmalainen, “State-based labelling for a sparse representation of speech and its application to robust speech recognition,” in *Proceedings of INTERSPEECH*, Makuhari, Japan, 2010.

[6] G. S. V. S. Sivaram, S. K. Nemala, M. Elhilali, T. D. Tran, and H. Hermansky, “Sparse coding for speech recognition,” in *Proceedings of IEEE International Conference on Audio, Speech and Signal Processing*, Dallas, USA, 2010.

[7] B. Schuller, F. Weninger, M. Wöllmer, Y. Sun, and G. Rigoll, “Non-negative matrix factorization as noise-robust feature extractor for speech recognition,” in *Proceedings of IEEE International Conference on Audio, Speech and Signal Processing*, Dallas, USA, 2010.

[8] H. Christensen, J. Barker, N. Ma, and P. Green, “The CHiME corpus: a resource and a challenge for computational hearing in multisource environments,” in *Proceedings of INTERSPEECH*, Makuhari, Japan, 2010.

[9] M. Cooke, J. Barker, S. Cunningham, and X. Shao, “An audio-visual corpus for speech perception and automatic speech recognition,” *Journal of the Acoustical Society of America*, vol. 120(5), 2006.

[10] T. Virtanen, “Separation of sound sources by convolutive sparse coding,” in *Proceedings of ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing*, München, Germany, 2004.

[11] A. Hurmalainen, J. F. Gemmeke, and T. Virtanen, “Non-negative matrix deconvolution in noise robust speech recognition,” in *Proceedings of IEEE International Conference on Audio, Speech and Signal Processing*, Prague, Czech Republic, 2011.

[12] P. Geladi and B. R. Kowalski, “Partial least-squares regression: a tutorial,” *Analytica Chimica Acta*, vol. 185, no. 1, 1986.

Publication II

Mapping Sparse Representation to State Likelihoods in Noise-Robust Automatic Speech Recognition

Katariina Mahkonen.[1], *Antti Hurmalainen.*[1], *Tuomas Virtanen.*[1], *Jort Gemmeke.*[2]

[1]Department of Signal Processing, Tampere University of Technology, Finland
[2]Centre for Language and Speech Technology, Radboud University Nijmegen, The Netherlands

katariina.mahkonen@tut.fi, antti.hurmalainen@tut.fi,
tuomas.virtanen@tut.fi, jgemmeke@amadana.nl

Abstract

This paper proposes learning-based methods for mapping a sparse representation of noisy speech to state likelihoods in an automatic speech recognition system. We represent speech as a sparse linear combination of exemplars extracted from training data. The weights of exemplars are mapped to speech state likelihoods using Ordinary Least Squares (OLS) and Partial Least Squares (PLS) regression. Recognition experiments are conducted using the CHiME noisy speech database. According to the results, both algorithms can be successfully used for training the mapping. We achieve improvements over the previous binary labeling system, and recognition scores close to 70% at -6 dB SNR.

Index Terms: automatic speech recognition, sparse representations, exemplar-based, regression

1. Introduction

There is a general agreement in the speech community about the need for novel approaches for improving Automatic Speech Recognition (ASR) in adverse conditions [1]. Recently, there has been a renewed interest in non-parametric ASR methods as an alternative for Gaussian Mixture Models (GMMs) [2, 3, 4]. In particular, methods that rely on representing speech as a linear combination of a small set of dictionary atoms have been shown to offer higher classification accuracy [2] and better noise robustness [3].

These methods work by first finding the sparsest possible linear combination of predefined atoms that describe the observed speech spectra. With each atom associated with a speech class [2, 4] or Hidden Markov Model (HMM) states [3], decoding is done by using the weights of atoms directly as evidence of the observed speech likelihoods. In Sivaram et al. [4], the phone classification was done by training a neural network to map the weights of atoms directly to phoneme classes.

In this paper, we employ the exemplar-based speech representation described in [3], where the dictionary atoms are spectrograms extracted from training data. The method has the advantage that noisy speech can be represented as a linear combination of speech and noise exemplars, allowing the method to obtain accurate sparse representations of the speech atoms even in the presence of corrupting background noise.

The recognition in [3] was done using state labels associated to the exemplars. This *canonical transcription* of the exemplars was obtained through forced alignment with a conventional GMM-based recognizer. A downside of using the canonical state association is that the sparse representation is formed of exemplars that represent the underlying speech states of the observed speech, i.e. the method is restricted to using exemplars that are realizations of speech. Furthermore, the use of canonical state association of the exemplars is not optimal for the recognition purpose. It may also require cumbersome handling of silence states, because those are not well represented as a linear combination of exemplars [3].

To circumvent these issues, we propose in this paper to *learn* the mapping between exemplar activations and state likelihoods using regression methods. Using Ordinary Least Squares (OLS) and Partial Least Squares (PLS) regression models, we show that learning the mapping between exemplar activations and likelihoods can handle phonetic ambiguity and labeling errors of dictionary exemplars, especially when enough training material is available.

The rest of the paper is organized as follows. In Section 2 we introduce our noisy speech representation model, describe how we retrieve the linear combination of exemplars used to represent speech and explain how the exemplar activations are used for speech recognition. In Section 3 we describe the two regression models used for mapping exemplar activations to speech state likelihoods. The experimental setup using the CHiME database is presented in Section 4. Results and discussion follow in Section 5, and conclusions in Section 6.

2. Exemplar-based representation of speech

The framework we use for robust speech recognition is shown in Figure 1. There are three steps to be done offline, namely dictionary building, training of conversion matrices and HMM training. Other phases of recognition are to be done with fixed system parameters.

We represent noisy speech using magnitudes within Mel-frequency bands. Magnitudes from T consecutive frames are concatenated to form a feature vector $\mathbf{y}$. This feature vector is then modeled as a sparse linear combination of weighted exemplar vectors $\mathbf{a}_m$, $m = 1...M$ from a dictionary matrix $\mathbf{A}$. As an equation this is:

$$\mathbf{y} \approx \sum_{m=1}^{M} \mathbf{a}_m x_m = \mathbf{A}\mathbf{x} \ , \qquad (1)$$

where x_m is a non-negative weight or activation of the m:th exemplar, and $\mathbf{x}$ is a vector containing all the activations. $\mathbf{A}$ holds in total M exemplars from both speech and noise, which allows representing noisy speech. The details of the representation are described in [3].

Activation values are obtained with a Non-negative Matrix Factorization (NMF) algorithm as in [3]. The algorithm min-

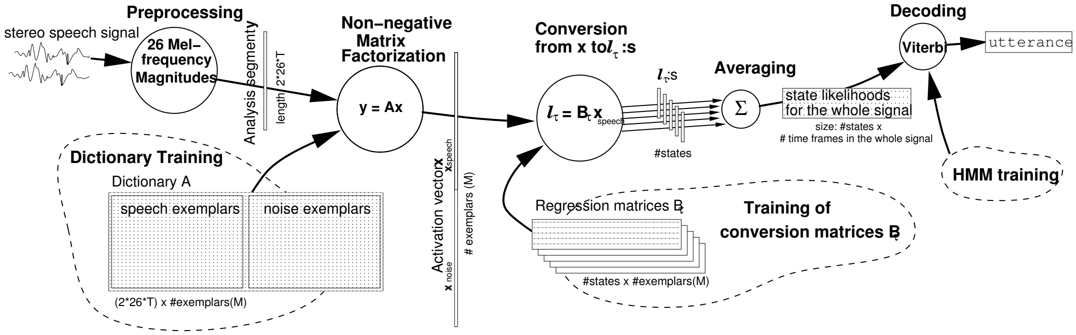


Figure 1: Speech recognition framework.

imizes the Kullback-Leibler divergence between the observations and the model plus L_1 norm sparsity promoting penalty using multiplicative update rules. The system processes long utterances by applying the sparse representation in overlapping, fixed-length segments, whose length is T frames. Shift of one frame is used between overlapping segments.

Our system does speech recognition by using a set of states, which are conceptually similar to the states in conventional HMM-based recognizers. However, in our system the likelihoods of the states are obtained differently. In the baseline system [3], the activations in $\mathbf{x}$ are converted to word state likelihoods by using labeled state lists of dictionary speech exemplars, which are obtained by forced alignment. Element x_m from an activation vector $\mathbf{x}$ is copied as a likelihood value to state likelihood vectors $l_\tau, \tau = 1...T$, which account to T frames inside the analysis segment. x_m is copied into l_τ for the state that is found as the τ:th element of $\mathbf{a}_m$:s state list.

In the end, the overlapping state likelihood vectors for each signal frame from consecutive analysis segments are averaged and combined to make up a state likelihood matrix $\mathcal{L}$ for the whole utterance. From $\mathcal{L}$, the most probable state sequence is tracked with a Viterbi decoder.

3. Proposed mapping from activations to state likelihoods

The goal of the proposed method is to find a mapping from exemplar activation vector $\mathbf{x}$ to state likelihood vector $\boldsymbol{l}_\tau$ in each frame $\tau = 1...T$ of a speech segment. For simplicity, we restrict ourselves to linear mappings. The mapping is given as

$$\boldsymbol{l}_\tau = \mathbf{B}_\tau \mathbf{x} \ , \tag{2}$$

where $\mathbf{B}_\tau$ is the mapping matrix for likelihoods in frame τ. The mapping is found by using training data consisting of activations and corresponding target state likelihood vectors, which are described in more detail in Section 3.1.

In our case the input space is very high dimensional, which makes the use of conventional methods, such as linear discriminant analysis, problematic. There are methods such as regularized discriminant analysis and shrinkage discriminant analysis that can be used with high-dimensional inputs. In this study, however, we explored two regression algorithms for the mapping, namely Ordinary Least Squares (OLS) and Partial Least Squares (PLS) [6]. PLS was chosen since it is known to produce good results in cases where the input data has a large number of dimensions that are highly collinear.

3.1. Training the regression matrices

In our case, the input space of regression matrices is of size 5000. It consists of truncated training data activation vectors $\mathbf{x}_S$, which only have the activations that correspond to speech exemplars of the dictionary $\mathbf{A}$. The target space consists of 250 states. Each training data segment has a labeled length T sequence of forced alignment target states obtained from its canonical transcription. Target vectors $\mathbf{l}_\tau, \tau = 1...T$ for the training are binary, having value 1 for the state that is labeled for the τ:th frame in the said training segment and value zero for all the other states. All the truncated training data activation vectors are collected into columns of matrix $\mathbf{X}$, and all the training data target vectors are gathered into columns of matrices $\mathbf{L}_\tau, \tau = 1...T$. The regression matrix training with the above explained input and output data is then performed with OLS and PLS algorithms.

3.2. Ordinary Least Squares

The *Ordinary Least Squares* (OLS) method finds a linear model for mapping the input space to the output space so that minimizes the total L_2 error of the model and target values. The solution of OLS regression with our variables becomes:

$$\mathbf{B}_\tau \ = \mathbf{L}_\tau \mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^{-1} \ . \tag{3}$$

The problem with the OLS formula above is that the inverse of a matrix $\mathbf{X}\mathbf{X}^T$ must be calculated, which may be singular or nearly singular and thus infeasible. To avoid such a situation, Tikhonov regularization with $\mathbf{\Gamma} = \alpha\mathbf{I}$ will be used to stabilize the inverse matrix, changing the formula to

$$\mathbf{B}_\tau \ = \mathbf{L}_\tau \mathbf{X}^T (\mathbf{X}\mathbf{X}^T + \alpha\mathbf{I})^{-1} \ . \tag{4}$$

3.3. Partial Least Squares

The *Partial Least Squares* (PLS) method [6], also called Projection to Latent Structures, is a regression method designed for input data with high number of dimensions and with high collinearity. PLS does not use the input vectors as such, but constructs another set of basis vectors to do the linear regression in a new space. PLS represents the input and output data as matrix decompositions: $\mathbf{X} = \mathbf{V}_X \mathbf{S}_X$ and $\mathbf{L}_\tau = \mathbf{V}_{L\tau} \mathbf{S}_{L\tau}$. $\mathbf{V}_X$ and $\mathbf{V}_{L\tau}$ hold the new spanning vectors for input and target data, respectively. $\mathbf{S}_X$ and $\mathbf{S}_{L\tau}$ give the coordinate values, often called scores, of the input and target data with respect to their new coordinate axes. The number of spanning vectors used to construct a new space is a dimensionality parameter for PLS, and it determines the rank of the model.

After iteratively rotating the spanning vectors of the new bases, $\mathbf{V}_X$ and $\mathbf{V}_{L\tau}$, PLS finds such directions, that the score matrices $\mathbf{S}_X$ and $\mathbf{S}_{L\tau}$ are as identical as possible. Then the conversion $\mathbf{D}$ from $\mathbf{S}_X$ to $\mathbf{S}_{L\tau}$,and from that, the conversion matrix $\mathbf{B}_\tau$ for the original space can be obtained by

$$\begin{aligned} \mathbf{D}_\tau &= \mathbf{S}_{L\tau}\mathbf{S}_X^T(\mathbf{S}_X\mathbf{S}_X^T)^{-1} \quad \text{and} \\ \mathbf{B}_\tau &= \mathbf{V}_{L\tau}\mathbf{D}\mathbf{V}_X^T(\mathbf{V}_X\mathbf{V}_X^T)^{-1} \ . \end{aligned}$$

In this work, PLS has been implemented with Statistically Inspired Modification of PLS, namely the SIMPLS-algorithm [7].

4. Experiments

4.1. Experimental setup

To compare the recognition quality of our previous state label based mapping and the mapping with learned regression matrices, we used the CHiME challenge database [8]. CHiME is based on the GRID corpus, where 34 speakers read aloud simple command sentences, consisting of linear grammar and a vocabulary of 51 words [9]. The GRID sentence structure is *verb-color-preposition-letter-digit-adverb*. There are 25 different letters and 10 digits. Other classes have four word options each. In CHiME database, the utterances are reverberated with a room response, and then mixed into stereo background noise sampled from a real living room. The task is to recognize words from 'letter' and 'digit' classes in 600 test utterances at six SNR levels: +9, +6, +3, 0, -3 and -6 dB. Test score is the number of correctly recognized keywords in the 1200 word instances at each SNR.

Training set of 300 sentences was used to create a dictionary, and another set of 200 sentences was used for training regression matrices.

In our test setup, we used features consisting of 26 Mel-scale spectral magnitudes for each stereo channel, sampled at 16 kHz. Frame length was 25 ms and frame shift 10 ms. Segment lengths of $T = 10$, 20 and 30 consecutive frames were used, thus the total length of a concatenated segment vector was $2 * 26 * T$ (520–1560).

4.2. Dictionaries and factorization

Two different speech dictionary types were generated. For *speaker-dependent* dictionaries, noiseless training set of each speaker was converted into partially overlapping exemplar-segments by going through the set with a random shift of 4–8 frames. The full dictionaries of approximately 10000-17000 exemplars were reduced to 5000 exemplars for each speaker / segment length combination, while maximising the flatness of included word distribution. In addition, a *speaker-independent* speech dictionary of 5000 exemplars was generated similarly for each segment length by combining 147–148 exemplars from each speaker, again with an attempt for maximally flat coverage of words and speakers. The remaining 40% of training utterances were used for learning the regression matrices. Speaker-dependent dictionaries were trained with utterances of the same speaker. Independent training used the combined utterances of all speakers.

In the factorization phase, we extracted a noise dictionary of 5000 exemplars for each utterance by picking partially overlapping pure noise segments from the immediate neighborhood of the utterance to be recognized. Speech and noise dictionaries were combined, and then re-weighted together to equal Euclidean norms over Mel bands and exemplars. The same band weights were applied to the utterance features. Factorization was performed with 300 NMF iterations, either in double precision CPU or single precision GPU computing. The difference between these was found negligible.

4.3. Decoding

The activations were converted into state likelihoods by using the label- and regression-based methods explained in sections 3.2 and 3.3. For the Tichonov regularization in OLS we used values 10^{-2}, 10^{-7} and 10^{-12}. PLS-regression was tested with dimensionalities 500, 650 and 800.

Finally, the likelihood matrices were decoded using a modified HVite binary from the HTK toolkit. Apart from the externally calculated state likelihoods, the original CHiME models were used 'as is' in decoding. Scoring of letter and digit keywords was performed with the standard CHiME scripts.

5. Results and discussion

The results of our recognition experiments are summarized in Table 1. Pane (a) shows the results for speaker-independent recognition, and pane (b) for speaker-dependent dictionaries. For each segment length T we show recognition rates for the previous binary label system, OLS regression and PLS regression. In addition, the baseline rates from CHiME documentation are shown on the topmost row [8]. The baseline system is standard HMM-based recognition using speaker-dependent GMMs. Similar speaker-independent baseline results were not available. For regression results, OLS with regularization parameter 10^{-2} and PLS with 800 dimensions were selected. In OLS, the differences between parameters were minimal. With PLS, using more dimensions improve the results slightly, but the computation burden becomes very heavy.

In speaker-independent recognition, we notice that regression provides improvements of 4.3–14.1% (absolute) in results over binary label. In general, a relative reduction of approximately 20% is present in the error rates. The likely reasons for this are twofold. First, the speaker-independent dictionaries are fairly small for this task. 5000 exemplars could suffice for covering the phonetic variation within the set. However, in the word-based labelling system, some speaker-state combinations may be underrepresented, thus similar phonetic features with different word labels may get activated. Trained regression matrices manage to overcome this problem by activating several potential states at once. The second reason for the success of regression in this case is that the matrices have been trained from combined utterances of all speakers. In other words, there is 34 times more training data than in the speaker-dependent case. Possibly due to this abundance of training data, OLS yields the highest recognition rate of the tested methods.

Overall, speaker-independent recognition does not seem to perform very well. It should be noted, though, that the baseline scores were obtained using speaker-dependent acoustic models. In CHiME data, a lot of the background noise consists of people speaking in the living room, or speech coming from television. Therefore a speaker-independent model will easily pick up inaccurate speech segments from these external sources.

The overall results of speaker-dependent recognition are substantially better. In high SNRs, approximately 90% of all keywords are recognised correctly. This is impressive, because

Table 1: CHiME test results using exemplar-based factorization and three likelihood generation methods. For each SNR and segment length T, keyword recognition percentages are shown using binary labels, OLS (10^{-2} regularization) and PLS (800 dimensions). The baseline system is CHiME reference decoder, which uses mono MFCC features and speaker-dependent GMMs.

(a) Speaker-independent results

SNR	(dB)	9	6	3	0	-3	-6
CHiME	baseline	82.4	75.0	62.9	49.5	35.4	30.3
T=10	labels	69.9	66.0	58.7	52.4	42.9	37.8
	OLS	84.3	77.8	71.4	65.3	56.4	48.6
	PLS	82.1	77.1	71.0	64.0	57.0	49.3
T=20	labels	77.3	72.8	68.2	62.7	51.1	44.0
	OLS	**85.2**	**80.5**	**78.7**	**71.1**	60.2	51.5
	PLS	82.9	78.8	74.8	70.1	59.5	50.6
T=30	labels	76.0	73.5	68.2	61.8	52.7	44.7
	OLS	82.8	**80.5**	76.3	70.7	**62.1**	**54.4**
	PLS	81.1	77.8	74.3	68.8	61.1	52.4

(b) Speaker-dependent results

SNR	(dB)	9	6	3	0	-3	-6
CHiME	baseline	82.4	75.0	62.9	49.5	35.4	30.3
T=10	labels	91.3	88.3	85.8	80.8	71.4	62.3
	OLS	89.8	86.8	85.0	79.7	70.1	62.7
	PLS	90.5	87.8	84.5	80.2	71.3	63.7
T=20	labels	91.6	89.2	87.6	84.2	74.7	68.0
	OLS	91.1	**90.0**	**88.5**	**85.2**	77.6	69.2
	PLS	**91.9**	89.3	88.2	85.0	**78.6**	**69.6**
T=30	labels	88.8	88.1	86.3	82.9	75.1	68.3
	OLS	88.8	86.0	86.4	83.3	76.1	69.2
	PLS	89.1	85.7	84.8	82.4	77.2	68.8

especially the letters are easily confused even by human listeners [9]. Noisy results are also convincing, with an increase of $\approx$ 30–40% (absolute) over the baseline at lower SNRs. Segment length 20 appears optimal in its combination of initial recognition rate and noise robustness.

In this scenario, the results of regression-based likelihood conversion are mixed. For this test set size, the differences between binary labels, OLS and PLS cannot be considered significant with sufficient confidence. The original labeling system performs well, because the speech dictionary only covers one speaker at a time and thus can contain a close approximation of almost every speech pattern of the current speaker. After sufficiently many NMF iterations, a reliable estimate of the underlying word is usually discovered, regardless of partial phonetic ambiguity. As the conversion errors are few to begin with, there is little to gain with regression. Still, we notice that both algorithms appear to produce slight improvements in the noisy end. The performance of OLS and PLS is mostly similar. It should be noted that OLS results were generally identical for all parameter sizes, while PLS performance depended on the dimension parameter. More careful tuning of it for different segment lengths would probably make it superior to OLS.

The training data set available for regression matrix learning was notably small in speaker-dependent recognition. After dictionary generation, only 200 training utterances were available per speaker. Therefore especially letter likelihoods had to be learned from only a few word instances. Comparing the results to the speaker-independent case, we can theorize that the advantages of PLS are higher, when training data is scarce. If this is not the case, OLS may be a better choice due to its lower computational cost for similar or higher quality.

6. Conclusions

A new, learning-based method was proposed for mapping speech exemplar activations into state likelihoods in automatic speech recognition. By training the conversion matrices with regression algorithms, it is possible to automatically handle phonetic ambiguity and resulting labeling problems of dictionary exemplars. Furthermore, the algorithms allow use of learned or synthetic exemplars without previous knowledge of their linguistic content.

The methods were tested using noisy speech utterances from the CHiME challenge corpus. In speaker-independent recognition, where the original state labeling was unreliable while regression training data was plentiful, all results improved by 4.3–14.1% in comparison to the previous labeling system. In speaker-dependent recognition, no significant increase or decrease was present. We conclude that automatically learned mapping can match or surpass the recognition quality of explicitly assigned state labels. Ordinary Least Squares regression was found straightforward and reliable for the purpose. Partial Least Squares requires careful parameter selection, but it may yield higher results especially for limited training data.

7. References

[1] L. Deng and H. Strik, “Structure-based and template-based automatic speech recognition - comparing parametric and non-parametric approaches,” in *Proc. INTERSPEECH*, 2007.

[2] T. N. Sainath, A. Carmi, D. Kanevsky and B. Ramabhadran, “Bayesian Compressive Sensing for Phonetic Classification,” in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, U.S.A., 2010.*

[3] J. F. Gemmeke, T. Virtanen and A. Hurmalainen, “Exemplar-based sparse representations for noise robust automatic speech recognition,” accepted for publication in *IEEE Transactions on Audio, Speech and Language processing*, 2011.

[4] G. S. V. S. Sivaram, S. K. Nemala, M. Elhilali, T. D. Tran and H. Hermansky, “Sparse Coding for Speech Recognition,” in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, U.S.A., 2010.*

[5] J. F. Gemmeke, T. Virtanen and A. Hurmalainen, “An exemplar-based framework for noise-robust automatic speech recognition,” in *Proc. INTERSPEECH 2010*, Makuhari, Japan, 2010.

[6] P. Geladi and B. R. Kowalski, “Partial Least-Squares Regression: a Tutorial,” in *Analytica Chimica Acta*, vol. 185, no. 1, 1986.

[7] S. de Jong, “SIMPLS: An alternative approach to partial least squares regression,” in *Chemometrics and Intelligent Laboratory Systems*, vol. 18, issue 3, Mar. 1993.

[8] H. Christensen, J. Barker, N. Ma and P. Green, “The CHiME corpus: a resource and a challenge for Computational Hearing in Multisource Environments,” in *Proc. INTERSPEECH 2010*, Makuhari, Japan, 2010.

[9] M. Cooke, J. Barker, S. Cunningham and X. Shao, “An audio-visual corpus for speech perception and automatic speech recognition,” in *Journal of the Acoustical Society of America*, 120(5), 2006.

Publication III

MUSIC DEREVERBERATION BY SPECTRAL LINEAR PREDICTION IN LIVE RECORDINGS

Katariina Mahkonen[1], Antti Eronen[2], Tuomas Virtanen[1],
Elina Helander[1], Victor Popa[1], Jussi Leppänen[2], Igor D.D. Curcio[2]

[1] Department of Signal Processing Tampere
University of Technology,
Tampere, Finland
firstname.lastname@tut.fi

[2] Nokia Research Center
Tampere, Finland
firstname.lastname@nokia.com

ABSTRACT

In this paper, we present our evaluations in using blind single channel dereverberation on music signals. The target material is heavily reverberated and dynamic range compressed polyphonic music from several genres. The applied dereverberation method is based on spectral subtraction regulated by a time-frequency domain linear predictive model. We present our results on enhancing music signal quality and automatic beat tracking accuracy with the proposed dereverberation method. Signal quality enhancement, measured by improvement in signal to distortion ratio, is achieved for both reverberant and dynamic range compressed signals. Moreover, the algorithm shows potential as a preprocessing method for music beat tracking.

1. INTRODUCTION

Reverberation is a phenomenon of sound energy persisting within a space due to a multitude of echoes from surrounding surfaces. Reverberation impacts the coloring of sounds. Usually the early reflections of the sound due to walls and other reflectors around are considered comfortable for human perception. Therefore, concert halls are designed to have some amount of reverberation and artificial reverberation is used as an artistic effect in music production. However, under heavy reverberation, the intelligibility of speech [1] and pleasantness of music decreases. Furthermore, the accuracy of automatic audio analysis algorithms decreases [2, 3].

The process of suppressing reverberation within audio signals is called dereverberation or acoustic channel equalization. When there is no information about the acoustic impulse response (AIR), dereverberartion is named blind. There are both time-domain [4,5] and frequency-domain [4,6,7,8,9] techniques for this task available. Time-domain techniques aim at estimating an AIR, and suppressing the echoes using signal deconvolution. There are however several problems in this approach although it is theoretically appealing. Estimating the AIR and its frequency domain representation, the acoustic transfer function (ATF), is fairly difficult as ATF is generally not minimum phase. Also it is very sensitive to even small deviations to recording geometry, thus in all practical cases it must be considered time-varying, and ideally infinite AIR must be estimated as a finite sequence. However, many techniques, for example [11], use this approach to remove few early reflections and use a spectral technique for suppressing late reverberation part. The methods operating purely in frequency domain have adopted the idea of spectral subtraction and attenuate amplitudes within spectral bands according to some criterion.

Most of the dereverberation studies conducted so far have considered speech signals, aiming at increasing both the intelligibility of speech for humans and automatic speech recognition (ASR) performance [4]. Some studies have included music signals in evaluations of applied dereverberation algorithms [5, 6], and few experiments have focused primarily on music signals [7, 8]. Wilmering & al. have explored using dereverberation as a preprocessing step for music onset detection and musical instrument recognition [8]. For evaluation they used single-instrument recordings generated from MIDI. In [7], Yasuraoka & al. have performed music dereverberation on monophonic musical recordings and evaluated the results with the log spectral distance improvement (LSDI). However, such a measure, which uses only the spectral magnitude does not take into account the phase disturbances, which are a very common source of artifacts in framewise audio processing.

Our goal in this work is to discover whether the chosen dereverberation method is effective in processing polyphonic musical signals which are deteriorated by dereverberation and subjected to dynamic range compression (DRC). Many dereverberation algorithms, including the proposed one, are based on linear prediction (LP), which assumes that reverberation is a linear process. However, DRC is often applied to audio recordings, and as it is a nonlinear operation, it potentially disturbs LP-based dereverberation algorithms.

We analyze how well the applied dereverberation method can suppress the reverberation in terms of improvement in the signal to distortion ratio (SDR) [10]. We compare the achieved SDR improvement (SDRI) for the signals initially deteriorated by reverberation and the same signals after subjecting them to DRC to see how the DRC affects the dereverberation performance. We also test how the dereverberation affects the accuracy of automatic beat tracking when used as a preprocessing step.

This paper is organized as follows. In Section 2 the used dereverberation method is explained. Section 3 describes the performed evaluations, whereas the results are presented in Section 4. Discussion and conclusions complete the paper in Section 5.

2. METHOD

In this section, the proposed dereverberation method, which is adopted mainly from [11], is introduced. The mathematical model for the reverberation, model parameter estimation and the methods used for dereverberating the observed signal are described. This is followed by a brief description of the beat tracking method used.

2.1. Spectral Subtraction via Linear-predictive model

The music signal is processed in successive frames which are partially overlapping and smoothed with a Hanning window. The spectrum of each frame is computed with the discrete Fourier transform (DFT). The model we use to describe the spectral magnitude $|X(n,f)|$ of the reverberant signal in each frame n and each frequency f is formulated as

$$|X(n,f)| = |S(n,f)| + |R(n,f)|.$$

$|S(n,f)|$ and $|R(n,f)|$ are respectively the spectral magnitudes of the clean source signal and the reverberation noise part in the observed signal in the same time-frequency-bin. The reverberation part of the signal within each frequency f is modeled by a linear predictive system

$$\left|\hat{R}(n,f)\right| = \sum_{p\epsilon P} a_f(p)\, |X(n-p,f)|, \tag{1}$$

where P defines a set of frame indices anterior to index n, which are considered to be involved with the late reverberation within frame n. Generally P = {1...$p_{\max}$} if the LP-model order is $p_{\max}$. However, some frames can be omitted from the full set P = {1...$p_{\max}$} to prevent subtraction of early reflections or to prevent the LP-solution from being affected by the regular beat of music. As an alternative to frequencies f given by DFT for the model, we may prefer to model the average of the spectral magnitude within some frequency bands, such as frequency bands with center frequencies spaced evenly on the perceptually motivated mel-scale. In this case we replace $|X(n-p,f)|$ in (1) with a mean spectral magnitude within a frequency band k for f = { $f_{\min}{}^{k}$ … $f_{\max}{}^{k}$ }. Then the magnitude of the reverberation noise $\left|\hat{R}(n,f)\right|$ is considered equal for all the frequencies f within the frequency band k.

The parameters $\boldsymbol{a}_f = [a_f(1), a_f(2), \ldots, a_f(p_{\max})]^{\mathrm{T}}$ of the reverberation model are estimated separately for each recording and frequency f or band k. We determined the weight vectors $\boldsymbol{a}_f$ by the standard Least Squares solution

$$\boldsymbol{a}_f = \left(V_f{}^T V_f\right)^{-1} V_f{}^T \boldsymbol{v}(p_{\max}+1,f)$$

where $V_f = [\boldsymbol{v}(p_{\max},f),\ \boldsymbol{v}(p_{\max}-1,f),\ldots,\boldsymbol{v}(1,f)\,]$ and $\boldsymbol{v}(i,f)$ is defined as $\boldsymbol{v}(i,f) = [|X(i,f)|,\ |X(i+1,f)|,\ldots,$ $|X(i+N-p_{\max}-1,\ f)|\,]^{\mathrm{T}}$. N is the number of frames in one recording. Additionally, we calculated vectors $\boldsymbol{a}_f$ with an algorithm from [15] forcing all the values to be non-negative, i.e. $\boldsymbol{a}_f(p) \geq 0$ for all p. This constraint was chosen heuristically, as a physical nature of sound energy is to decay in time throughout the spectrum almost invariably in natural environments.

In order to prevent undesirable processing effects, a frequency dependent parameter $\beta(f)$ is used in the dereverberation stage to limit the amount of dereverberation as follows

$$\left|\hat{S}\,(n,f)\right| = |X(n,f)| - \beta(f)\left|\hat{R}(n,f)\right|.$$

The complex spectrum of the dereverberated signal is generated from the dereverberated magnitude spectrum $\left|\hat{S}\,(n,f)\right|$ using the phase information from the originally observed signal spectrum as

$$\hat{S}(n,f) = \left|\hat{S}(n,f)\right| e^{-i\angle X(n,f)}\ .$$

Each dereverberated signal frame is produced via the inverse discrete Fourier transform (IDFT) and the frames n ={1 … N } are combined after IDFT by summing their contributions together with the overlap-add method.

2.2. Beat tracking

The proposed dereverberation method is also evaluated as a preprocessing method for a music beat tracker. The beat tracker combines the elements from the methods presented in [12] and [13] and is only briefly described here, highlighting the essential novel parts. Beat tracking starts by obtaining an estimate of the average tempo of the signal with the tempo estimation method of [12]. The method computes a pitch-chroma based accent signal to measure the degree of spectral change and music accentuation over time. The accent signal is processed by a generalized autocorrelation function to compute periodicity vectors, and then k–nearest neighbor regression is applied on the periodicity vectors to obtain an estimate of the signal tempo. The beat tracking step takes the tempo as an input and estimates the most likely sequence of beat times from the signal, using the effective dynamic programming routine described in [13]. Compared to the beat tracking system described in [13], this beat tracker provides superior accuracy which is attributed to the inclusion of the robust k-nearest-neighbor based tempo estimation step described in detail in [12].

An obvious way to implement the dereverberation as a preprocessing for beat tracking is to input the dereverberated signal to the beat tracker. However, this was not found to give any improvement in beat tracking accuracy on the used dataset. On the contrary, often a decrease in the accuracy was observed. Instead, it was found better to input the dereverberated signal to the tempo estimation step, and to perform the beat tracking step on an accent signal calculated from the original, non-dereverberated signal. That is, we perform the accent signal analysis described in [12] on both the original signal and the dereverberated signal. The accent signal computed from the dereverberated signal is used in tempo estimation. Then, the tempo estimate and the accent signal computed from the original signal are input to the beat tracking step.

The accent signal measures the change in the spectrum of the signal and exhibits peaks at onset locations. The goal of the beat tracking step is to find the most likely sequence of beat times, given the tempo estimate and the accent signal. Beat tracking is performed with the method described in [13]. The dynamic programming step takes as inputs the accent signal and the beat period, performs smoothing of the accent signal with a Gaussian window, and then finds the optimal sequence of beat times through the smoothed accent signal.

3. EVALUATION AND RESULTS OF SIGNAL QUALITY ENHANCEMENT

Both artificially reverberated and real-world reverberant signals were used in testing the algorithm. For objective signal quality evaluation purposes, two sets of non-echoic polyphonic musical signals were generated from tracks stored in MIDI format using the Timidity synthesis software. The tracks used were from classical, pop and jazz genres. The evaluation set consists of 13 sound segments of length from 7s to 29s, and these sounds were used for system parameter estimation. The test set consists of 31 segments of 20s to 9min in duration to be used for evaluation of the dereverberation performance. To resemble a reasonable real-life concert situation, a room impulse response (RIR) from the AIR database [14] with a reverberation time $T_{60} \approx 3$s was used for reverberating the dry music signals. To evaluate the effect of dynamic range compression, DRC with a compression ratio 3:1 above the threshold -20dB was applied. The timespans for root mean square (RMS) signal power level estimation for increasing and decreasing the DRC-gain were $\tau_{\text{attack}}=$ 5ms and $\tau_{\text{release}}=$ 200ms respectively.

As an evaluation metric for these artificially distorted signals we used the signal to distortion ratio (SDR) [10]. SDR is more suitable for evaluation of dereverberation performance than the measures operating purely in the frequency domain, such as the log spectral distance improvement used in [7]. This is due to the fact that the SDR-calculation segregates the source+early reflections part s_{clean} from the evaluated signal s in the time domain by considering s's projections to the known clean source signal s_{dry} and its slightly delayed versions. Then SDR in dB is calculated as the logarithmic energy ratio of s_{clean} and the remaining noise part as

$$\mathrm{SDR} = 10\log_{10}\frac{||s_{\text{clean}}||^2}{||s - s_{\text{clean}}||^2} \;,$$

where s is either the distorted or the dereverberated signal. The amount of signal quality enhancement was calculated as the SDR-improvement (SDRI) as

$$\mathrm{SDRI} = \mathrm{SDR}_{\text{dereverberated}} - \mathrm{SDR}_{\text{reverberant}} \;.$$

The optimal values for most of the system parameters were selected according to SDRIs given by the evaluation sound set and kept unchanged for producing the results with the test sound set. According to the SDRIs on the evaluation sound set, forcing all the linear prediction weights $\boldsymbol{a}$ to be non-negative was found beneficial. The set P={1,2,3} for the LP-model was selected as sufficient. Only with very short processing frames, say 20ms, increasing $p_{\max}$ was found beneficial. Reducing the full set from P = {$1\ldots p_{\max}$} was found to decrease the performance. Empha sizing dereverberation on certain spectral area with frequency dependent $\beta(f)$ was tested with ascending, descending and smooth window-function –like $\beta(f)$:s. None of these was found out to have strong positive effect on dereverberation result, the weight for the lowest frequencies dominated the result in every case, thus constant β –value was used in the test phase.

The results with the test set, introduced in Figure 1, show that dereverberation can be done successfully with this method. Comparisons of SDRIs when the processing frame length and the amount of dereverberation β are varied are shown in Figure 1 (a) and (b) respectively. Nonlinear DRC was found not to deteriorate the dereverberation performance. The average SDR-values prior to dereverberating are 6.1 for only reverberated and 5.2 for the reverberated and DRC-processed signals. Thus the achieved higher SDRIs for DRC-modified signals do not imply higher final SDR.

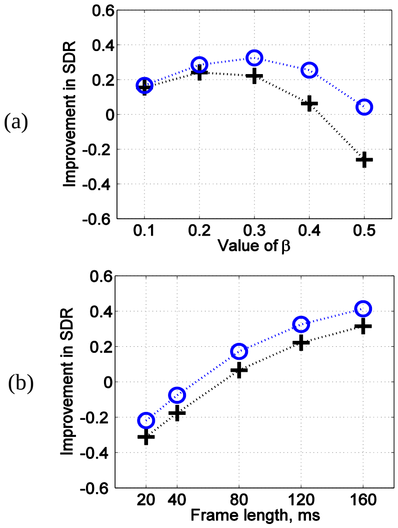


Figure 1: *Improvement in signal to distortion ratio due to dereverberating the signal, when (a) the amount of dereverberation, i.e. value of β or (b) the length of a processing frame are varied, and the rest of the system parameters are kept constant. The crosses correspond to results with signals suffering only from reverberation. The circles correspond to results with signals subjected also to DRC.*

4. EVALUATION AND RESULTS IN MUSIC BEAT TRACKING

The dataset for testing the effectiveness of the method as a preprocessing step for beat tracking comprises 113 musical excerpts captured with mobile devices in live situations. The music material is mainly from mainstream pop, rock, and dance genres, with a few salsa and progressive tracks. The amount of reverberation in the recordings varies from highly reverberant to not reverberant and also the amount of DRC varies. It is desired that the method should not decrease the beat tracking accuracy even if reverberation is not present, and therefore also non-reverberant examples are included. Some of the signals contain distortion and noises from the audience, presenting a very challenging scenario for beat tracking.

The ground truth beat annotations were input by human experts by tapping along to the pieces. The beat tracking accuracy is measured with the performance criteria described in more detail in [15]. "Correct" denotes the percentage of pulse estimates where both the period and phase are correct within a 17.5 % tolerance. "Accept d/h" allows consistent tempo halving and doubling whereas "Period correct" ignores the phase and considers only the period, i.e., the tempo.

The results for beat tracking are depicted in Table 1. The baseline denotes the results of the beat tracking method when no dereverberation is applied, and the results in the row "Dereverberated" denote the results when the tempo estimation is performed on the dereverberated signal. The results are shown for the best parameter combination, where the length of the processing frame was 120ms, P = 1, the number of mel-frequency bands K used for estimating the reverberation $|R(n,k)|$ for k=1...K is 128, and β = 0.2. This parameter combination was obtained by varying the dereverberation method parameters and running the system on the complete dataset, using the beat tracking accuracy as the parameter selection criterion. From the results, a small improvement is observed, indicating potential of the method as a preprocessing stage for tempo estimation and beat tracking in reverberant conditions.

Table 1: *Results of beat tracking.*

	Correct	Accept d/h	Period correct
Baseline	58%	65%	81%
Dereverberated	60%	67%	84%

5. CONCLUSIONS

The goal of this paper was to verify whether the proposed dereverberation method, which is based on spectral subtraction regulated by a linear predictive model, is effective in enhancing reverberant polyphonic music signals. The performance of the method was evaluated using a signal to distortion ratio improvement measure. Using SDRI, we also investigated the effect of dynamic range compression on dereverberation performance. The performance of the method on automatic beat tracking, when the dereverberation was done in a preprocessing step, was measured too.

The results show that music dereverberation can be achieved by this method and the presence of dynamic range compression does not deteriorate the performance. Even better, the signal to distortion ratio improvement turned out to be higher when the music had been subjected to DRC. Generally the quiet signal parts are relatively harder to dereverberate than the loud signal parts. Thus the more even dynamics of DRC-processed music is beneficial for dereverberation. Also, as the louder signal parts dominate the value given by SDR-measure, the more stationary and even dynamics of DRC-processed signals may be an asset. Anyhow, the absolute SDR-values both before and after dereverberation were lower for the signals distorted by DRC than for the signals containing only reverberation. Altogether, this is very interesting result and it gives us verification that this kind of nonlinearity is not a problem for the linear dereverberation method used.

The method shows promise for improving beat tracking accuracy in highly reverberant conditions but the improvement is too small for strong conclusions to be made. Somewhat unexpectedly, no improvement in beat tracking accuracy was observed when beat tracking was performed on the dereverberated signal, although one could have anticipated such behavior based on the reported increase in sound onset detection accuracy in [8]. A small increase in overall beat tracking accuracy was observed only when tempo estimation was performed on the dereverberated signal while performing the beat tracking on the original signal. A possible explanation for this behavior is that the dereverberation is successful in enhancing the pulse sensation in music. Indeed, informal listening experiments indicate that the beat pulse is slightly better audible in the highly reverberant signals after dereverberation, which may explain why it helps the tempo estimation. However, since the beat tracking accuracy is not improved if performed on the dereverberated signal, the dereverberation may be causing too much artifacts on the temporal accent signal shape for the beat positioning accuracy to improve.

6. REFERENCES

[1] M. Klatte, T. Lachman and M. Meis, “Effects of noise and reverberation on speech perception and listening comprehension of children and adults in a classroom-like setting”, *in Noise and Health*, Vol. 12, Issue 49, 2010, pp. 270-282.

[2] T. Wilmering, G. Fazekas and M. B. Sandler, “The effects of Reverberation on Onset Detection Tasks”, *Audio Engineering Society Convention 128*, London, UK, May 2010.

[3] T. Virtanen, R. Singh and B, Raj, *Techniques for Noise Robustness in Automatic Speech Recognition*, pp. 42-43, Wiley, 2012

[4] P.A. Naylor, N.D. Gaupitch, “Speech Dereverberation*”, Signals and Communication Technology*, Springer, London 2010.

[5] T. Okamoto, Y. Iwaya and Y. Suzuki, “Wide-band dereverberation method based on multichannel linear prediction using prewhitening filter”, *in Applied Acoustics*, Vol.73, Nr. 1, 2012, pp. 50-55.

[6] A. Tsilfidis and J. Mourjopoulos, “Blind single-channel suppression of late reverberation based on perceptual reverberation modelling”, *Journal of the Acoustical Society of America*, Vol. 129, Issue 3, March 2011.

[7] N. Yasuraoka, T, Yoshioka, T. Nakatani, A. Nakamura, H. G. Okuno, ”Music dereverberation using harmonic structure source model and Wiener filter”, *IEEE International Conference on Acoustics, Speech and Signal Processing,* 2010, pp. 53-56.

[8] T. Wilmering, M. Barthet and M. B. Sandler, “Dereverberation of Musical Instrument Recordings for Improved Note Onset Detection and Instrument Recognition”, *Audio Engineering Convention 131*, New York USA, October 2011.

[9] K. Lebart, J.M.Boucher, P.N.Denbigh, “A new nethod Based on Spectral Subtraction for speech Dereverberation”, *Acta Acustica*, Vol 87, 2001, pp. 359-366.

[10] E. Vincent, R. Gribonval and C. Févotte, “Performance measurement in Blind Audio Source Separation”, *IEEE Transactions on Audio, Speech and Language,* Vol. 14, Nr. 4, 2006.

[11] K. Furuya and A.Kataoka, “Robust speech dereverberation using multichannel blind deconvolution with spectral subtraction”, *IEEE Transactions on Audio, Speech and Language Processing,* Vol.15, No. 5, July 2007.

[12] A. Eronen, A. Klapuri,“Music tempo estimation with *k*-NN regression” *IEEE Transactions on Audio, Speech, and Language Processing,* Vol. 18, Nr. 1, 2010, pp. 50-57.

[13] D.P.W. Ellis, ”Beat tracking by dynamic programming” *in Journal of New Music Research,* Vol. 36, Nr. 1, 2007, pp. 51-60.

[14] M. Jeub, M. Schäfer and P.Vary, “A binaural room impulse response database for the evaluation of dereverberation algorithms” *in proceedings of the 16th international conference on Digital Signal Processing,* DSP’09, Greece, 2009, pp. 550-554.

[15] A. Eronen, A. Klapuri, J. Astola,” Analysis of the meter of acoustic musical signals”, *IEEE Transactions on Audio, Speech, and Language Processing,* Vol. 14, Nr. 1, 2006, pp. 342-355.

[16] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems,* Prentice Hall, 1974, Chapter 23, p.161.

Publication IV

Lifelog Scene Change Detection Using Cascades of Audio and Video Detectors

Katariina Mahkonen, Joni-Kristian Kämäräinen, Tuomas Virtanen

Department of Signal Processing
Tampere University of Technology
Finland

Abstract. The advent of affordable wearable devices with a video camera has established the new form of social data, lifelogs, where lives of people are captured to video. Enormous amount of lifelog data and need for on-site processing demand new fast video processing methods. In this work, we experimentally investigate seven hours of lifelogs and point out novel findings: 1) audio cues are exceptionally strong for lifelog processing; 2) cascades of audio and video detectors improve accuracy and enable fast (super frame rate) processing speed. We first construct strong detectors using state-of-the-art audio and visual features: Mel-frequency cepstral coefficients (MFCC), colour (RGB) histograms, and local patch descriptors (SIFT). In the second stage, we construct a cascade of the trained detectors and optimise cascade parameters. Separating the detector and cascade optimisation stages simplify training and results to a fast and accurate processing pipeline.

1 Introduction

Wearable devices with a video camera, such as Google Glasses, are becoming commodity hardware and it seems that consumers are willing to push personal blogs even further: video and audio logging of their lives from the first-person view (Figure 1), *lifelogs*. The lifelog applications have recently become under active investigation, but it is still unclear how to adopt and adapt the existing video processing techniques. In addition, a huge number of video streams and on-device processing require computationally economic but fast methods.

One important application for lifeloggers is to automatically annotate important moments which can be quickly stored, indexed and shared. This can be achieved by condensing important moments into a short "skim" and thus research on video skimming (summarisation/abstraction) has recently gained momentum [1, 2]. The best skimming methods are too heavy for on-device processing, but their core components, such as *scene detection* which produces the smallest data pieces for skimming, scenes, may be doable. On-device scene change detection can help fast (on-line) generation of summaries. State-of-the-art scene detection methods rely on visual information, but another important cue, audio, provides an alternative modality with orders of magnitude faster processing. Visual and audio cues are often complementary and therefore many hybrids have

Fig. 1. Video frames from our CASA2 lifelog dataset.

been proposed [3, 4]. However, these works mainly concentrate on maximising accuracy and omit potential for faster computation.

What is the best approach to combine detectors using features of varying importance and even from different modalities? In machine learning literature, a particularly suitable technique for cost (computation time) sensitive learning are detector cascades introduced by Viola and Jones [5]. State-of-the-art cascade construction methods do not operate on stages [5], but simultaneously optimise the whole cascade and its parameters using all training data at once [6–8]. That sets restrictions on used detectors while in our case they can be very different and therefore joint optimisation of the methods, cascade structure and its parameters is too complicated and slow, even impossible. In this work, we take a novel approach: we adopt the cascade structure but cast the problem as a classifier combination [9]: the detectors are trained separately as "strong detectors", then cascaded based on their complexity (audio detectors first) and finally the cascade parameters are optimised similar to expert weights in [9]. Our relaxed design results to simpler cascade construction and training, allows using pre-trained detectors by others, and still our "soft cascades" achieve fast (super frame rate) processing and superior accuracy.

2 Related Work

Detector Cascades - Viola and Jones [5] is the seminal work introducing cascades as a machine learning approach to tackle the real-time requirement in face detection. Their method operated on stages each aiming at high recall and false positives passed to the next more complicated classification stage. The approach is effective but sub-optimal and recent holistic approaches optimising detectors, cascade structure and cascade parameters simultaneously with all training data can provide better cascades [6–8]. That, however, sets requirements for the detectors which we wish to avoid in our work to be able to to exploit the best available detectors. We use all training data to train a set of binary detectors, we combine them using a free-form logical rule (a fixed cascade structure) and then optimise cascade parameters by exhaustive or beam search. In that sense our model is close to combining classifiers theory [9] adapted to cascades.

Combining Audio and Video Cues - Many novel video applications based on visual features have been proposed [10, 11], but combinations of audio and visual features seem always superior [3, 4, 12]. In contrast to the previous works, we do not explicitly engineer a combined audio-video-detector, but take a set of detectors, train them separately for the specific task, and then construct and optimise a cascade structure. That is also justified by the fact that lifelog data is different from the previously used full length movies [13], TV news [14], filmstrips [3] or the mixture TrecVid data [15]. The lifelog data is raw, abruptly moving, unedited, first-person-shot video.

Contributions – Our contributions in this work are two-fold: 1) we investigate how existing visual and audio processing methods work on lifelog data and 2) introduce "soft cascades" of strong detectors for efficient scene change detection in super frame rate. We have collected seven hours of real lifelog data and in our cascades we utilise the best performing cues from various studies: colour (RGB) histograms [16], local image patch descriptor (SIFT) based visual bag-of-words [17] and Mel-frequency cepstral coefficients (MFCCs) [18]. Our focus is on the essential low-level task in video summarising and indexing: *scene detection* [19]. We also report interesting results for shot detection [20]. We point out the following interesting findings:

- In lifelog analysis, audio cues seem to be much more important than in the previous works on movie or broadcast videos (e.g., the TRECVid campaign [20]) or camcorder recorded home videos. In our experiments, visual cues often fail in scene detection.
- Video cues, however, provide partially complementary information to audio, and that can be used to boost the detection accuracy without too much computational increase using the soft decision cascade paradigm proposed in this work.
- The cascades can be constructed easily by separating the detector parameter and cascade parameter optimisation into two separate stages.

3 Decision Cascades

The general goal of constructing cascades is to find a set of detectors (nodes) that minimise a target function consisting of penalties for accuracy loss, (computational) cost of evaluating nodes, and a regularisation term to avoid overfitting [6]. Optimisation of such target function requires inter-operability of the detectors, for example, access to the internal decision tree nodes in [6], i.e. cascade methods operate on "weak classifiers". In our case, we may have N very different type of detectors pre-trained for the same task and we wish to explicitly cascade them into the computationally fastest order. A same detector may appear multiple times but its execution is needed only once. In that sense, our approach is not consistent with the assumptions with the cascading works [6–8], but more resembles the combining classifiers ideology [9] where "strong classifiers" are trained and their combination weights optimised. Our combination, however, is a cascade structure and weights are detection thresholds.

A strong detector cascade is constructed by combining N detectors $D_i, i = 1, \ldots, N$ that map an input feature space X to a decision $t \in T$ in the decision space T. For simplicity, we may assume that t is binary, i.e. $D : X \rightarrow \{0, 1\}$. Our scene change detection is also a binary task. A cascade can be represented as a logical function, such as

$$D = (D_1 \cap D_2 \cap \ldots) \tag{1}$$

or

$$D = (D_1 \cup D_2 \cup \ldots) \tag{2}$$

or any disjunction of conjunctions. It is clear, that significant computational improvement can be achieved if the detector D_i returns 0 in (1) or 1 in (2), since then execution of D_j for $j > i$ is then unnecessary. In particular, if the detectors are indexed such that the computational complexities increase, $\Omega(D_i) \ll \Omega(D_j)$ for $i < j$, then the cascade can provide remarkable computational speedup.

The problem is that there are a large number of ways to combine outputs of the detectors, especially when the number of detectors increase, and only a few are optimal for a certain task. Moreover, the optimal configuration does not only mean an optimal form of the logical function D, but also optimal values for each detector's internal parameters $\mathbf{\Theta}_i$ and cascade parameters $\mathbf{\Phi}$ (detection thresholds). The only approach guaranteeing the global optimum is the exhaustive search which easily becomes unfeasible. We propose a doable but still effective optimisation procedure by the following assumptions:

- The cascade structure is built such that the complexity of detectors increase gradually: the computationally lightest detector first and heaviest last.
- The detectors are pre-trained: the parameters $\mathbf{\Theta}_i$ are optimised independently for the given task.
- The task is to optimise the cascade parameters $\mathbf{\Phi}$ for the fixed structure and pre-trained detectors.

The first assumption is justified by the fact that it can provide the lowest computational complexity for similar performance. If the detectors mutually correctly detect (in case of D as in Eq.(2)) and leave undetected (in case of D as in Eq.(1)) the same part of the input space, the detection performance can be even improved in addition to saving in the computational load. The second and third assumptions are justified by the fact that since the exhaustive search is not feasible, a separate optimisation of each detector still provides the best average performance and their mutual relationship is compensated on the cascade level parameter optimisation. A greedy algorithm for the optimisation is given in Algorithm 1. The algorithm is in the sense greedy that it moves thresholds one by one always selecting the threshold that provides the smallest amount of negative examples while including one more positive example. This iteration is repeated until all positive examples are covered.

4 Audio and Visual Cues

For our cascade construction in Section 3 we only need that a selected classifier outputs classification scores for tested example (y_n^i). For scene detection we

```
Data: Target class classification scores $y_n^i$, for $i = 1 \ldots M$ data points and $N$
      classifiers; logical cascade expression in the disjunctive normal form;
Result: Precision($P$) – recall($R$) curve and cascade parameters $\mathbf{\Theta}$ for every
        point on it.
Init: $\mathbf{\Theta} = [\theta_1, \theta_2, \ldots, \theta_N] = [\infty, \infty, \ldots, \infty]$; // P=R=0
while $R < 1$ do
    for each conjunctive ($\wedge$) part of the cascade do
        Find the new thresholds $\hat{\theta}_j$ for participating sub-classifiers $\mathcal{D}_j$ that
        select one new positive example and count the number of negative
        examples introduced.
    end
    Set $\theta_j \leftarrow \hat{\theta}_j$ based on the component providing the smallest amount of
    negative examples;
    Store $\mathbf{\Theta}$;
    Compute and store $P$ and $R$ ;
end
```

selected the audio and visual cues most successful in earlier works. These cues are shortly reviewed next.

4.1 MFCC Detector

As audio features we use Mel-frequency cepstral coefficients (MFCC) [18] which have proved to be useful in many audio information retrieval tasks like speech recognition [21], audio event detection [22, 23] and music information retrieval [24].

The audio track is analysed in successive, non-overlapping frames (not to be conflicted with video frames). From an audio frame at time t, one MFCC-vector $\mathbf{x}(t)$ of length $D_{\mathbf{x}}$ is extracted. The context change with MFCC cut detector is measured according to changes in distributions of vectors $\mathbf{x}$. A mean $\mu_d(t)$ and variance $\sigma_d(t)$ of each MFCC, indexed by d, is calculated within a sliding audio frame sequence of length T_s preceding time t. A distance between consecutive audio frames, $L_{\mathrm{MFCC}}(t)$, for scene and shot change detection at time t is then given by

$$L_{\mathrm{MFCC}}(t) = \sum_{d=1}^{D_{\mathbf{x}}} \left| \frac{\mu_d(t) - \mu_d(t+T_s)}{\sigma_d(t) + \sigma_d(t+T_s)} \right|^2 \tag{3}$$

that is slightly different to Fisher's linear discriminant, but found better in our experiments.

4.2 Colour (RGB) Detector

Despite of its simplicity, variants of colour (RGB) histogram distance have been used in the most state-of-the-art shot detection methods [20] and since it is also one of the computationally cheapest visual features it was selected for our experiments. An RGB histogram is computed from each video frame. The histogram vector $\mathbf{h}(t)$ of the frame t is of length 192, containing the incidence frequencies of pixel values 1-64 on red, green and blue channel.

A distance $L_{\mathrm{RGB}}(t)$ of two consecutive RGB frames is calculated as

$$\begin{aligned} L_{\mathrm{RGB}}(t) = &\left| \varDelta_{\mathbf{h}}(t) - \overline{\varDelta_{\mathbf{h}}(t-1)} \right| \\ &+ \left| \varDelta_{\mathbf{h}}(t) - \overline{\varDelta_{\mathbf{h}}(t+T_{\mathbf{h}})} \right| . \end{aligned} \tag{4}$$

The idea is, that gradual change is a natural way of RGB histogram evolving. Thus we compare the L_1 change $\varDelta_{\mathbf{h}}(t) = \|\mathbf{h}(t) - \mathbf{h}(t-1)\|_1$ between RGB-histograms of consecutive frames to running average change $\overline{\varDelta_{\mathbf{h}}(t-1)}$ over $T_{\mathbf{h}}$ preceding frames to see whether the view has changed entirely instead of natural evolution. The second term in (4) accounts for comparing the current change to the forthcoming video frames respectively (not available for on-line processing).

4.3 SIFT Bag-of-Words Detector

This approach is computationally much slower than MFCC and RGB based detectors, but it has been the mainstream approach in detection of visual object classes [25, 26]. At the core of this method are histograms of codes of local patch descriptors (SIFT) extracted from each video frame. For patch encoding, a visual codebook must be constructed from extracted descriptors. It has been reported that specific codebooks constructed from the input video perform much better than general codebooks and therefore this approach was adopted by us. The codebook is constructed from k-means clustering with a fixed k (codebook size). For each video frame, SIFT descriptors are extracted on a dense grid, assigned to the best matching codes, and the histogram of codes computed and used as a feature. To compute a shot or scene change score at time t from SIFT-histograms $\mathbf{b}$, a plain L_1-distance $L_{\mathrm{BOW}}(t) = \|\mathbf{b}(t) - \mathbf{b}(t-1)\|_1$ is used. Overall, the L_1 distance instead of the Euclidean distance for evaluating the difference between consecutive histograms, both RGB and BoW detectors, worked clearly best. The settings were selected based on the best found in unsupervised image classification using SIFT bag-of-features [27].

5 Experiments

Data, experiments, performance measures, and results for the selflog video scene detection are reported in this section. Since the same method also applies for shot detection (camera switched) we also report our shot detection results.

5.1 Captured selflog data set

We have collected over 7 hours of video data for our evaluations (Fig. 1). The videos were shot with a small spy camera with the frame rate 15 frames/second and frame size of 176x144 pixels. The frames are YUV420p encoded with h263 compression and stored in an mp4 container. The stereo sound tracks are recorded by a pair of in-ear microphones with 44.1kHz sampling rate and stored without compression.

The database contains video from 23 different types of environments (scenes), 6 - 16 shootings from each: amusement park, basketball game, beach, bus, cafeteria, inside car, family yard, football game, hallway, home, inside train, nature, office, outdoor festival, outdoor market, party, pub/club, railway station, restaurant, shop, sports event, at street and track'n'field.

The video was annotated for shot and scene detection. Shot detection corresponds to the situation that the scene remains the same, but the camera was turned off and turned back on in a different location in the same scene. The scene change corresponds to the situation that the user moves to another environment.

In our evaluation, the automatically found change points were compared to the known true scene and shot change times (groundtruth). If the found change point was within 0.25 seconds from a true change point, the detection was assigned correct.

5.2 Performance Measures

To compare the performances of the used shot and scene detection methods, we use precision, $P = tp/(tp + fp)$, recall $R = tp/N$ where tp stands for the number of correct shot or scene changes depending on the task, fp stands for the number of incorrectly identified change points and N is the total number of true change points in the video. A combination of R and P, an F-measure $F = 2 \cdot R \cdot P/(R + P)$, is also used as it simplifies comparison by describing the detection performance with a single value. We are also taking the computation time needed by different systems into account. The computation time CT is given as a number relative to the length of a video, i.e. for $CT = 1$ the system works tightly in real time.

5.3 Detector parameters

To avoid overfitting to our test data, we trained the detector parameters with separate material of home videos collected before the selflog data. The data is similar to lifelog data, but does not contain the same scenes and was recorded with a standard-quality hand-held camcorder.

In the colour histogram based RGB detector the only method parameter is the length of the time interval to calculate the average change of consecutive RGB-histograms T_{RGB}. The value $T_{\mathrm{RGB}} = 10$ video frames was found best.

In the BoW detector the main method parameter is the SIFT codebook size. We also experimented different detectors and descriptors, but the dense SIFT in the VLFeat toolbox (http://vlfeat.org) was found the best. The codebook is computed from the input data and the optimal codebook size was $D_{\mathrm{SIFT}} = 100$.

Based on experiments with the homevideo data, the following MFCC parameters were selected:

- audio window length = 80 ms
- number of Mel-frequency bands = 80
- number of MFCCs, $D_{\mathbf{x}} = 20$
- audio frame sequence length, $T_s = 10$ s

The number of Mel-frequency bands and the number of used MFCCs did not make a big difference in performance. The audio frame length and the sequence length for distribution estimation were more sensitive. Another finding was that the longer the audio window and the longer the sequence length, the better is the performance. However, to be able to detect also short scenes, these parameters were restricted.

5.4 Results

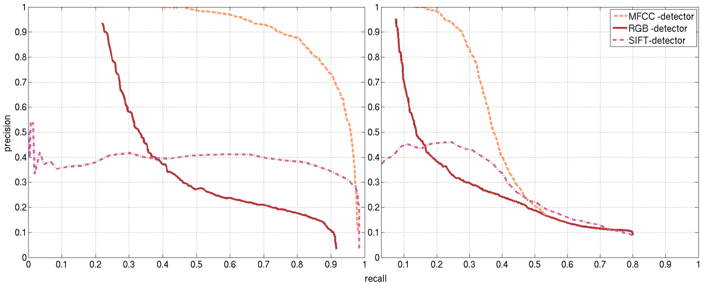


Fig. 2. Precision-recall curves for the single MFCC, RGB and SIFT detectors in scene detection (left) and shot detection (right).

Single detectors - The results of the single detectors in scene and shot detection are shown in Figure 2. It is noteworthy that all detectors have very different behaviour with respect to precision and recall. The striking result, however, is that for selflog data the audio cue outperforms the both visual cues with clear margins and being more prominent in scene detection where it is almost twice better. The result is quite opposite to state-of-the-art results with pre-edited material such as movies and TV programs [13, 14, 3, 15].

Detector Cascades - The results for various cascades are shown in Table 1 including the single detectors. The single audio MFCC detector performs surprisingly well (F-score: 0.84), but as indicated by the different behaviour of the single precision-recall curves in Figure 2 the other detectors also provide strong complementary information about scene changes. This is evident as the optimal relationship is AND ($\cap$) and for the two cascades MFCC and RGB and MFCC and SIFT the results are 0.90 and 0.95: when two detectors make a wrong decision the third corrects it. Note that for the both cases the computation time

Table 1. Selflog scene and shot detection cascade performances. Performances are reported as the best F-scores in the precision-recall curve with the corresponding cascade computing time (CT) (processing time in seconds per second of video).

Cascade	Scene detection		Shot detection	
	F-score	CT (s/s)	F-score	CT (s/s)
MFCC only	0.84	0.01	0.46	0.01
RGB only	0.40	0.43	0.31	0.43
SIFT only	0.52	184.00	0.37	184.00
MFCC ∪ RGB	0.85	0.44	0.46	0.44
MFCC ∩ RGB	0.90	0.02	0.49	0.03
MFCC ∪ SIFT	0.84	184.00	0.45	184.00
MFCC ∩ SIFT	0.95	0.30	0.51	0.81
RGB ∩ SIFT	0.68	1.30	0.42	1.30
MFCC ∩ RGB ∩ SIFT	0.91	0.30	0.48	0.38
(MFCC∩RGB) ∪ (MFCC∩SIFT)	0.96	0.30	0.52	0.31
(MFCC∩RGB) ∪ (MFCC∩SIFT) ∪ (RGB∩SIFT)	0.96	0.30	0.53	0.32

is 2× faster than real-time (super frame rate). The best scene detection accuracy is F-score 0.96 which is achieved with classifiers trained with completely separate data and only optimising the cascade parameters. The resulting classifier is a disjunction of the two available strong conjunctions and achieves the performance with computation time 0.30 seconds needed to process 1.0 seconds of video input ($>$ 3× frame rate). It is noteworthy that the SIFT detector is essential for the performance while it is active only in very few cases as apparent by comparing its single detector and cascade detector computing times.

The same findings hold also for shot detection (best single 0.46, best cascade 0.53) which is much more difficult task in the case of lifelog data.

It should be noted that the selection of cascade parameters is not critical for good performance, since they mutually compensate each other providing smooth and intuitive performance change.

6 Conclusions

The ultimate goal of our work is fast streaming, storing, indexing, retrieval and sharing of selflog video produced by millions of users using their wearable video capturing devices. Past research on video analysis has provided effective but often too slow methods for the above tasks. In this work, we sought to improve the existing techniques with the help of two hypotheses: *multiple video modalities* provide complementary information and *cascade type processing* improves efficiency. The both assumptions were found valid in our experiments where scene and shot detection from real lifelog recordings of more than seven hours were investigated. The strikingly important role of audio, complementary of audio and video, and finally the optimised cascade structure provided us superior detection accuracy in super frame rate. These results indicate that cascades are the tools of future, fusing even more modalities (GPS, accelerometer, gyroscope, compass, barometer, proximity etc.) can be beneficial, and computationally light methods can be constructed from the existing methods. In our future work, we will follow these findings and investigate a light-weight cascade for on-line video skimming and scene indexing.

References

1. Gygli, M., Grabner, H., Riemenschneider, H., Gool, L.V.: Creating summaries from user videos. (2014)
2. Zhao, B., Xing, E.: Quasi real-time summarization for consumer videos. In: Proc. of the CVPR. (2014)
3. Kyperountas, M., Kotropoulos, C., Pitas, I.: Enhanced eigen-audioframes for audiovisual scene change detection. **9** (2007)
4. Song, Y., Zhao, M., Yagnik, J., Wu, X.: Taxonomic classification for web-based videos. In: Proc. of the CVPR. (2010)
5. Viola, P., Jones, M.: Robust real-time face detection. Int J Comput Vis **57** (2001)
6. Chen, M., Xu, Z., Weinberger, K., Chapelle, O., Kedem, D.: Classifier cascade for minimizing feature evaluation cost. In: AISTATS. (2012)
7. Wu, T., Zhu, S.C.: Learning near-optimal cost-sensitive decision policy for object detection. In: ICCV. (2013)
8. Shen, C., Wang, P., Paisitkriangkrai, S., van den Hengel, A.: Training effective node classifiers for cascade classification. Int J Comput Vis **103** (2013) 326–347
9. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classfiers. IEEE PAMI **20** (1998)
10. Wang, M.: Movie2comics: Towards a lively video content presentation. IEEE Transactions on Multimedia **14** (2012) 858–870
11. Yip, S.: The automatic video editor. In: ACM Multimedia. (2003) 596–597
12. Chen, S.C., Shyu, M.L., Liao, W., Zhang, C.: Scene change detection by audio and video clues. In: ICME (2). (2002) 365–368
13. Pfeiffer, S., Lienhart, R., Effelsberg, W.: Scene determination based on video and audio features. In: Multimedia Tools and Applications. (1999) 685–690
14. Jiang, H., Lin, T., Zhang, H.: Video segmentation with the assistance of audio content analysis. In: IEEE International Conference on Multimedia and Expo (III). (2000) 1507–1510
15. Smeaton, A.F., Over, P., Kraaij, W.: Trecvid: Evaluating the effectiveness of information retrieval tasks on digital video. In: Proceedings of ACM Multimedia, New York, USA (2004)
16. Gargi, U., Kasturi, R., Strayer, S.H.: Performance characterization of video-shot-change detection methods. **10** (2000)
17. Lowe, D.G.: Distinctive features from scale-invariant keypoints. Int J Comp Vis **60** (2004) 91–110
18. Steven B. Davis, P.M.: Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing **ASSP-28** (1980) 357–366
19. Fabro, M., Boszormenyi, L.: State-of-the-art and future challenges in video scene detection: a survey. Multimedia Systems **19** (2013) 427–454

20. Smeaton, A., Over, P., Doherty, A.: Video shot boundary detection: Seven years of TRECVid activity. Computer Vision and Image Understanding **114** (2010) 411–418
21. L. R. Rabiner, B.J.H.: Fundamentals of Speech Recognition. Prentice Hall (1993)
22. Heittola, T., Measaros, A., Virtanen, T., Eronen, A.: Sound event detection in multisource environments using source separation. In: Workshop on Machine Listening in Multisource Environments, Florence, Italy (2011) 36–40
23. J.-J. Aucouturier, B. Defreville, F.P.: The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscape but not for polyphonic music. Journal of Acoustical Society of America **122** (2007) 881–891
24. Downie, J.: Music information retrieval. Annual Review of Information Science and Technology **37** (2003) 295–340
25. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Proc. of the ICCV. (2003)
26. Csurka, G., Dance, C., Willamowski, J., Fan, L., Bray, C.: Visual categorization with bags of keypoints. In: ECCV Workshop on Statistical Learning in Computer Vision. (2004)
27. Tuytelaars, T., Lampert, C., Blaschko, M., Buntine, W.: Unsupervised object discovery: A comparison. Int J Comput Vis **88** (2010)

Publication V

CASCADE PROCESSING FOR SPEEDING UP SLIDING WINDOW SPARSE CLASSIFICATION

Katariina Mahkonen, Antti Hurmalainen, Tuomas Virtanen, Joni-Kristian Kämäräinen

Department of Signal Processing, Tampere University of Technology, Finland

ABSTRACT

Sparse representations have been found to provide high classification accuracy in many fields. Their drawback is the high computational load. In this work, we propose a novel cascaded classifier structure to speed up the decision process while utilizing sparse signal representation. In particular, we apply the cascaded decision process for noise robust automatic speech recognition task. The cascaded decision process is implemented using a feedforward neural network (NN) and time sparse versions of a non-negative matrix factorization (NMF) based sparse classification method of [1]. The recognition accuracy of our cascade is among the three best in the recent CHiME2013 benchmark and obtains six times faster the accuracy of NMF alone as in [1].

***Index Terms*—** Automatic speech recognition, non-negative matrix factorization, cascade classification, cascade processing

I. INTRODUCTION

Classification based on sparse representations (SR) [2], originally invented for image processing [3], has raised to be very popular and provides state-of-the-art results in many disciplines. The model is specifically suitable for modeling data that consists of multiple sources. Recent application fields are for example classification of handwritten characters [4], [5], tracking and classification of vehicles in videos [6], MRI image analysis [7] and EEG signal analysis [8]. Some works [2], [9] no less optimize the sparse object representation specifically for classification.

In the field of audio processing SR have been also widely used, for example in audio classification [10], source separation [11] and content analysis [12]. Also, in the recent CHiME 2013 evaluation [13] the best noise-robust automatic speech recognition (ASR) results [1], [14], [15], [16] were achieved using the sparse non-negative matrix factorization (NMF) method of [1] in combination with two other methods. However, the drawback of SR acquired by iterative non-negative matrix factorization (NMF) algorithms, despite the work on faster algorithms [5], [17], is their high computational demand.

On the other hand, in the field of computer vision, for example, in face recognition [18] and in object detection [19], *cascade processing* has been succesfully used to boost the decision process. Whenever the difficulty of the classification task of the input is not known beforehand, the amount of processing can be regulated with a cascade. The simple decisions can be made with less computing while the most sophisticated methods are used at ambiguous cases.

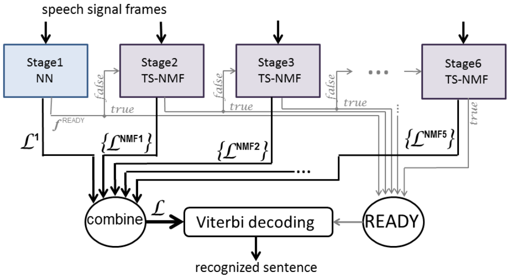


Fig. 1. Block diagram of the proposed ASR cascade.

In this work, our aim is to bring spectrogram factorization based noise robust automatic speech recognition closer to real time, while not sacrificing accuracy. Our strategy to reduce computational load is to build a cascade of classifiers (Figure 1), where the amount of computation is determined according to the interpretability of the input. The decision about instantaneous speech content can be made with simple classifiers if the certainty of the estimate is high enough. Estimation certainty assessment in automatic speech recognition has been studied e.g. in [20] and [21], but we propose a simple probability score (section III-C). For our cascade, we develop a time sparse version (TS-NMF) of the NMF method of [1]. We present also an evenly time sparse NMF (ETS-NMF) as a comparison to the cascade structure.

II. DECISION CASCADE

A decision cascade (DC) for a classification task constitutes of multiple stages where on each the confidence on the input class is evaluated and the decision about completion of the recognition process can be made. This stage-wise processing accounts for the high computational savings that are possible with a DC. A DC is able to preserve the recognition accuracy while at the same time evading redundant computation via early decisions. The effectivity of a DC results from the fact that the easily distinguishable inputs can be recognized with less processing, i.e. with fast classifiers, while heavier and the most accurate methods need to be executed only for the most ambiguous inputs. The general cascade decision process for classification is presented in Algorithm 1.

There are two functions of special importance within the algorithm, namely $f_s^{\text{READY}}(I) \in \{\textit{false}, \textit{true}\}$ and $f_s^{\text{CLASS}}(I) \in \mathbf{C}$ (class labels). f_s^{READY} is used to decide whether the decision is ready at stage s, and f_s^{CLASS} gives the class prediction at the stage s.

Algorithm 1: Decision cascade of N stages.

```
  Input: an item $\mathbf{x}$ to be classified
  Output: class decision C
1 Set READY $= \textit{false}$
2 Set S $= 0$
3 while READY $\neq \textit{true} \;\wedge\;$ S $\leq N$ do
4     S $=$ S $+ 1$
5     C $= f_{\text{S}}^{\text{CLASS}}(\mathbf{x})$
6     READY $= f_{\text{S}}^{\text{READY}}(\mathbf{x})$
7 return $C$
```

III. PROPOSED SPEECH RECOGNITION CASCADE

In ASR a speech signal is converted to sequence of words. Each word is modeled as sequence of states, and likelihoods $\mathcal{L}_t$ of states are estimated in short frames, indexed by t. Due to continuous nature of audio signal, the final class decisions are made following a hidden Markov model of the grammar by the Viterbi algorithm, in contrast to independent classification in Algorithm 1 used in [18] and [19].

The stages of the proposed cascade are used to provide increasingly accurate state likelihood estimates, which are accumulated into a state likelihood matrix $\mathcal{L}^s$. Thus the line 5 of Algorithm 1 is replaced with $\mathcal{L}_t^s = f_s^{\mathcal{L}}(\mathbf{x}_t)$, where $\mathcal{L}_t^s \in \mathbb{R}^{N_c \times 1}$ and N_c is the number of states in the grammar.

The proposed ASR cascade aims at speeding up a computationally intensive, but well performing method based on SR and NMF. The cascade works maximally at 6 stages as shown in Figure 1. The first stage uses a NN and subsequent stages use TS-NMF method up to five times to make $f_s^{\text{READY}} = \textit{true}$ if possible. The order of methods within the cascade is defined by the computation time they need to extract the state likelihood information.

Both methods in our cascade extract spectral features of the audio in a 25 ms frame after every 10 ms.

III-A. Neural Network classifier

The NN classifier at the first stage of the cascade has a topology of two hidden layers, 200 neurons each, and the output layer with N_c neurons. All the neurons use the sigmoid function. The input to the NN is formed of 40 Mel cepstrum coefficients (MFCCs) and delta MFCCs, together 80 features.

Interpretation of NN-output values as probabilities has been investigated in several works, e.g. [22], [23], [24], but we convert NN outputs $\mathbf{y}_t$ to Bayesian posterior probabilities of states as

$$\mathcal{L}_t^{\text{NN}}(c) = P\big(\mathbf{y}_{t}(c) \,|\, c\big)\; P(c) \;/\; P\big(\mathbf{y}_{t}(c)\big), \tag{1}$$

with equal priors $P(c)$. For each class $c \in \mathcal{C}$, a histogram based probability density functions (PDF) $P(\mathbf{y}(c))$ and $P(\mathbf{y}(c)|c)$ are collected from the training data.

III-B. Time Sparse NMF classifier

The later stages of our decision cascade adapt a time sparse versions (TS-NMF) of the original NMF classifier [1]. The NMF classifier processes the input signal in windows of $T = 20$ frames. Spectral magnitudes from $B = 40$ Mel bands from the T frames of a window make an input vector $\mathbf{x}_t$ of length BT for NMF classifier.

A dictionary $\mathbf{D} \in \mathbb{R}_+^{(BT) \times N_d}$ of $N_d = 10000$ such example vectors from training material is used for modeling the input as $\hat{\mathbf{x}}_t = \mathbf{D}\mathbf{w}_t$, where $\mathbf{w}_t$ holds non-negative scores of dictionary elements. The scores $\mathbf{w}_t$ are solved iteratively minimizing Kullback-Leibler divergence between $\hat{\mathbf{x}}_t$ and $\mathbf{x}_t$, which is computationally the heaviest part of the method. Half of the example spectrograms in the dictionary are taken from speech content and the other half from the noise part of training data, notated as $\mathbf{D} = \big[\mathbf{D}^{\text{speech}}, \mathbf{D}^{\text{noise}}\big]$.

State likelihood estimation from scores $\mathbf{w}_t$ is done according to equation (2). Each example vector in $\mathbf{D}^{\text{speech}}$ entails state labels for T consecutive frames. The labels are encoded as binary matrices $\mathbf{L}^d \in \{0, 1\}^{N_c \times T}$ to allow mapping the scores $\mathbf{w}_t$ to state likelihoods. An NMF state likelihood window $\mathcal{L}_t^{\text{NMF}}$ spans over time points $t \ldots t + T - 1$ and is given by

$$\mathcal{L}_t^{\text{NMF}} = \sum_{d=1}^{N_d/2} \mathbf{L}^d \cdot \mathbf{w}_t(d). \tag{2}$$

In this work we are targeting to reduce computational load, while not giving up the accuracy achievable with a computationally heavy method. The NMF in [1] performs the classification with overlapping windows where an NMF window is factorized for each frame $t = 1, 2, 3, \ldots$. For evenly time sparse NMF (ETS-NMF) a new NMF window is factorized at uniformly spaced frame indices, while in TS-NMF the NMF windows to be factorized can be selected freely. When evaluating ETS-NMF, we found out that with sparsity $p = 3$, i.e. factorizing NMF windows for every third t, ETS-NMF produces enough state likelihood information to achieve the accuracy of [1]. Thus in our ASR-cascade, the NMF factorization is allowed only for every third t.

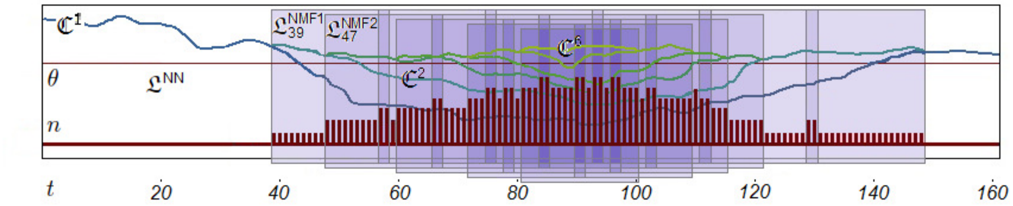


Fig. 2. Schema of constructing state likelihoods by NN - TS-NMF cascade processing. The state likelihood matrix $\mathcal{L}^{\text{NN}}$ (white background) is computed at the first stage. The colored curves represent values of $\mathfrak{C}^s$ after each stage s. The threshold θ is shown with the straight line. Where $\mathfrak{C}^s$ does not exceed θ, NMF windows (shaded rectangles) are taken into use by stage $s+1$. Red bars show the value of n_t at each t.

III-C. Cascade decisions

To decide whether the stage $s+1$ should be used to improve state likelihood estimations, the function

$$f^{\text{READY}}(\mathcal{L}^s, t, \theta) = \begin{cases} \mathit{true} & \text{if } \mathfrak{C}^s_t \geq \theta \\ \mathit{false} & \text{else} \end{cases}, \qquad (3)$$

is used. f^{READY} makes its decisions based on state likelihood matrix $\mathcal{L}^s$ and threshold θ. In (3), $\mathfrak{C}^s_t$ represents the certainty of the information in $\mathcal{L}^s$ at time point t as

$$\mathfrak{C}^s_t = \frac{1}{2l} \sum_{\tau=t-l}^{t+l-1} \max \mathcal{L}^s_\tau$$

The state likelihoods $\mathcal{L}^s_t$ are calculated as a weighted sum of likelihood information acquired from $\mathcal{L}^{\text{NN}}$ given by the NN stage and sets $\left\{\mathcal{L}^{\text{NMF}i}\right\}$ for TS-NMF stages $i = 2 \ldots s$ as

$$\mathcal{L}^s_t = (1 - \frac{n_t}{m}) \cdot \mathcal{L}^{\text{NN}}_t + \frac{n_t}{m} \cdot \left[\sum_{i=2}^{s} \sum_{\tau=1}^{T} \mathcal{L}^{\text{NMF}(i-1)}_{t-\tau+1}(\tau) \right]_{\mathbf{1}},$$

where $[\cdot]_{\mathbf{1}}$ denotes normalization to the ℓ_1 length 1. n_t is determined by the number of overlapping NMF state likelihood windows at t and $m = 12$ is used, as it gave the best results. The procedure is elucidated in Figure 2.

The selection of points τ for NMF windows $\mathcal{L}^{\text{NMF}(s-1)}_\tau$ at each TS-NMF stage s is done as follows. First, each interval of t where $f^{\text{READY}}(\mathcal{L}^s, t, \theta) = \mathit{false}$ is enlarged back- and forward by $T/2$ frames to yield target domain intervals for the new NMF windows. For each interval $\tau_\alpha \ldots \tau_\omega$, $J = \lceil (\tau_\omega - \tau_\alpha + 1)/T \rceil$ new NMF window slots $\tau_j, j = 1 \ldots J$, from U unused slots are selected if possible. The $K = U - J$ slots are left unused as evenly distributed as possible. Finally a new set of NMF factorizations is computed to produce the set of state likelihood windows $\left\{\mathcal{L}^{\text{NMF}(s-1)}_{\tau_j} \text{ for } j = 1 \ldots J\right\}$. New NMF state likelihood windows are generated at subsequent stages until $f^{\text{READY}}(\mathcal{L}^s, t, \theta) = \mathit{true} \;\forall\; t$ or the end of the cascade is encountered. In Figure 2 the set $\left\{\mathcal{L}^{\text{NMF1}}_{\tau_j} \text{ for } j = 1 \ldots 6\right\}$ produced at the second stage of the cascade is illustrated as the uppermost row of shaded NMF windows.

III-D. Utilizing state unions

In the state space of the used grammar there are many states representing the same phone in different words. For the cascade, it is more advantageous to report the likelihood of a phone instead of a designated state among the phonetically similar states. Thus, considering correlations of the NN output on training data and the states' power in discriminating words, we selected 11 groups to be used as unions. States of the grammar, marked as ‘word‘$_{\text{state}}$, within unions are $\mathcal{U}_1 = \{‘b‘_2, ‘v‘_2\}$, $\mathcal{U}_2 = \{‘b‘_3, ‘v‘_3, ‘p‘_3, ‘g‘_3, ‘d‘_3\}$, $\mathcal{U}_3 = \{‘c‘_3, ‘t‘_3\}$, $\mathcal{U}_4 = \{‘b‘_4, ‘v‘_4, ‘p‘_4, ‘g‘_4, ‘d‘_4, ‘e‘_4, ‘c‘_4, ‘t‘_4\}$, $\mathcal{U}_5 = \{‘a‘_4, ‘j‘_4, ‘k‘_4\}$, $\mathcal{U}_6 = \{‘i‘_4, ‘z‘_2\}$, $\mathcal{U}_7 = \{‘m‘_1, ‘n‘_1\}$, $\mathcal{U}_8 = \{‘m‘_4, ‘n‘_4\}$, $\mathcal{U}_9 = \{‘f‘_1, ‘s‘_1\}$, $\mathcal{U}_{10} = \{‘g‘_1, ‘j‘_1\}$ and $\mathcal{U}_{11} = \{‘q‘_4, ‘u‘_4\}$.

In $\mathcal{L}^s_t$ the likelihoods of the states within an union are substituded with the highest of them as

$$\mathcal{L}^s_t(c \in \mathcal{U}_i) = max\left\{\mathcal{L}^s_t(c \in \mathcal{U}_i)\right\}$$

for $i = 1 \ldots 11$. The keyword accuracies of both the NMF- and NN-recognizers outside the cascade when using state unions are reported in the experiments (Table I).

IV. EVALUATION

The evaluation is done using CHiME2013 automatic noisy speech recognition challenge track 1 data [13], which consists of utterances from 34 speakers in highly non-stationary background of domestic noise. Average SNR varies from -6 dB to 9 dB. The spoken sentences have strict grammar with 51 words. The state space used to represent the words is defined by the CHiME2013 challenge baseline system and has 4-10 states per word, $N_c = 250$ states in total. The speciality of this data set is the task of recognizing 'coordinates' composed of a letter and a number, e.g. 'D7'. There are 500 and 600 sentences per each SNR level in the training and evaluation set, respectively. The training data is used for training the NN and picking the example vectors for dictionary $\mathbf{D}$ of NMF. The presented recognition accuracies are achieved with the evaluation data set.

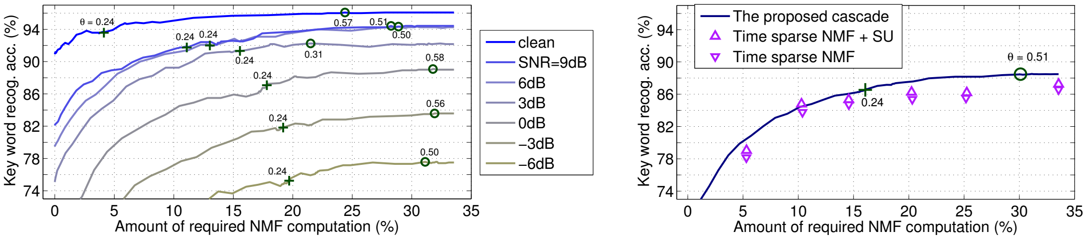


Fig. 3. The keyword recognition accuracies of the proposed cascade versus its computational load. The curves build up by changing the threshold θ of f^{READY} in eq. (3). The axes on the left show the different SNR levels separately and the average performance is shown on the right. Triangles show average accuracies of ETS-NMFp with sparsities $p = 3, 4, 5, 7, 10, 20$.

IV-A. Performance with ETS-NMF, NN and state unions

The keyword recognition accuracies on evaluation data with ETS-NMF and the used NN classifier outside the cascade are shown in Table I. The ETS-NMF classifier with time sparsity $p = 3$ utilizing state unions (SU) 'ETS-NMF3+SU' reaches recognition accuracy 87.3 % on average over all the noise conditions. Without SU post processing, 'ETS-NMF3' can be seen to reach the level of the reference 'NMF[1]'. These average accuracies of ETS-NMF3 are shown also as the rightmost triangles in Figure 3. The positive effect of utilizing state unions on ETS-NMF is 0.8 % on average.

The NN classifier of the first stage of the cascade, 'NN+B+SU' in the Table I, reaches accuracy 72.6 % on average. The positive effects of Bayesian post processing (B) of NN outputs and utilizing state unions (SU) are about 1.5 % and 0.9 % respectively.

SNR	mean	-6dB	-3dB	0dB	3dB	6dB	9dB
ETS-NMF3+SU	87.3	75.4	82.4	87.8	91.3	93.0	93.5
ETS-NMF3	86.6	75.1	82.0	87.4	89.9	92.3	92.8
NMF [1]	86.5	75.6	81.4	87.5	89.9	92.4	92.3
NN+B+SU	72.6	56.4	58.3	66.5	74.8	79.3	82.1
NN+B	71.7	55.0	57.5	65.9	73.8	78.6	81.1
NN	70.2	54.5	54.8	63.7	71.2	77.8	79.8

Table I. Keyword recognition accuracies with ETS-NMF3 and NN classifiers with and without using state unions (SU) and Bayesian post processing (B).

IV-B. Accuracy and computational load of the cascade

The operating point of the proposed cascade is defined by the threshold θ of f^{READY} in (3), which rules the usage of stages of the cascade. The threshold θ is set to achieve a desired accuracy with as small computational load as possible, or to reach as good accuracy as possible with the available computation power. Curves of keyword recognition accuracy, resulting from giving different values for θ, versus the amount of needed NMF computation as percentage of the computational load of [1] are shown in Figure 3. On these curves we pay attention specifically to two operating points. The first one, shown with a cross on each curve, is the operating point with $\theta = 0.24$. It is where the average accuracy reaches 86.5 %, the accuracy of the original NMF framework [1] requiring only 16.0 % of its computation. The second crucial operating point of the cascade, which is shown as a circle on each curve, is where the maximal keyword recognition accuracy is reached with smallest computation load. On average over all noise levels, this operating point occurs with $\theta = 0.51$ reaching accuracy 88.5 % and requiring the computation of 31 % of NMF frames.

IV-C. Comparison to state-of-the-art

The recognition accuracy of the proposed cascade ranks among the three best in CHiME 2013 challenge Track 1 results in [25]. However, an important aspect of required computational resources was not considered in CHiME 2013 evaluation. Thus in Table II we compare the results with the proposed cascade in comparison to the methods for which we can estimate the computational load: the NMF method of [1] and the winning method [26] of CHiME2013. The computation time of the CHiME2013 winner is obviously higher than NMFs as NMF [1] is one of the three methods in the winning classifier combination.

	accuracy	computation time
CHiME2013 winner [26]	92.8	$>$ 100 % *)
Proposed cascade at $\theta = 0.510$	88.5	30.9 %
ETS-NMF3	87.3	33.6 %
Proposed cascade at $\theta = 0.237$	86.5	16.0 %
NMF [1]	86.5	100 %

Table II. Keyword recognition accuracy of the proposed cascade in comparison to the baseline NMF method and the CHiME2013 challenge winning method (* utilizes the NMF method as one of its three detectors).

V. CONCLUSIONS

As automatic noisy speech recognition has proved to be hard problem to solve, the most accurate methods currently are far from real time processing. With clean speech simpler methods might do well, while with noisy environment the more advanced processing is required. A decision cascade is a way to combine these and it is a structure to consider when one wants to meet both the requirements, word accuracy and computational speed, in varying conditions. In this work we have showed that a decision cascade can be successfully applied in ASR task. Our experiments show that the accuracy of well performing NMF method for noisy ASR can be achieved with a fraction of its computation time with a decision cascade utilizing faster classifiers. In CHiME2013 keyword recognition task with our cascade utilizing a neural network and Time Sparse NMF classifiers we achieve the meritorious accuracy of [1] with less than 17 % of its computation time. The full accuracy of the cascade ranks among the three best in CHiME 2013 Track 1 challenge and it is three times faster than the winner.

VI. REFERENCES

[1] A. Hurmalainen, J. F. Gemmeke, and T. Virtanen, "Modelling non-stationary noise with spectral factorisation in automatic speech recognition," *Computer Speech & Language*, vol. 27, no. 3, 2013.

[2] K. Huang and S. Aviyente, "Sparse representation for signal classification," in *Adv Neural Inf Proc Syst*, 2006.

[3] N. Costen, M. Brown, and S. Akamatsu, "Sparse models for gender classification," in *IEEE Int. conf. Automatic Face and Gesture Recognition*, 2004.

[4] A. Makhzani and B. J. Frey, "Winner-take-all autoencoders," in *Adv Neural Inf Process Syst 28*, 2015.

[5] J. H. Friedman, "Fast sparse regression and classification," *Int J of Forecasting*, vol. 28, no. 3, 2012.

[6] X. Mei and H. Ling, "Robust visual tracking and vehicle classification via sparse representation," *TPAMI*, vol. 33, no. 11, 2011.

[7] M. Liu, D. Zhang, D. Shen, A. D. N. Initiative *et al.*, "Ensemble sparse classification of alzheimer's disease," *NeuroImage*, vol. 60, no. 2, 2012.

[8] S. B. Nagaraj, N. Stevenson, W. Marnane, G. Boylan, and G. Lightbody, "A novel dictionary for neonatal eeg seizure detection using atomic decomposition," in *Int Conf on Eng in Medicine and Biology Soc*, 2012.

[9] L. Trottier, B. Chaib-draa, and P. Giguère, "Incrementally built dictionary learning for sparse representation," in *Neural Information Processing*. Springer, 2015.

[10] R. Grosse, R. Raina, H. Kwong, and A. Y. Ng, "Shift-invariance sparse coding for audio classification," *Cortex*, vol. 9, 2012.

[11] J. Le Roux, J. R. Hershey, and F. Weninger, "Deep NMF for speech separation," in *In proc ICASSP*, 2015.

[12] C.-T. Lee, yi-hsuan Yang, and H. Chen, "Multipitch estimation of piano music by exemplar-based sparse representation," *IEEE Trans Multimedia*, no. 14, 2012.

[13] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni, "The second CHiME speech separation and recognition challenge: Datasets, tasks and baselines," in *In proc ICASSP*, 2013.

[14] J. Gemmeke, T. Virtanen, and A. Hurmalainen, "Exemplar-based sparse representations for noise robust automatic speech recognition," in *IEEE TASLP*, vol. 19, 2011.

[15] E. Yılmaz, J. F. Gemmeke, and H. Van hamme, "Noise Robust Exemplar Matching Using Sparse Representations of Speech," *IEEE/ACM TASLP*, vol. 22, 2014.

[16] J. T. Geiger, F. Weninger, J. F. Gemmeke, M. Wöllmer, B. Schuller, and G. Rigoll, "Memory-Enhanced Neural Networks and NMF for Robust ASR," *IEEE/ACM TASLP*, vol. 22, no. 6, 2014.

[17] T. Virtanen, B. Raj, J. F. Gemmeke *et al.*, "Active-set newton algorithm for non-negative sparse coding of audio," in *In proc.ICASSP*, 2014.

[18] P. Viola and M. Jones, "Robust real-time face detection," *Int J Comput Vis*, vol. 57, no. 2, 2001.

[19] T. Wu and S. Zhu, "Learning near-optimal cost-sensitive decision policy for object detection," in *IEEE Int Conf on Comput Vis*, 2013.

[20] H. Jiang, "Confidence measures for speech recognition: A survey," *Speech Communication*, vol. 45, no. 4, 2005.

[21] H. Kallasjoki, J. Gemmeke, and K. Palomaki, "Estimating uncertainty to improve exemplar-based feature enhancement for noise robust speech recognition," in *Audio, Speech, and Language Process*, vol. 22, 2014.

[22] J. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing NATO ASI Series*, vol. 68, 1990.

[23] M. Richard and R. Lippmann, "Neural network classifiers estimate Bayesian a posteriori probabilities," *Neural Computation*, vol. 3, no. 4, 1991.

[24] H. Ney, "On the probabilistic interpretation of neural network classifiers and discriminative training criteria," in *TPAMI*, vol. 17, 1995.

[25] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni, "The second chimespeech separation and recognition challenge: An overview of challenge systems and outcomes," in *IEEE workshop on Automatic Speech Recog and Understanding*, 2013.

[26] J. Geiger, F. Weniger, A. Hurmalainen, J. Gemmeke, B. Schuller, G. Rigoll, and T. Virtanen, "The TUM+TUT+KUL approach to the 2nd CHiME challenge: multi-stream ASR exploiting BLSTM networks and sparse NMF," in *The 2nd CHiME workshop on machine listening in multisource environments,*, 2013.

Publication VI

Mahkonen *et al. EURASIP Journal on Image and Video Processing* (2018) 2018:61
https://doi.org/10.1186/s13640-018-0303-9

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Cascade of Boolean detector combinations

Katariina Mahkonen*, Tuomas Virtanen and Joni Kämäräinen

Abstract

This paper considers a scenario when we have multiple pre-trained detectors for detecting an event and a small dataset for training a combined detection system. We build the combined detector as a Boolean function of thresholded detector scores and implement it as a binary classification cascade. The cascade structure is computationally efficient by providing the possibility to early termination. For the proposed Boolean combination function, the computational load of classification is reduced whenever the function becomes determinate before all the component detectors have been utilized. We also propose an algorithm, which selects all the needed thresholds for the component detectors within the proposed Boolean combination. We present results on two audio-visual datasets, which prove the efficiency of the proposed combination framework. We achieve state-of-the-art accuracy with substantially reduced computation time in laughter detection task, and our algorithm finds better thresholds for the component detectors within the Boolean combination than the other algorithms found in the literature.

Keywords: Binary classification, Classification cascade, Boolean combination

1 Introduction

Detection and binary classification are fundamental tasks in many intelligent computational systems. They may be considered as the same problem, where an input sample is to be determined into one of two groups, either one of two predefined classes, or as having some property or not. In the field of computer vision, face detection, pedestrian detection, and car detection are canonical examples that have received a lot of attention [1, 2]. Event detection from audio signal is of wide interest [3]. Detection tasks with multiple measurement modalities available are present, e.g., in biometric identity verification [4] and for medical decisions [5].

For detection of observation from a certain category, i.e., a class, many different types of detectors, trained with different data with different statistics—possibly even from different measurement modalities—are often available. Most of the detectors reported in the literature output a score, which denotes the likelihood of the existence of the quested target class, in the input data. A threshold value is then used to provide the classification "target" or "no target" for the input. Thus, a threshold value may be used to control the false negative-false positive trade-off, i.e., an operating point of the detector.

The different detectors may have very different performance, and the scores given by them are not fully correlated. Therefore, the combination of their outputs provides an opportunity to obtain a combined detector with performance superior to any of the components.

The cost of classification in terms of time and computational power, besides accuracy, is an important factor in many detection problems. Some of the detectors are very fast to execute while others are computationally heavy. An effective way to reduce the cost of classification is to use a sequential decision making process which asks for new resources only if needed for required accuracy.

We propose a new method for combining multiple sensitivity tunable detectors, i.e., detectors which output likelihood scores, to form a computationally efficient binary classification cascade. The component detectors are not restricted to be based on a single feature set, but may even operate on different measurement modalities. They have preferably been trained with different datasets to introduce uncorrelatedness in their output scores. For

*Correspondence: katariina.mahkonen@tut.fi
Tampere University of Technology, Korkeakoulunkatu 1, 33720 Tampere, Finland

combining the available sensitivity tunable detectors, we propose to utilize a monotone Boolean function built using **AND** ($\wedge$) and **OR** ($\vee$) operators in disjunctive normal form (DNF). A Boolean function (BF) is said to be *monotone*, if changing the value of any of the input variables from 0 to 1 cannot decrease the value of the function from 1 to 0. For continuous data binarization, we use similar procedure as presented in [6]. Thus, a monotone BF on this data performs a monotone partition of the space of measurement values.

A BF lends itself naturally to sequential evaluation, which is an integral property of a decision process of a classification cascade. Also, by utilizing a BF of thresholded detector scores, we avoid inferring class probabilities from the scores, which would be error prone while having only a small dataset for combined system training. In the proposed **OR** of **AND**s function (BOA), each detector score is compared to multiple threshold levels, which allows formulating any monotonic decision boundary while making the classification decision in a computationally efficient way. The BOA cascade detector itself is trained to be sensitivity controllable as well.

The contributions of the paper are (1) a monotone Boolean **OR** of **AND**s (BOA) binary classification function to build a cascaded combination of multiple sensitivity tunable detectors, (2) an algorithm to train a BOA combination, and (3) utilizing a cascaded decision making process for audio-visual detection task.

For evaluating the proposed BOA detector cascade and the training algorithm to set its parameters, we use two audiovisual databases for two detection tasks, namely MAHNOB laughter dataset [7] for laughter detection task and CASA dataset [8] for video context change detection task. In the laughter detection task, we show that the accuracy of detection with a BOA cascade is superior to the other detection accuracies reported in the literature, while the computation time of detection is remarkably reduced compared to the other solutions. With three component detectors for the video context change detection, we show that the proposed BOA training algorithm outperforms alternative Boolean combination training algorithms found in the literature.

In the following section, we introduce the work related to Boolean detector combinations and algorithms for training Boolean combination parameters, as well as the work on cascaded detectors presented in the literature. The proposed Boolean OR of ANDs combination, and the algorithm to set its parameters are presented in Section 3. The experimental setup and the results obtained are presented in Section 4.

2 Related work

This paper proposes combining multiple tunable detectors robustly utilizing a monotone DNF-BF, named BOA, the evaluation of which is formulated as a computationally efficient classification cascade. Thus, we first review the literature on Boolean detector combinations and BFs in general. Then, we review the algorithms suitable for training a Boolean combination. Finally, we discuss the literature on classification cascades.

2.1 Boolean detector combinations

Using a Boolean conjunction or a Boolean disjunction for combining multiple detectors has been proposed in several studies, for example in [9–11]. Sensitivity tunable detector functions $f_m : \boldsymbol{x} \to \mathbb{R}$ for $m = 1 \ldots M$ are utilized within a combination. Each detector function $f_m(\boldsymbol{x})$ produces a score l_m, which denotes likelihood of the target appearing in the sample $\boldsymbol{x}$. The Boolean conjunction of M sensitivity tunable detectors is

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigwedge_{m=1}^{M} \left(f_m(\boldsymbol{x}) \geq \theta_m^{\text{AND}} \right), \qquad (1)$$

and the Boolean disjunction is

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{m=1}^{M} \left(f_m(\boldsymbol{x}) \geq \theta_m^{\text{OR}} \right), \qquad (2)$$

where $\boldsymbol{\theta}$ denotes all the thresholds θ_m^* used within the combination. All of the studies [9–11] report that either a conjunctive or a disjunctive Boolean combination of detectors do improve the detection accuracy over component detectors, provided that the thresholds $\theta_1{}^*, \theta_2{}^*, \ldots, \theta_M^*$ are set appropriately.

Mixtures of **AND** and **OR** operators within a Boolean combination have been investigated in [12]. Utilizing notation, where the detector function $f_m(\boldsymbol{x})$ identifiers m are listed in vectors z_q, $q = 1 \ldots Q$, each z_q containing M_q identifiers, this kind of Boolean **OR** of **AND**s combination is

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{q=1}^{Q} \left[\bigwedge_{i=1}^{M_q} \left(f_{z_q(i)}(\boldsymbol{x}) \geq \theta_{z_q(i)} \right) \right]. \qquad (3)$$

As a big limitation of (3) proposed in [12], compared to the BOA combination that we suggest, is that only one threshold θ_m for each target likelihood score $f_m(\boldsymbol{x}) = l_m$ is allowed.

In addition to **AND** and **OR** operators, the Boolean negation, (**NOT**), and as a consequence also the exlusive-OR (**XOR**) are utilized in the detector combinations in [13–15]. The 2^{2^M} possible Boolean combinations that can be formed by M fixed, i.e., non-tunable, detectors utilizing **AND**, **OR**, **XOR**, and **NOT** operators are studied in [13].

Boolean detector combinations where each of the available target likelihood scores $f_m(\boldsymbol{x}) = l_m, \; m = 1 \ldots M$ may be cast to Boolean values using multiple thresholds $\theta_m^1, \; \theta_m^2, \ldots$ are first made use of in [14]. However, the space of the Boolean combinations generated by their algorithm is left unspecified.

A question of how to select the best performing Boolean combination for a certain problem, while having M sensitivity tunable detectors, has been posed in many of the abovementioned works. To select between conjunctive (1) and disjunctive (2) combinations, in [10, 16], it is suggested to investigate the class-conditional cross-correlations of detector scores and to consider whether the specificity or the sensitivity is more important. The conjunctive fusion rule (1), which emphasizes specificity, should be used if there is negative correlation between detector outputs for samples of the "non-target" class. If on the other hand the correlation of detector output scores for samples from the "target" class is weak, disjunctive fusion rule (2) emphasizing sensitivity should be used. All in all, a Boolean combination is able to exploit negative or weak correlation of detector scores.

To select among the combinations of the form (3), rules of thumb have been drawn in [12] according to average cross-correlations between the scores from the used detectors. It is shown for three detectors with Gaussian score distributions and identical pairwise cross-correlations that either a conjunctive combination (1), a disjunctive combination (2), or a type (3) combination

$$\begin{aligned} B(\boldsymbol{x};\boldsymbol{\theta}) = & \left[\left(f_1(\boldsymbol{x}) \geq \theta_1^{\text{vote}}\right) \wedge \left(f_2(\boldsymbol{x}) \geq \theta_2^{\text{vote}}\right) \right] \quad \vee \\ & \left[\left(f_1(\boldsymbol{x}) \geq \theta_1^{\text{vote}}\right) \wedge \left(f_3(\boldsymbol{x}) \geq \theta_3^{\text{vote}}\right) \right] \quad \vee \\ & \left[\left(f_2(\boldsymbol{x}) \geq \theta_2^{\text{vote}}\right) \wedge \left(f_3(\boldsymbol{x}) \geq \theta_3^{\text{vote}}\right) \right], \end{aligned}$$

which stands for a majority vote rule, is the best and outperforms the component detectors. The one of those to be selected depends on class conditional cross-correlations between detectors.

The Iterative Boolean Combination (IBC) method in [14] is specifically designed to find the best possible Boolean combination, not restricted to monotone functions, for a certain sensitivity level of a combination. The search space of BFs is nevertheless restricted to avoid an unfeasibly large number of possibilities. The IBC method results in variety of Boolean detector compounds, but the study does not provide analysis of the form of the generated compounds nor characteristics of their resulting decision boundaries.

Theory of constructing BFs of unrestricted form, specifically in DNF as well as in CNF (conjunctive normal form), has been studied in depth, e.g., in [17]. BFs for classification have been studied vastly under terms logical analysis and inductive inference. Logical Analysis of Data (LAD) [18, 19] is a combinatorics- and optimization-based data analysis method first introduced in [20]. LAD methodology focuses on finding DNF-BF-type representations for classes.

The term inductive inference is used in many early texts concerning topics of machine learning, many of those discussing Boolean decision-making, e.g., [21, 22]. Using data binarization, e.g., as proposed in [6], all these results concerning BFs may be utilized in conjunction with continuous valued data.

Any BF may be converted into a binary decision tree, while the structure of the tree is generally not unique. In case of the proposed BOA DNF-BF, the corresponding deterministic read-once binary tree has depth $\geq \lceil \log_2(N_\theta + 1) \rceil$. In maximally deep node arrangement, the tree becomes a single branch tree with depth equal to the number N_θ of thresholds used in the BOA function. However, this kind of binary tree representation does not highlight the computational advantages of BOA cascade that we are interested in.

2.2 Algorithms for training a Boolean combination

The parameter $\boldsymbol{\theta}$ of a Boolean combination function $B(\boldsymbol{x};\boldsymbol{\theta})$ denotes all the thresholds $\theta_m^n \in \theta$ for $m = 1 \ldots M$, $n = 1 \ldots N_m$ used in the combination. For a Boolean combination $B(\boldsymbol{x};\boldsymbol{\theta})$ to perform well, suitable values for the set $\boldsymbol{\theta}$ of thresholds must be found. Most of the studies rely on training data-based exhaustive search for selecting the threshold values for $\boldsymbol{\theta}$, e.g., [10, 12, 13]. The computational load of this approach is $\mathcal{O}\left(T^{|\boldsymbol{\theta}|}\right)$, where $|\boldsymbol{\theta}| = \sum_{m=1}^{M} N_m$ is the total number of thresholds in $\boldsymbol{\theta}$ and T is the number of threshold values tested for each detector. The exhaustive search becomes computationally prohibitive if there are more than a couple of threshold values to find. Thus, more efficient algorithms are needed. In addition to algorithms readily proposed for tunable classification function training, we shortly review algorithms which have been developed for BF training for one operating point and their extensions to incremental learning.

A fast method for finding sets $\boldsymbol{\theta}$ of thresholds for different sensitivity levels of a Boolean combination $B(\boldsymbol{x};\boldsymbol{\theta})$ is presented in [10]. The method exploits the receiver operating characteristic (ROC) curve of each utilized detector $d_m(\boldsymbol{x}, \theta) = \left(f_m(\boldsymbol{x}) \geq \theta\right)$. The ROC curve shows the true positive rate (*tpr*) against the false positive rate (*fpr*) at every operating point, defined by the threshold θ, of the detector. When $\theta = -\infty$, the classification by $d(\boldsymbol{x}, \theta)$ results in *tpr* = 100% and *fpr* = 100%. On the other hand, when $\theta = +\infty$, then *tpr* = *fpr* = 0. The method selects the thresholds for the Boolean combination iteratively by fusing two BF components—individual detectors or partial BFs—at a time according to their ROC curves. Formulas for ROC curves of a conjunctive and disjunctive combination of detectors d_A and d_B, $d_A \neq d_B$, are provided as

$$tpr_{\wedge}\left(fpr_{\wedge}\right) = \max_{fpr_A \cdot fpr_B = fpr_{\wedge}} \left(tpr_A(fpr_A) \cdot tpr_B(fpr_B)\right) \tag{4}$$

and

$$tpr_{\vee}\left(fpr_{\vee}\right) = \max_{fpr_A + fpr_B - fpr_A \cdot fpr_B = fpr_{\vee}} \left(tpr_A(fpr_A) + tpr_B(fpr_B) - tpr_A(fpr_A) \cdot tpr_B(fpr_B)\right), \tag{5}$$

where $tpr_*(fpr_*)$ denotes the true positive rate of a detector $d_*(\boldsymbol{x};\theta)$ at an operating point θ where its false positive rate is fpr_*.

The efficiency of the method is based on an assumption that the classifications made by different detectors are independent. Unfortunately, this often does not hold in practice. If the same measurement set or the same set of features are used for multiple detectors, or if multiple thresholds are to be found for a certain target likelihood l_m within a Boolean combination, dependencies between classifications are very likely. We compare our algorithm to this Boolean algebra of ROC curves in the Section 4 and use an implementation for BOA training shown in Appendix 1.

Another algorithm that does not assume independence of the used detectors was proposed in [11]. It suggests training the combination iteratively by finding thresholds for two detectors or partial combinations at a time, similarly to the Boolean algebra of ROC curves presented above. In this approach, the search of the best thresholds for a Boolean combination is done via exhaustive search over all the possible threshold settings for the two systems to be merged. In the ROC space, with all the possible threshold settings, a Boolean combination produces a constellation of performance points. The left top edge of this constellation, consisting of the operating points of superior performance, was introduced by [23] as the convex hull of the ROC constellation. In the algorithm of [11], before each new component fusion, the set of possible threshold values for the newly built partial combination is pruned to constitute of only the thresholds corresponding the performance points at the convex hull of this ROC constellation. The algorithm is originally designed for pure conjunctive (1) or disjunctive (2) Boolean combinations, but we have implemented it to deal with a BOA as described in the Appendix 1, and we use it for comparison to our algorithm.

In the literature concerning BFs, there are many algorithms, which are designed to find a BF which perfectly classifies the training data $\left\{X^0, X^1\right\}$ in $\{0,1\}^{N_{\mathrm{attr}}}$. Finding the simplest possible BF to explain some data is an NP complete optimization problem with $2^{2^{N_{\mathrm{attr}}}}$ possible solutions. Some of the algorithms are designed assuming monotonicity of data, the assumption which diminishes the number of possible solutions remarkably [24]. The number of possible BFs is further reduced in the case of continuous data which is binarized as in [6]. In this case, the data with $M \ll N_{\mathrm{attr}}$ continuous attributes actually resides in the M-dimensional manifold of the N_{attr} dimensional space of binarized data. However, the number of possible BFs is still exponential. A few of the approaches target finding a BFn with imperfect classification performance, which usually is the desirable learning result with imperfect data.

Because of NP completeness of finding the best BF to explain some data, most of the algorithms in the literature operate in iterative manner using some greedy heuristics. An A^q algorithm [25] and LAD [20]-based methods construct a DNF-BF via iteratively searching for good conjunctions, each of which covers a part of positive training samples, to be combined disjunctively. On the contrary, OCAT-RA1 -algorithm [26], based on idea of one-clause-at-a-time (OCAT) [27], builds a CNF-BF via iterative selection of disjunctions. In case of continuous data binarized as in [6], algorithms developed for decision tree learning, e.g., ID3 [28], C4.5 [29], CART [30] are also suitable for DNF-BF building.

The A^q algorithm and LAD-based methods are to find two DNF-BFs which provide perfect classification of the training data. One function is to be used for detection of the positive class, and the other one for detecting the negative class. The covers, i.e., subspaces for which BF $=$ *true*, of these DNF-BFs are disjoint, leaving part of the input space uncovered by either function. The algorithms use different heuristic criteria when searching for suitable conjunctions, i.e. *complexes* in terms of A^q.

For A^q algorithm the user may choose the criterion, one possible choice beings the number of positive samples covered by the *complex*, that is, conjunction. For LAD methodology, different criteria for optimality of conjunctions, called *patterns* in LAD, are discussed in [31]. Selectivity criterion favors minterms based on data, and evidential criterion favors patterns covering as many data samples as possible. Algorithms for constructing patterns according to these different criteria are given in [18, 32, 33].

Algorithms for BF inference allowing imperfect classification, which is generally associated with better generalization of data with outliers, are for example AQ15 algorithm [34], which is based on A^q, and OCAT-RA1 algorithm proposed in [26]. A procedure for pruning an overfit DNF-BF representation for better generalization is provided within AQ15 algorithm. It is based on counts of samples covered by each conjunction individually and together with other conjuntions. The conjunctions which are small in these numbers are the ones to be pruned. OCAT-RA1 constructs each disjunction of a CNF-BF by iteratively selecting attributes for it based on their rank of $N_{\mathbf{tp}}(a)/N_{\mathbf{fp}}(a)$, where $N_{\mathbf{tp}}(a)$ ($N_{\mathbf{fp}}(a)$) is the number of positive (negative) training samples, which have attribute $a = 1$. New attributes are selected until all the positive samples are covered by their disjunction.

The binary tree building algorithms, which iteratively build the tree by starting from the root node and performing a new split at every iteration, implicitly facilitate different level generalizations of data and generate a decision function of DNF-BF form. The splitting criterion for selecting attributes for new nodes in ID3, C4.5, and C5.0 is gain in information entropy. ID3 is applicable with binarized data, while C4.5 and C5.0 can handle continuous data by implicitly performing the binarization by usage of thresholds. The CART algorithm uses either Gini impurity or Twoing criterion to decide about the attributes used in nodes of the tree.

Incremental learning algorithms enable updating a classification function when new data becomes available. Some of the algorithms keep all the data available for future updates, while some algorithms discard the original data and perform the update based on new data only. Incremental algorithms, which utilize all the original training data aside of some new data, for updating a BF are for example GEM [35] and IOCAT [36].

Both of the algorithms assume a DNF-BF, and their update procedures consist of two phases. At the first phase, if some of the new negative samples are misclassified by the original DNF-BF, the faulty conjunctions are located and specialized to not to cover those new samples. Both of the algorithms perform this step by replacing each faulty conjunction by new conjunctions which are trained using data inside the cover of the original conjunction. GEM utilizes A^q algorithm and IOCAT utilizes OCAT-RA1 algorithm for this re-training. At the second phase of BF update, the DNF-BF is updated in terms of the uncovered new positive samples. GEM generalizes the existing conjunctions to cover the new positives using A^q.

In IOCAT, for each uncovered new positive sample, a conjunction, i.e., *clause* in terms of IOCAT, to be generalized is selected based on ratio $N_{\mathbf{tp}}(\text{clause})/N_{\text{attr}}(\text{clause})$ of the number of positive samples covered by the clause $N_{\mathbf{tp}}(\text{clause})$ and the number of attributes in the clause $N_{\text{attr}}(\text{clause})$. The selected conjunction is then retrained with non-incremental OCAT-RA1 algorithm using all the negative samples, the new positive sample and the positive samples within the space covered by the selected conjunction.

2.3 Cascade processing for reduced computational load of classification

The goal in cascaded processing for detection is in reducing the computational cost of classification. The idea is to evaluate the input in stages, such that at each cascade stage new information about the input is acquired and then either the classification is released or the next cascade stage is entered for new information. Decision cascades have been investigated mostly in the field of machine vision starting from [37, 38]. Face detection and pedestrian detection are the most common application areas where decision cascades have been used, e.g., in [39–42]. Decision cascades have been utilized in other fields, e.g., in [43] for cancer survival prediction and in [44] for web search.

In the task of object detection from images, the heavily imbalanced class distribution, as most of the search windows of different sizes and positions do not contain the target object, offers great possibilities to make "non-target" classification with minor examination. Object detection cascades are designed such that gradually more and more features are extracted for increased classification certainty. A class estimate is released as soon as the classification certainty is high enough. If this is the case before all the obtainable features or measurements have been extracted, computational savings appear.

The first generation object detection cascades, used for example in [38], are able to make early classification to the "non-target" class only, as illustrated in Fig. 1 (left). To classify the input into the "target" class, the input must pass all tests $\left(f_s(\boldsymbol{x}) \geq \theta_s\right)$ of the cascade stages $s = 1 \ldots S$. This kind of one-sided cascade performs a conjunctive Boolean combination function

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigwedge_{s=1}^{S} \left(f_s\left(\boldsymbol{x}\right) \geq \theta_s\right).$$

The solution $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{true}$ denotes classification to the "target" class, and $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{false}$ denotes classification to the "non-target" class.

The second generation object detection cascades introduced in [45] and used also in [46] are able to make the early classification to both the classes, as illustrated in Fig. 1 (right). They utilize two thresholds on the target likelihood score $f_s(\boldsymbol{x}) = l_s$ at each cascade stage $s = 1 \ldots S - 1$. One threshold, θ_s^{reject}, is used for early rejection, i.e., early classification to "non-target" class, if $\left(f_s(\boldsymbol{x}) < \theta_s^{\text{reject}}\right) = \textit{true}$. Another threshold, θ_s^{accept}, is used for early detection if $\left(f_s(\boldsymbol{x}) \geq \theta_s^{\text{accept}}\right) = \textit{true}$. This means that at each stage, either the classification is released, or the next stage is entered in case that $\theta_s^{\text{reject}} \leq l_s < \theta_s^{\text{accept}}$. At the last cascade stage, the classification is enforced by $\theta_S = \theta_S^{\text{reject}} = \theta_S^{\text{accept}}$. This kind of symmetrical cascade corresponds to a BF

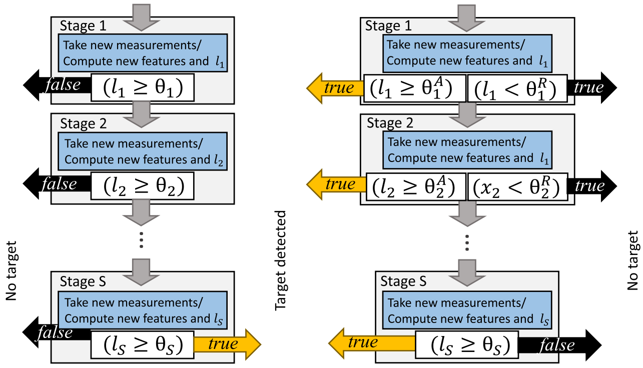

Stage 1
Take new measurements/ Compute new features and l_1
false
$(l_1 \geq \theta_1)$
Stage 2
Take new measurements/ Compute new features and l_2
false
$(l_2 \geq \theta_2)$
Stage S
Take new measurements/ Compute new features and l_S
false
$(l_S \geq \theta_S)$
true
No target
Target detected
Stage 1
Take new measurements/ Compute new features and l_1
true
$(l_1 \geq \theta_1^A)$
$(l_1 < \theta_1^R)$
true
Stage 2
Take new measurements/ Compute new features and l_1
true
$(l_2 \geq \theta_2^A)$
$(x_2 < \theta_2^R)$
true
Stage S
Take new measurements/ Compute new features and l_S
true
$(l_S \geq \theta_S)$
false
No target

Fig. 1 Two types of binary classification cascades. Typical object detection cascades utilized in computer vision. Left: an asymmetrical type cascade for classifying efficiently the "non-target" windows. Right: a symmetrical detection cascade which is capable of early classification to both classes

$$B(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{s=1}^{S} \bigwedge_{m=1}^{s-1} \left(f_m(\boldsymbol{x}) \geq \theta_m^{\text{reject}} \right) \wedge \left(f_s(\boldsymbol{x}) \geq \theta_s^{\text{accept}} \right), \tag{6}$$

whose output $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{true}$ denotes the classification to the "target" class and $B(\boldsymbol{x};\boldsymbol{\theta}) = \textit{false}$ denotes classification to the "non-target" class.

A cascade may be seen as a one branch decision tree, if the notion of tree is broadened from the traditional definition that a node makes a decision based on only one input attribute. In a "cascade-tree," a node function may utilize multiple input attributes, and the function may partition the corresponding input space freely to assign inputs to any of the leaves, i.e., classes, or down the branch to the next level node (stage of the cascade). In a cascade, the order of attribute acquisition is fixed in contrast to input-dependent order of attribute usage with a traditional decision tree.

For training, a detection cascade for computer vision applications, where the detectors to be utilized are designed having close to infinite pool of image features, e.g., Haar, HoG, an efficient cascade structure is guaranteed by concurrent design of detector functions f_s, $s = 1 \ldots S$, their thresholds θ_s^* and the cascade length S as proposed in [40, 47]. For a cascade with fixed length S, a method for concurrent learning of object detectors and their operating points is proposed in [39]. The methods proposed in the literature for finding operating points for pre-trained detectors within a detection cascade mostly assume strong correlation among detector scores. This is the case in [48], where an object detection cascade is designed using cumulative classifier scores, as well as in [45, 46], where the proposed algorithms are based on the assumption that the detector scores are highly positively correlated. If the detector scores are negatively or not correlated, those cascade training strategies turn unsuitable.

3 Methods

For combining multiple detector functions $f_m(\boldsymbol{x}) = l_m$, $m = 1 \ldots M$, which output likelihood scores $l_1, l_2, \ldots, l_M$ for the same target class, we propose to use a a BF. The proposed combination function utilizes Boolean AND ($\wedge$) and OR ($\vee$) operators and it is defined in disjunctive normal form. The proposed Boolean OR of ANDs function (BOA) B yields a Boolean output $\mathrm{B} : \boldsymbol{x} \rightarrow \{\textit{false, true}\}$. The BOA output $\mathrm{B}(\boldsymbol{x}) = \textit{true}$ denotes input $\boldsymbol{x}$ classification to the "target" class and the BOA output $\mathrm{B}(\boldsymbol{x}) = \textit{false}$, i.e., $\neg\mathrm{B}(\boldsymbol{x}) = \textit{true}$, denotes classification to the "non-target" class.

Generally, a BF—possibly infinite—over a combination of thresholded detector scores is capable of producing any binary partition of the input space $\boldsymbol{x}$ or the space of target likelihood scores $(l_1, l_2, \ldots, l_M)$. Due to exclusion of the Boolean NOT rule, a BOA combination restricts the space of different partitions such that the spaces $\left\{ (l_1, l_2, \ldots, l_M) \mid \mathrm{B}(\boldsymbol{x}) = \textit{false} \right\}$ and $\{ (l_1, l_2, \ldots, l_M) \mid \mathrm{B}(\boldsymbol{x}) = \textit{true} \}$ are simply connected and the decision boundary is monotonic. This is illustrated in the example of Fig. 2, where the data points indicate laughter likelihoods from videos of MAHNOB laughter dataset [7], which is used in our evaluations.

We build a BOA combination of detector functions $f_m(\boldsymbol{x}) = l_m$, $m = 1 \ldots M$ using Boolean OR ($\vee$) and AND ($\wedge$) operators as

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{q=1}^{Q} \bigvee_{n=1}^{N_q} \left[\bigwedge_{i=1}^{M_q} \left(f_{z_q(i)}(\boldsymbol{x}) \geq \theta_{z_q(i)}^{q,n} \right) \right], \tag{7}$$

where in each vector $z_q \in \{1 \ldots M\}^{M_q}$ there are M_q detector identifiers $m \in \{1 \ldots M\}$ for BOA construction. Each

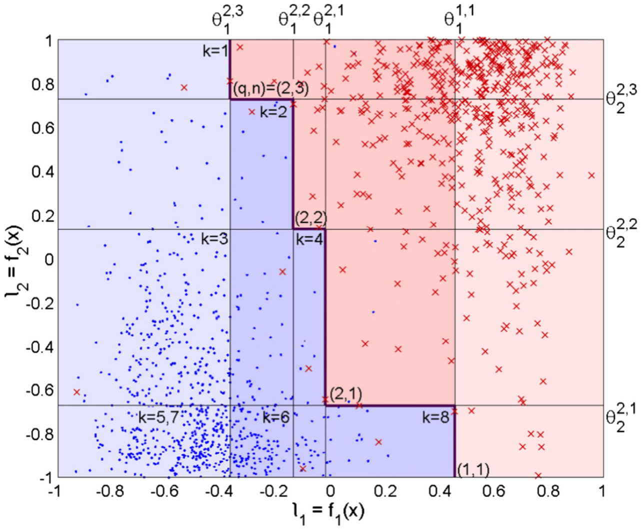


Fig. 2 Example of BOA decision boundary. Illustration of classification of MAHNOB Laughter dataset videos with BOA. Data $\boldsymbol{x} \in X = \{X^0, X^1\}$ from two classes, "laughter" and "speech," is represented in terms of two target likelihood scores l_1 and l_2. The data samples from the "laughter" class X^1 are shown with red crosses and the data samples of the "speech" class X^0 are shown with blue dots. The resulting decision boundary by the BOA combination (10) is shown with the bold angular line. Each threshold $\theta_m^{q,n}$, $m=1,2,\ q=1,2,\ n=1,2,3$ is illustrated with a thin line. The space of target likelihood scores where $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = true$ is colored with pink background, and the space where $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = true$ is colored with blue background. The palest background colors illustrate the subspaces, where the decision is done using the score l_1 only

term $\left[\bigwedge_{i=1}^{M_q}\left(l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n}\right)\right]$ in (7) is a *conjunction* over the Boolean threshold comparisons of the target likelihood scores $\{l_m \,|\, \exists i \;\; m = z_q(i)\}$. The multiplicity of a conjunction type z_q is denoted by N_q.

Every conjunction, enumerated by (q,n), operates with a distinct set of thresholds $\theta_{z_q(i)}^{q,n}$, $i=1\ldots M_q$.

The negation of the BOA function (7) is used for the cascade implementation of its evaluation. In the BOA cascade, the classification to the "non-target" class is formulated via the negation of the BOA function—whenever the negated BOA function equals *true*. The Boolean negation of $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ in (7), in disjunctive normal form, is

$$\begin{aligned}\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) &= \bigvee_{k=1}^{K}\left[\bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q}\left(f_{z_q(\mathcal{I}(k,q,n))}(\boldsymbol{x}) < \theta^{q,n}_{z_q(\mathcal{I}(k,q,n))}\right)\right]\\ &= \underbrace{\bigvee_{i_{1,1}=1}^{M_1}\bigvee_{i_{1,2}=1}^{M_1}\bigvee_{i_{1,3}=1}^{M_1}\cdots\bigvee_{i_{1,N_1}=1}^{M_1}}_{N_1 \bigvee\text{-operators , i.e., } M_1^{N_1}\text{ conjunctions}}\ \underbrace{\bigvee_{i_{2,1}=1}^{M_2}\bigvee_{i_{2,2}=1}^{M_2}\cdots\bigvee_{i_{2,N_2}=1}^{M_2}}_{N_2 \bigvee\text{-operators, i.e., } M_2^{N_2}\text{ conjunctions}}\ \cdots\\ &\quad\cdots\underbrace{\bigvee_{i_{Q,1}=1}^{M_Q}\bigvee_{i_{Q,2}=1}^{M_Q}\cdots\bigvee_{i_{Q,N_Q}=1}^{M_Q}}_{N_Q \bigvee\text{-operators, i.e., } M_Q^{N_Q}\text{ conjunctions}}\left[\bigwedge_{q=1}^{Q}\bigwedge_{n=1}^{N_q}\left(f_{z_q(i_{q,n})}(\boldsymbol{x}) < \theta^{q,n}_{z_q(i_{q,n})}\right)\right].\end{aligned} \tag{8}$$

where the number of conjunctions is given by $K = \prod_{q=1}^{Q} M_q^{N_q}$, and the index $\mathcal{I}(k,q,n)$ of the detector function identifier m within vector z_q of the first representation is given by

$$\mathcal{I}(k,q,n) = \left\lfloor \frac{\left\lfloor \frac{k-1}{\prod_{i=q+1}^{Q} M_i^{N_i}} \right\rfloor}{M_q^{N_q-n}} \right\rfloor \bmod M_q \quad +1, \quad (9)$$

Figure 2 illustrates the decision boundary using a BOA combination with $z_1 = [1]$, $z_2 = [1,2]$, and $N_1 = 1, N_2 = 3$, which is

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \left(l_1 \geq \theta_1^{1,1}\right) \vee \bigvee_{n=1}^{3} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right) \right] \quad (10)$$

and its negation is

$$\begin{aligned}
\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) &= \bigvee_{k=1}^{8} \left[\bigwedge_{q=1}^{2} \bigwedge_{n=1}^{N_q} \left(f_{z_q(\mathcal{I}(k,q,n))}(\boldsymbol{x}) < \theta^{q,n}_{z_q(\mathcal{I}(k,q,n))} \right) \right] \\
&= \bigvee_{i_1=1}^{2} \bigvee_{i_2=1}^{2} \bigvee_{i_3=1}^{2} \left[\left(f_1(\boldsymbol{x}) < \theta_1^{1,1} \right) \wedge \bigwedge_{n=1}^{N_q} \left(f_{z_q(i_n)}(\boldsymbol{x}) < \theta^{q,n}_{z_q(i_n)} \right) \right] \\
&= \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_1 < \theta_1^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_1 < \theta_1^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_1 < \theta_1^{2,3}\right) \right] \\
&\quad \vee \left[\left(l_1 < \theta_1^{1,1}\right) \wedge \left(l_2 < \theta_2^{2,1}\right) \wedge \left(l_2 < \theta_2^{2,2}\right) \wedge \left(l_2 < \theta_2^{2,3}\right) \right].
\end{aligned} \quad (11)$$

The corners of the resulting decision boundary are formed by the conjunctions $(q,n) = (1,1), (2,1), (2,2)$, and $(2,3)$ of (10), which are designated in Fig. 2 by the conjunction indexes (q,n) next to each corresponding outer corner of space $\{ \ (l_1,l_2) \mid \mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true} \ \}$. The outer corners of space $\{ \ (l_1,l_2) \mid \neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true} \ \}$, which are generated by the conjunctions $k = 1 \ldots 8$ of (11), are similarly designated in Fig. 2.

There may be redundancy in the BOA equation or its negation, depending on values of the thresholds selected for $\boldsymbol{\theta}$. A conjunction within a BOA is redundant, if the BOA decision boundary does not change by removing that conjunction from the BOA equation.

Considering a BOA with conjunction lists $z_1, z_2, \ldots, z_Q$ and conjunction multiplicities $N_1, N_2, \ldots, N_Q$, to find out whether a conjunction (q, n_q) is redundant or not, its thresholds $\left\{ \theta^{q,n_q}_{z_q(i)} \mid i = 1 \ldots M_q \right\}$ must be examined. Each threshold $\theta^{q,n_q}_{z_q(i)}$ must be compared to thresholds $\theta^{p,n_p}_{z_p(j)}$, $z_q(i) = z_p(j) = m$, on the same target likelihood score l_m, which are used within other conjunctions (p, n_p) of the BOA. The conjunctions (p, n_p) to be considered are those with z_p containing $m = z_q(i)$ and possibly other identifiers from z_q. The list z_p may not contain identifiers not listed in z_q. Formally, $\{ z_p \mid p \neq q \text{ and } \exists i,j \ni m = z_p(j) = z_q(i) \text{ and } \forall i \in \{1 \ldots M_p\} \ \exists j \ni \ z_p(i) = z_q(j) \ \}$. The range of n_p for (p, n_p) is naturally $n_p = 1 \ldots N_p$. For a conjunction (q, n_q) to be non-redundant, one of its thresholds $\theta^{q,n_q}_{z_q(i)}$, $i = 1 \ldots M_q$ must be smaller than any threshold $\theta^{p,n_p}_{z_p(j)}$, $z_q(i) = z_p(j) = m$, in its corresponding conjunctions (p, n_p). That is, in conjunction (q, n_q) there must exist at least one threshold θ_m^{q,n_q} for which $\theta_m^{q,n_q} < \theta_m^{p,n_p}$ of all the corresponding conjunctions (p, n_p).

3.1 BOA as a binary classification cascade

Algorithmically, a BF is evaluated in steps, i.e., sequentially. If any of the conjunctions of BOA function (7) or its negation (8) resolves as *true*, the entire functions (7) and (8) become determinate. In other words, as soon as any of the conjunctions (q,n), $q = 1 \ldots Q$, $n = 1 \ldots N_q$ of a BOA $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ outputs *true*, i.e., $\left[\bigwedge_{i=1}^{M_q} \left(l_{z_q(i)} \geq \theta^{q,n}_{z_q(i)} \right) \right] = \mathit{true}$, it means that $\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true}$. Without evaluating the rest of the BOA conjunctions the detection result "target event detected" may then be announced. Similarly, if any of the conjunctions $k = 1 \ldots K$ of the negation of the BOA $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta})$ outputs "true," that is, if $\left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \left(l_{z_q(\mathcal{I}(k,n,q))} < \theta^{q,n}_{z_q(\mathcal{I}(k,n,q))} \right) \right] = \mathit{true}$, it means that $\neg\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \mathit{true}$. The evaluation can then be stopped and the classification result "non-target" can be released.

Computationally, the heaviest part of BOA evaluation is the acquisition of target likelihood scores l_m for an input sample $\boldsymbol{x}$ by computing the functions $f_m(\boldsymbol{x}) = l_m$, $m = 1 \ldots M$. The cost of threshold comparisons within BOA may be considered negligible. From computational aspect of evaluating a BOA, once the likelihood score l_m is acquired, all the Boolean comparisons $(l_m \geq \theta_m^*)$ and $(l_m < \theta_m^*)$, which are based on the score l_m, become immediately available. In case the BOA function (7) or its negation (8) becomes determinate with the Boolean comparisons of already computed subset of scores l_m, $m = 1 \ldots M$, the classification may be released without running the rest of the detector functions at all.

We have implemented the BOA as a binary classification cascade, where a cascade stage $s \in \{1 \ldots S\}$ calculates a score $f_m(\boldsymbol{x}) = l_m = l_s$ using a predefined detector function f_m and offers a possibility for releasing the classification result, as shown in Fig. 3. Internal decisions at each stage $s = 1, 2, \ldots, S$ of the BOA cascade, whether to release a class estimate or to enter the next cascade stage, are made with BFs $B_s^{\text{class}}((l_1, l_2, \ldots, l_s))$, $s = 1 \ldots S$, i.e. $B_1^1(l_1)$, $B_1^0(l_1)$, $B_2^1(l_1, l_2)$, $B_2^0(l_1, l_2)$,..., $B_S^1(l_1, l_2, \ldots, l_S)$ and $B_S^1(l_1, l_2, \ldots, l_S)$. That is, the functions B_s^1 and B_s^0 of cascade stage s utilize the target likelihood scores $l_1, l_2, \ldots, l_s$. All these functions are partitions of the BOA function $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ of (7) and its negation $\neg\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ of (8) such that

$$\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = B_1^1 \vee B_2^1 \vee \ldots \vee B_S^1 \tag{12}$$

and

$$\neg\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = B_1^0 \vee B_2^0 \vee \ldots \vee B_S^0. \tag{13}$$

Formal expressions for the partition of the BOA function (7) into functions $B_1^1, B_2^1, \ldots., B_S^1$ and the BOA negation (8) into functions $B_1^0, B_2^0, \ldots., B_S^0$ are derived in Appendix 2.

As an example, operation process of BOA cascade $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = \left(l_1 \geq \theta_1^{1,1}\right) \vee \bigvee_{n=1}^{3} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right) \right]$ for MAHNOB laughter data classification is illustrated in the Fig. 2 with the background color of the l_1-vs. l_2-axis and is as follows. The classification takes place at the first cascade stage for all the samples $\boldsymbol{x}$ for whom $B_1^1(\boldsymbol{x}) = \left(l_1 \geq \theta_1^{1,1}\right) = true$ or $B_1^0(\boldsymbol{x}) = \left(l_1 < \min\left(\theta_1^{2,1}, \theta_1^{2,2}, \theta_1^{2,3}\right)\right) = \left(l_1 < \theta_1^{2,3}\right) = true$. In the first case, the classification is "Laughter detected," and in the second case "No Laughter." These subspaces of (l_1, l_2) on the left and right outskirts of Fig. 2 are indicated with a pale background color. In the second stage of the cascade processing, the likelihood $f_2(\boldsymbol{x}) = l_2$ is computed only for the samples with $\theta_1^{2,3} \leq l_1 < \theta_1^{1,1}$, although l_2 is shown for all the samples in the Fig. 2. With the dataset in the Fig. 2, it means that classification of approximately 65% of the samples are made using the detector function f_1 only.

The computational efficiency of the cascade naturally depends on the order of detector methods to be utilized at cascade stages. Generally, the faster methods should be evaluated first, and the slower ones later. If the methods f_m, $m = 1 \ldots M$ have very different computational loads $\mathcal{L}_m$, $m = 1 \ldots M$, it is very likely that a cascade ordered such that $\mathcal{L}_s \ll \mathcal{L}_{s+1}$, $s = 1 \ldots S - 1$ is the most efficient one. Precisely, the most computationally efficient cascade structure may be defined via local inequalities among each two consecutive stages s and $s + 1$ as follows. If we denote the probability of a sample arriving stage s to be classified at stage s after computing $l_s = f_s(x)$ with P_1, and the probability of a sample arriving stage s to be classified at stage s if the detector method f_{s+1} would be utilized instead of the method f_s with P_2, it must hold that $P_1 \geq (\mathcal{L}_s/\mathcal{L}_{s+1})P_2$.

In our work, the computational loads of the detector methods are very different from each other, i.e. $\mathcal{L}_s/\mathcal{L}_{s+1} \ll 1$. Thus within the BOA cascade the detector methods f_m $m = 1 \ldots M$ are ordered according to their computational loads. For notational simplicity we assume that the detector methods f_m used in a BOA cascade are enumerated such that for their computational loads $\mathcal{L}_m$ it holds that $\mathcal{L}_m \ll \mathcal{L}_{m+1}$, and now in a BOA cascade $f_s = f_m$. The Table 3 demonstrates the computational efficiency achieved in our experiments.

For a sample in a dataset X, the computational load of classification with a BOA cascade is on average

$$\sum_{m=1}^{S} \left(1 - \frac{\left| \left\{ \boldsymbol{x} \in X, \ \bigvee_{s=1}^{m-1} B_s^1(\boldsymbol{x}) \vee B_s^0(\boldsymbol{x}) = true \right\} \right|}{|X|} \right) \cdot \mathcal{L}_m.$$

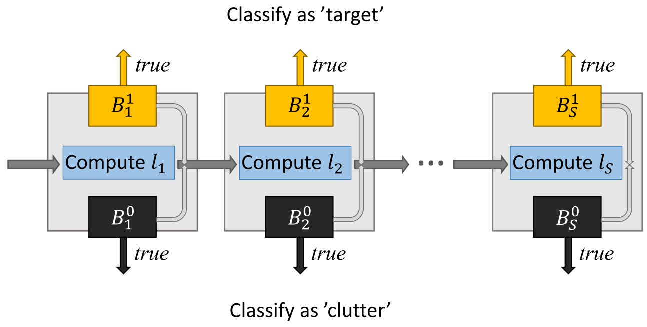


Fig. 3 BOA cascade. Classification process with BOA classification cascade. At each stage s of the cascade new target likelihood score $f_s(\boldsymbol{x}) = l_s$ is computed and either classification is made, or next stage is entered. The internal Boolean decision makers $B_1^1, B_2^1, \ldots, B_S^1$ are partitions of BOA function (7), and $B_1^0, B_2^0, \ldots, B_S^0$ are partitions of (8)

To design a specific type of BOA cascade, e.g., one-sided or symmetrical, the lists z_q, $q = 1 \ldots Q$, which determine the detector functions to be utilized within the conjunctions of the BOA, must be selected appropriately. For data with clearly unbalanced class distribution, one-sided cascade is computationally efficient if the early classification option is available for the prevalent class. This is the case if the decision-makers $B_s^{\text{prevalent}}$, $s = q \ldots S$ for the prevalent class are functioning while the decision-makers B_s^{rare}, $s = 1 \ldots S-1$ for the rare class are null/nonexistent as $B_s^{\text{rare}}(\boldsymbol{x}) = \mathit{false}\ \forall\ \boldsymbol{x}$. Thus, for the usual case, where the "target" class is rare and the "non-target" class is the prevalent one, to ensure a computationally efficient one-sided BOA cascade, the BOA must be conjunctive, designed with only one conjunction list $z_1 = [1, 2, \ldots, S]$. In case the target likelihood scores are negated, i.e., $-f_1(\boldsymbol{x}), -f_2(\boldsymbol{x}), \ldots, -f_S(\boldsymbol{x})$ are used, conjunction list of every subvector of $[1, 2, \ldots, S]$, should be used to build a one-sided cascade capable of early classification to "non-target" class. For example in case $S = 3$, the conjunction lists would thus be $z_1 = [1]$, $z_2 = [2]$, $z_3 = [3]$, $z_4 = [1, 2]$, $z_5 = [1, 3]$, $z_6 = [2, 3]$ and $z_7 = [1, 2, 3]$.

A symmetrical cascade, which enables early classification to both the classes at all the cascade stages, is suitable for classification tasks with both even and unbalanced class distributions. The time to decision efficiency of the cascade depends on capability of all the internal decision makers B_s^1 and B_s^0 for $s = 1 \ldots S$ of the cascade to make early classifications. Functioning decision makers for all the stages and both the classes to build a symmetrical BOA cascade are ensured by constructing the BOA from cumulative conjunction lists $z_1 = [1]$, $z_2 = [1, 2], \ldots, z_S = [1, 2, \ldots, S]$, that is $z_s = [1, 2, \ldots, s]$.

3.2 BOA tunability property

Classification performance of the BOA depends on all the values of thresholds $\theta_m^{q,n}$, $m = 1 \ldots M$, $q = 1 \ldots Q$, $n = 1 \ldots N_q$ in $\boldsymbol{\theta}$. Classifying data $X = \{X^0, X^1\}$ from two classes with a BOA $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ results in certain true positive rate $tpr_{\boldsymbol{\theta}}$ and false positive rate $fpr_{\boldsymbol{\theta}}$, which produces one point into a space of precision (P) vs. recall (R). Classifying the data X with the BOA $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$ with all the possible sets of different threshold values in $\boldsymbol{\theta}$ results in a constellation of performance points in (P, R) space. Best performing threshold values for the BOA are those corresponding to the classification performance on the upper frontier of this (P, R) constellation.

We want to make the BOA sensitivity tunable with a single parameter in similar way to individual detectors. For that, we introduce a parameter $\alpha \in [0 \ldots 1]$, which denotes the sensitivity setting of a BOA. A value of the parameter α corresponds to a fixed set $\boldsymbol{\theta}_\alpha$ of the BOA threshold values such that $\mathrm{B}(\boldsymbol{x}; \alpha) = \mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}_\alpha)$. In the next section, we introduce an algorithm to select threshold values for $\boldsymbol{\theta}_\alpha$ for a range of values of the sensitivity parameter α. These operating points result in the BOA performance to be close to the upper frontier of the (P, R) constellation of BOA performance with all the possible settings of $\boldsymbol{\theta}$.

The user may then select for a BOA $\mathrm{B}(\boldsymbol{x}; \alpha)$ the operating point α with the most desirable behavior with the factual costs of a false positive $\mathcal{C}_{fp}$ and a false negative $\mathcal{C}_{fn}$ of the problem. The operating point α^* of minimal expected misclassification cost can be found at

$$\alpha^* = \min_\alpha \left(P\left(\boldsymbol{x} \in X^1\right) \cdot \left(1 - tpr_\alpha\right) \cdot \mathcal{C}_{fn} + P\left(\boldsymbol{x} \in X^0\right) \cdot fpr_\alpha \cdot \mathcal{C}_{fp} \right).$$

where $P\left(\boldsymbol{x} \in X^1\right)$ and $P\left(\boldsymbol{x} \in X^0\right)$ are the prior probabilities of the classes.

3.3 The proposed algorithm to set parameters of a BOA

We train the BOA $\mathrm{B}(\boldsymbol{x}; \alpha)$ by finding suitable values for thresholds $\boldsymbol{\theta}_\alpha$ for a range of values of $\alpha \in [0 \ldots 1]$ in terms of training data X. The possible threshold values ϑ_m considered for a target likelihood score l_m are given by the scores of target class samples $\boldsymbol{x} \in X^1$ as $\vartheta_m = f_m(\boldsymbol{x}) = l_m$.

The proposed algorithm, BOAThresholdSearch, for training a BOA is presented in Algorithm 1. As input, the algorithm needs training data $X = \{X^0, X^1\}$ from two classes, the conjunction lists $z_1, z_2, \ldots, z_Q$, maximal conjunction set multiplicities $N_1, N_2, \ldots N_Q$ of the BOA and the maximal number $N_S^{\max}$of candidates for $\boldsymbol{\theta}_\alpha$ saved by the algorithm for each α. The algorithm produces sets $\boldsymbol{\theta}_{\alpha_t}$ of fixed threshold values for BOA operating points $\alpha_t = \frac{t}{T}$, $t = 0 \ldots T$, where T equals the number of samples $\boldsymbol{x} \in X^1$. These operating points correspond to true positive rates $0, \frac{1}{T}, \frac{2}{T}, \ldots, \frac{T-1}{T}, 1$ on training data X. The algorithm searches for suitable threshold values step by step starting by selecting values for $\boldsymbol{\theta}_0$ for $\alpha_0 = 0$ and terminating after selecting values for $\boldsymbol{\theta}_1$ for $\alpha_T = 1$. The method is greedy in a sense that when searching for values for α_t at iteration t, the search starts from a potential set of threshold values for α_{t-1} provided by iteration $t-1$, and the threshold values are allowed to change only gradually for minimizing the number of false positives locally.

The algorithm starts by fixing the BOA thresholds for sensitivity level $\alpha_0 = 0$ to be $\boldsymbol{\theta}_{\alpha_0} = \{\infty\}$. The BOA with parameter setting $\alpha_0 = 0$ does not accept any sample to the "target" class, i.e., $\mathrm{B}(\boldsymbol{x}; \alpha_0) = \mathit{false}\ \ \forall\ \ \boldsymbol{x} \in X$. Thus, the algorithm starts with $tpr_{\alpha_0} = fpr_{\alpha_0} = 0$. The threshold setting $\boldsymbol{\theta}_{\alpha_0}$ and the corresponding number 0 of false positives are placed into a set $\mathcal{S}_0$ as an entry $(\boldsymbol{\theta} = \{\infty\}, fp = 0)$ for the next step to start with.

At each step $t = 1 \ldots T$, every threshold setting $\boldsymbol{\theta}$, given by entries $(\boldsymbol{\theta}, fp)$ in $\mathcal{S}_{t-1}$, provided by the step $t-1$, is adjusted. One adjusted set $\boldsymbol{\theta}^{\text{new}}$ is obtained by mitigating one or multiple thresholds $\theta_m^{q,n} \in \boldsymbol{\theta}$ of one

Algorithm 1 An algorithm to find thresholds for a BOA combination

```
procedure BOAthresholdSearch
    input: $\mathcal{Z}, \mathrm{N}, \mathcal{F}, X, N_{\mathcal{S}}^{\max}$
        # $\mathcal{Z}$ contains the $Q$ conjunction lists $z_1, \ldots, z_Q$.
        # N contains the maximal conjunction multiplicities $N_1, \ldots, N_Q$.
        # $\mathcal{F}$ contains the detector functions $f_m$, $1 = 1 \ldots M$.
        # $X = X^0 \cup X^1$ is the training data from two classes.
        # $N_{\mathcal{S}}^{\max}$ is the maximal size of set $\mathcal{S}_t$ used within the algorithm.

    for m= 1...M do
        Set $\vartheta_m = \{f_m(\boldsymbol{x}) \mid \boldsymbol{x} \in X^1\}$
    end for
    Set $\alpha_0 = 0$, $\boldsymbol{\theta}_0 = \{\infty\}$, $tp_0 = 0$, $fp_0 = 0$
    Set $\mathcal{S}_0 = \{(\boldsymbol{\theta} = \{\infty\}, fp = 0)\}$.
    for $t = 1 \ldots |X^1|$ do
        Set $\mathcal{S}_t = \emptyset$
        for each $(\boldsymbol{\theta}, fp) \in \mathcal{S}_{t-1}$ do
            for each $(q, n)$ for $q = 1 \ldots Q$, $n = 1 \ldots N_q$ do
                for each subvector $\zeta$ of $z_q$ do
                    Set $\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}$
                    Within $\boldsymbol{\theta}^{\text{new}}$, set $\theta_m^{q,n} = \theta_m^{q,n} - \delta_m$, for all $m = \zeta(i)$, $i = 1 \ldots M_\zeta$
                        using such $\delta_m$ that $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$ accepts exactly one more
                        sample $\boldsymbol{x} \in X^1$ than $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$, and $\theta_m^{q,n} \in \vartheta_m$.
                    Count the number of false positives $fp^{\text{new}}$ with $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$.
                    Set $\theta_{\text{all}}^{\gamma,\nu} = \infty$ for every conjunction $(\gamma, \nu)$ of $\boldsymbol{\theta}^{\text{new}}$ which is redundant.
                    Set $\mathcal{S}_t = \mathcal{S}_t \cup (\boldsymbol{\theta}^{\text{new}}, fp^{\text{new}})$.
                end for
            end for
        end for
        Set $\alpha_t = \frac{t}{T}$, $tp_t = tp_{t-1} + 1$, $fp_t = \min_{(\boldsymbol{\theta}, fp) \in \mathcal{S}_t} fp$
        Set $\boldsymbol{\theta}_t = \boldsymbol{\theta}^*$ such that $(\boldsymbol{\theta}^*, fp_t) \in \mathcal{S}_t$ and within $\boldsymbol{\theta}^*$ the number of
            conjunctions $(\gamma, \nu)$ for which $\theta_{\text{all}}^{\gamma,\nu} < \infty$ is the smallest.
        Prune $\mathcal{S}_t$ by keeping $N_{\mathcal{S}}^{\max}$ entries with the smallest $fp$.
    end for
    return: $\alpha_t$, $\boldsymbol{\theta}_t$, $tp_t$, $fp_t$  $\forall$  $t = 1 \ldots |\{p\}|$.
end procedure
```

conjunction (q,n) of the BOA. Within each BOA conjunction (q,n), there are $2^{M_q} - 1$ subsets of thresholds $\left\{\theta_{z_q(i)}^{q,n} | i \subseteq \{1 \ldots M_q\}\right\}$ to search for the best change from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^{\text{new}}$. Thus in the complete BOA function there are $P = \sum_{q=1}^{Q} N_q \cdot \left(2^{M_q} - 1\right)$ possible subsets of thresholds to change, and thus one $\boldsymbol{\theta}$ generates up to P changed threshold settings $\boldsymbol{\theta}^{\text{new}}$.

When mitigating the values of thresholds $\left\{\theta_{z_q(i)}^{q,n} \mid i \subseteq \{1 \ldots M_q\}\right\}$ of a conjunction (q,n) from their values in $\boldsymbol{\theta}$ for $\boldsymbol{\theta}^{\text{new}}$, the amount of changes are such that $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})$ accepts exactly one more sample $\boldsymbol{x} \in X^1$ than $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta})$. That is, $\mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}) = \mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}^{\text{new}})\ \forall\ \boldsymbol{x} \in X^1 \backslash \boldsymbol{x}_*$, $\mathrm{B}(\boldsymbol{x}_*; \boldsymbol{\theta}) = \mathit{false}$ and $\mathrm{B}(\boldsymbol{x}_*; \boldsymbol{\theta}^{\text{new}}) = \mathit{true}$. If redundancy of BOA function appears with the new threshold set $\boldsymbol{\theta}^{\text{new}}$, all the thresholds $\theta_{z_q(i)}^{q,n}$, $i = 1 \ldots M_q$ of the redundant conjunctions (q,n) are reset to be $\theta_*^{q,n} = \infty$. All the acquired new settings $\boldsymbol{\theta}^{\text{new}}$ are saved with their resulting false positive counts into a set $\mathcal{S}_t$ as entries $\{(\boldsymbol{\theta}, fp)^{\text{new}}\}$ to be potential settings for α_t.

After processing every entry $(\boldsymbol{\theta}, fp) \in \mathcal{S}_{t-1}$ and saving all the generated new entries into $\mathcal{S}_t$, the best set $\boldsymbol{\theta}^*$ of BOA thresholds among the entries of $\mathcal{S}_t$ is selected for $\boldsymbol{\theta}_{\alpha_t} = \boldsymbol{\theta}^*$ to correspond to α_t. The best set $\boldsymbol{\theta}^*$ is a selected to be the one corresponding to the smallest number of false positives among the entries in $\mathcal{S}_t$ and using as few BOA conjunctions as possible with non-infinite thresholds. The set $\mathcal{S}_t$ is then pruned to keep the maximal allowed number $N_{\mathcal{S}}^{\max}$ of the best entries for the next step to start with. In the experiments, we used $N_{\mathcal{S}}^{\max} = 10$, as larger number did not improve the recognition accuracy notably while making the algorithm run remarkably slower.

Figure 4 illustrates the thresholds $\boldsymbol{\theta}_\alpha$ found by the algorithm with $N_{\mathcal{S}}^{\max} = 1$ for a BOA

$$\mathrm{B}(\boldsymbol{x}; \alpha) = \mathrm{B}(\boldsymbol{x}; \boldsymbol{\theta}_\alpha) = \left(l_1 \geq \theta_1^1\right) \vee \left[\left(l_1 \geq \theta_1^2\right) \wedge \left(l_2 \geq \theta_2^2\right)\right]. \tag{14}$$

for $\alpha = 0, \frac{1}{T}, \frac{2}{T}, \ldots, \frac{T-1}{T}, 1$.

The memory requirement of the algorithm, besides the training data and the output variables, during the algorithm run is the storage needed for the set $\mathcal{S}_t$ of the potential operating points to be stored at each iteration. As maximally $N_{\mathcal{S}}^{\max}$ operation points are passed from one iteration to the next one, the number of operation points to be held in memory during an iteration of the algorithm run is maximally $N_{\mathcal{S}}^{\max} \times \sum_{q=1}^{Q} N_q \left(2^{M_q} - 1\right)$.

Computational complexity of the BOATS algorithm is $\mathcal{O}\left(|X^1|\, N_{\mathcal{S}}^{\max}\, N_{\text{conj}}^{\max}\, 2^M\right)$. In practice, multiple positive samples are often selected concurrently, diminishing the multiplier $|X^1|$. The limit $N_{\mathcal{S}}^{\max}$ is an input parameter which allows the user to decide about the accuracy vs time and memory complexity trade-off of the algorithm. $N_{\text{conj}}^{\max}$ is the maximum number of conjunctions in the DNF-BF BOA-function, which takes place at the operating point of recall= 1. At operating points with lower recall values, the true value is generally lower, and using $N_{\text{conj}}^{\max}$ sets upper

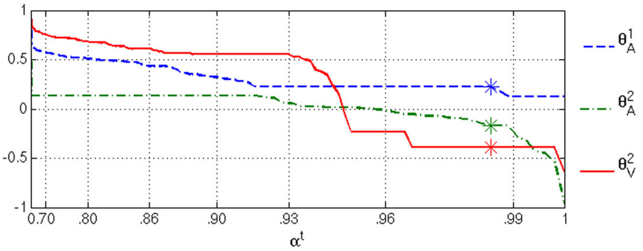


Fig. 4 BOA training. The sequence of thresholds $\boldsymbol{\theta}_{\alpha_t} = \left[\theta_A^1, \theta_A^2, \theta_V^2\right]_t$, $t = 0 \ldots T$ found by the proposed BOATS algorithm for a BOA (14) with $N_{\boldsymbol{S}}^{\max} = 1$. Thresholds of the operating point $\boldsymbol{\alpha}_t$ with highest accuracy on train data is marked with asterisks

limit for the time complexity. The number 2^M is upper limit of options tested when processing each conjunction, the true number for each conjunction (q, n) is $2^{M_q} - 1$.

4 Results and discussion

In this section, we report our experiments to evaluate the performance of the proposed BOA cascade of multiple sensitivity tunable detectors both in terms of detection accuracy and computational load of classification. We also analyze the proposed BOA training algorithm to showcase how good operating points it can find for a BOA combination. To substantiate the eligibility of our work, we compare the acquired results with others found in the literature.

We first introduce the datasets used for the two explored tasks, namely laughter detection and context change detection, and discuss the used performance measures. Then, we contrast our results with the proposed BOA classifier and a C5.0 -tree classifier in laughter detection task to results by other solutions found in the literature. We also compare the proposed BOA training algorithm to other training algorithms adopted from literature and explore the detection performance with different BOA combinations.

4.1 Data and performance measures

4.1.1 MAHNOB Laughter dataset

For laughter detection, i.e., laughter vs speech classification, we use data from the MAHNOB Laughter dataset of [7]. The data consists of 1399 video clips of lengths from 0.15 s to 28 s of 22 different persons. 845 of the video clips represent speech and 554 of them represent laughter. The data is recorded in two modilities; frontal closeup video with frame rate 25 fps, and audio from a lapel microphone with sampling frequency 44.1 kHz.

A frame from one of the videos is shown in Fig. 5 to demonstrate the data.

We run the tests using 22-fold cross-validation where at each fold the videos of one person are left out for testing, and all the rest of the videos are used for training.

We build the BOA combinations using similar classifiers as are used for the baseline method in [7]. Those are an audio stream based detector, which provides laughter likelihood $f_A(\boldsymbol{x}_{\text{audio}}) = l_A$, and a video frame based detector, which provides laughter likelihood $f_V(\boldsymbol{x}_{\text{visual}}) = l_V$ for each video clip. The computational load of the audio stream based detector is very small compared to the computational load of the visual stream based detector.

The audio stream-based laughter detector utilizes the 6 first MFCC features from audio frames of length 20 ms. A single output feedforward neural network (NN) is trained to produce audio frame-wise target class likelihoods l_a using mean squared error (MSE) error function. The NN has one hidden layer with 20 neurons and all the neurons of the network use tangential sigmoid transfer function. The target class likelihood l_A for a video clip is an average over the frame-wise values as $l_A = \frac{1}{N_a} \sum_{\tau=1}^{N_a} l_a(\tau)$, where N_a is the number of audio frames in the clip.

The video frame-based laughter detector starts with extracting the 20 face points, shown in Fig. 6, from each video frame using an algorithm from [49]. The utilized face points correspond to points used in [7]. Then, the dimensionality of each face point feature vector is

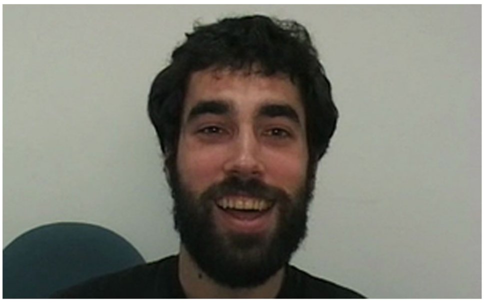


Fig. 5 A video frame from MAHNOB Laughter data set. A video frame from MAHNOB Laughter data set. This frame is from a video which contains laughter

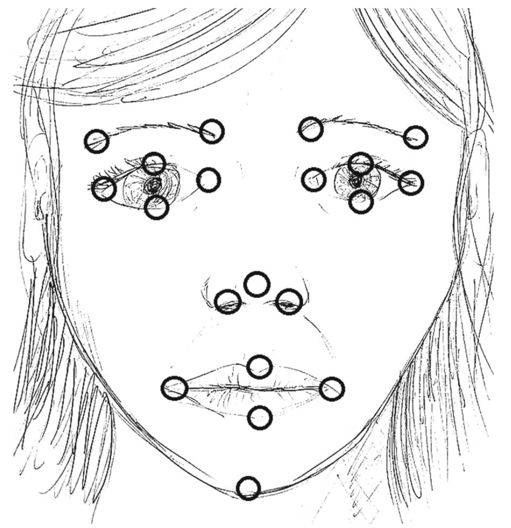


Fig. 6 Face points. The 20 face points used as features for laughter-speech classification of MAHNOB Laughter dataset videos

reduced from 40 to 20 by principal component analysis (PCA). For frame-wise laughter likelihood estimates l_v an NN is trained. It is built of 1 hidden layer of 10 neurons. All the neurons use tangential sigmoid transfer function, and mean squared error (MSE) loss function is applied for training. Video clipwise laughter likelihood is given as an average over the frame-wise values as $l_V = \frac{1}{N_V}\sum_{\tau=1}^{N_V} l_v(\tau)$, where N_V is the number video frames in the clip.

4.1.2 CASA dataset

For video context change detection problem we use CASA database[1] from [8]. Over 7 h of lifelog video material is filmed with a small pen camera, which operates at frame rate 15 frames/second and frame size 176×144 pixels. The stereo sound track is recorded by a pair of in-ear microphones with 44.1 kHz sampling rate and stored without compression. The database contains video material from 23 different types of environments.

For a context change detection task we created 30 video files of length 5–20 min. Each file is concatenated on average of 105 clips of length 1–30 s from the video material of CASA database. The context—one of the 23 different environments included in the database—is kept the same for 1–5 successive clips, otherwise each clip is taken from a randomly selected video file. There are on average 42 context changes within each created video file. We run our tests using 6-fold cross-validation, where at each fold 5 files are reserved for testing and the remaining 25 files are used for system training.

We use three different detectors to spot context changes in the created videos. Brief descriptions of the used detectors are given here, while the details of them can be found in [50]. The fastest one of the used detectors operates on the audio stream of the video. The audio is analyzed in frames of length 80 ms with 40 ms overlap of successive frames. From each audio frame, MFCC features are computed, and within a sliding window of 125 audio frames, mean and variance of 20 MFCC coefficients are computed. Transitions in these statistics are converted to a context change likelihood l_1 for each audio frame. The computation time of scores l_1 on a single CPU desktop computer is 0.8 ms per audio frame, that is 10 ms per one second of audio.

Two other utilized context change detectors operate on the image modality of the video. The faster one of the detectors on visual modality collects RGB histograms of video frames and produces the context change likelihood value l_2 for each video frame according to the city block distance between adjacent RGB histograms. The computation time of l_2 is about 29 ms per video frame, that is approximately 435 ms per one second of video.

The more accurate one of the used detectors on visual modality, proposed in [51], counts incidences of SIFT descriptor codebook elements within each video frame, and collects a SIFT histogram, i.e., so-called bag-of-words feature vector, for each video frame. The context change likelihood value l_3 for each video frame is computed as the city block distance between SIFT-histograms of successive video frames. The computation time of l_3 is about 12.3 s per video frame, that makes about 184 s per one second of video.

4.1.3 Performance measures

In the literature the performance of detectors is often presented by a receiver operation characteristic (ROC) curve. However, in our evaluations, we prefer the curve of *precision* vs *recall* (P-R curve) because in case of imbalanced class distributions P-R curve is more faithful to the absolute number of erroneous classifications than the ROC -curve of *tpr* in respect to *fpr*. To demonstrate the performance of a certain operating point of a detector, we use measures like accuracy, F_1-score, and computational load.

Average values of these performance numbers over cross-validation folds are presented as results. With MAHNOB laughter dataset, 22-fold cross-validation is used. In each fold, video files of one speaker are used for testing, and the rest of the files are used for training the component detectors and the BOA -cascade. With CASA dataset, 6-fold cross validation is used similarly. In each fold, 25 video files are used for training the individual classifiers and the BOA -combination, and 5 files are used for testing the system.

4.2 Comparing BOA cascade to existing work in laughter vs speech classification

We compare the performance of the proposed BOA cascade to results we obtained with C5.0 -tree building algorithm [52] as well as results obtained by other authors in laughter vs speech classification, i.e., laughter detection, with the MAHNOB laughter dataset. For the task, we use a BOA detector

$$\mathrm{B}(\boldsymbol{x};\boldsymbol{\theta}) = \left(l_A \geq \theta_A^1\right) \vee \bigvee_{n=1}^{N_2} \left[\left(l_A \geq \theta_A^{2,n}\right) \wedge \left(l_V \geq \theta_V^{2,n}\right) \right], \tag{15}$$

whose threshold parameters θ_A^1, $\theta_A^{2,1}$, $\theta_V^{2,1}$, $\theta_A^{2,2}$, $\theta_V^{2,2}$, $\ldots$,$\theta_A^{2,N}$, $\theta_V^{2,N}$ are learned by the proposed training algorithm. The Fig. 7 illustrates the BOA cascade of (15). The computational load of acquiring l_A from audio stream is only a fraction of the load of computing l_V from video frames. Thus, the ratio of samples that need the computation of l_V reflects well the average computational load of classifying a sample with BOA cascade of (15).

Table 1 presents results with C5.0 tree building algorithm as well as those found in the literature in contrast to our solution. We report performance numbers with a BOA cascade of (15) with $N = 1$ and also with N selected adaptively by the proposed training algorithm. The decision trees obtained with C5.0 algorithm [52] are converted to DNF-BF -form (15) and evaluated in cascaded manner similarly to BOA evaluation. The number N in the DNF-BF (15) of a tree varies according to the structure of the tree, which is given by the algorithm. The minimal leaf size of a tree was defined by 10-fold cross validation using the training data. The boosted C5.0 forest contains 10 trees trained with different weightings by the training algorithm on training samples. The classification of the forest is obtained via voting by the trees.

The C5.0 forest outperforms all the other solutions in terms of classification accuracy, whereas the performance of single C5.0 tree is comparable to performance obtained with BOA classifiers. When a C5.0 tree is evaluated in cascaded manner, very similar computational savings as with a BOA cascade are obtained. Both the BOA detectors outperform the solutions of [53, 54], albeit the classifier in [54] is trained with another database, which likely explains its lower detection accuracy. The results obtained by [55] reach similar accuracy and F_1-scores than our BOA cascades, but their result is not fully comparable as they use only a subset of 15 speakers out of 22 used by all the other authors. However, the computational load of our solution is significantly lower, compared to all these other multimodal solutions. With our BOA cascade of (15) with $N = 1$, only 11% of samples needed the computation of l_V, thus it is about nine times faster than the other solutions. The BOA cascade of (15) with N selected by the proposed training algorithm reaches slightly higher accuracy than the reference solutions while being still three times faster than them.

4.3 Comparing training algorithms for BOA combination

We use the CASA lifelog data and the context change detection task for illustrating the capability of the proposed training algorithm to find successful operating points for a BOA combination. For context change detection we use BOA combinations built of three detectors,

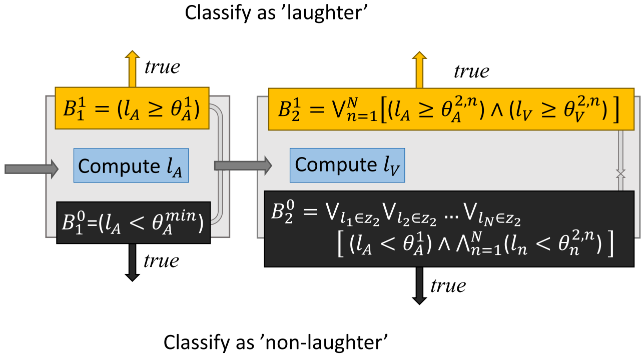

Classify as ’laughter’
true
$B_1^1 = (l_A \geq \theta_A^1)$
Compute l_A
$B_1^0{=}(l_A < \theta_A^{min})$
true
true
$B_2^1 = \vee_{n=1}^N[(l_A \geq \theta_A^{2,n}) \wedge (l_V \geq \theta_V^{2,n})\,]$
Compute l_V
$B_2^0 = \vee_{l_1\in z_2}\vee_{l_2\in z_2}\ldots\vee_{l_N\in z_2}$
$[\,(l_A < \theta_A^1) \wedge \wedge_{n=1}^N(l_n < \theta_n^{2,n})\,]$
true
Classify as ’non-laughter’

Fig. 7 Symmetrical BOA cascade for laughter detection. Symmetrical BOA cascade, which is capable of making early classification to both classes, realizing of BOA (15) for laughter detection. The conjunction set $z_2 = \{l_A, l_V\}$ and the threshold θ_A^{min} is $\theta_A^{min} = \min\left(\theta_A^1, \theta_A^{2,1}, \theta_A^{2,2}, \ldots, \theta_A^{2,N}\right)$. B_2^0 contains $K = 2^N$ conjunctions, where the first threshold comparison is always $\left(l_A \geq \theta_A^1\right)$. The comparisons indexed by $n{=}1\ldots N$ operate either on l_A or l_V according to binary (M_2-ary) N-digit representation of the conjunction index $k{=}1\ldots K$, bin(k). If the n:th digit of bin(k) is 0, l_A is used, and l_V is used if the n:th digit of bin(k) is 1

Table 1 Results in laughter detection task

	acc.	F_1^{sp}	F_1^{lg}	v.f. %
$(l_A \geq \theta_A)$	95.1	.944	.927	0%
$(l_V \geq \theta_V)$	84.4	.818	.795	100%
BOA cascade of (15), $N = 1$	96.0	.966	.958	11%
BOA cascade of (15), N by BOATS	96.9	.972	.955	33%
C5.0 tree	96.7	.965	.948	23%
Boosted C5.0 forest	97.3	.978	.958	≈ 100%
[53]	92.7	.943	.905	100%
[54][a]	91.7	.932	.893	100%
[55][b]	96.9	.973	.963	100%

Comparison of laughter detectors on MAHNOB laughter data. The used measures of performance are the overall accuracy, F_1 -scores for both speech (F_1^{sp}) and laughter (F_1^{lg}), and percentage of video clips, the classification of which utilized also visual features (**v.f.**). The BOA detectors are used at the operating point $\boldsymbol{\alpha}$ of the highest accuracy on training set

[a] Comparison with [54] is not directly comparable, as the classifier in [54] is trained with another dataset

[b] Results of [55] are with 15 speakers while the other authors use 22 speakers in their tests

which are introduced in "Data and performance measures." We train the thresholds of a BOA with the proposed training algorithm (BOATS) and two reference algorithms adapted form literature, and then compare the resulting F_1-scores of classification. The reference algorithms that we use for this evaluation are iterative exhaustive search (IES) based on work in [11] and Boolean algebra of ROC curves (BAROC) introduced in [10]. The implementations of IES and BAROC, adapted for BOA training, are presented in Algorithms 2, 3 and 4 in Appendix 1. The iterative framework used of both the algorithms is presented in Algorithm 2. The Algorithm 3 shows the core operations of IES, and the Algorithm 4 presents the operations for BAROC.

Figure 8 shows the F_1-scores with operating points obtained with three algorithms, BOATS, IES and BAROC, for a BOA

$$\mathrm{B}_{\mathrm{AND}} = \bigvee_{n=1}^{N} \left[\left(l_1 \geq \theta_1^n\right) \wedge \left(l_2 \geq \theta_2^n \wedge \left(l_3 \geq \theta_3^n\right) \right] \quad (16)$$

with different conjunction multiplicities N. The IES algorithm can be seen to find the best operating point when $N = 1$ with its exhaustive search. However, when N is increased, IES is unable to improve the BOA performance due to that the suboptimal operating points of each individual conjunction, which nevertheless might produce better performance when used within a disjunctive combination, are pruned by the algorithm.

The BAROC algorithm performs worse than the other algorithms due to its assumption of detector independence, which does not hold with the two visual stream based detectors. Moreover, by the definition of the Boolean algebra of ROC curves in (4) and (5), BAROC is unable to find the opportunities provided by utilizing multiple conjunctions over the same conjunction set.

The proposed BOATS algorithm finds suboptimal operating points for the BOA, but is able to utilize the opportunities offered by using multiple conjunctions over the same conjunction set, and thus outperforms the IES algorithm with $N > 1$. The performance ceases to improve when the conjunction multiplicity grows larger than 7. This is due to both the data characteristics and algorithm behavior favoring small number of conjunctions, i.e., small N.

In a Table 2, we show the best F_1-scores of the operating points found by the three algorithms for BOA combinations

$$\begin{aligned}
\mathrm{B}_{\mathrm{OR}} &= (\, l_1 \geq \theta_1 \,) \vee (\, l_2 \geq \theta_2 \,) \vee (\, l_3 \geq \theta_3 \,) \\
\neg\mathrm{B}_{\neg\mathrm{OR}} &= \neg\Big[(-l_1 \geq \theta_1) \vee (-l_2 \geq \theta_2) \vee (-l_3 \geq \theta_3) \Big] \\
\mathrm{B}_{\mathrm{AND}} &= \bigvee_{n=1}^{N} \Big[\; \left(l_1 \geq \theta_1^n\right) \wedge \left(l_2 \geq \theta_2^n\right) \wedge \left(l_3 \geq \theta_3^n\right) \; \Big]
\end{aligned} \qquad (17)$$

$$\begin{aligned}
\mathrm{B}_{\mathcal{P}} &= \bigvee_{q=1}^{7} \; \bigvee_{n=1}^{N_q} \; \bigwedge_{i=1}^{M_q} \left(l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n} \right) \\
\neg\mathrm{B}_{\neg\mathcal{P}} &= \neg\left[\bigvee_{q=1}^{7} \; \bigvee_{n=1}^{N_q} \; \bigwedge_{i=1}^{M_q} \left(-l_{z_q(i)} \geq \theta_{z_q(i)}^{q,n} \right) \right].
\end{aligned}$$

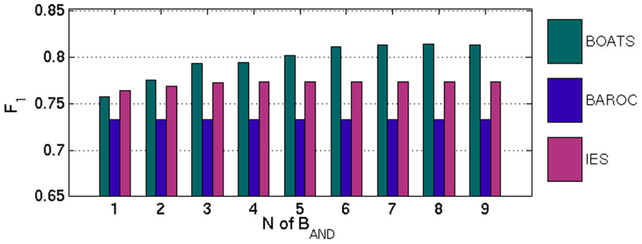


Fig. 8 Experiments 1. Comparing context change detection performance of BOA $\mathrm{B}_{\mathrm{AND}}$ (16) with different conjunction multiplicities N when the thresholds are selected by different algorithms

Table 2 Comparison of algorithms for BOA training

BOA	BOATS	IES	BAROC
B_{OR}	68.3	68.6	68.2
$\mathrm{B}_{\mathrm{AND}}, N=1$	75.7	76.5	73.5
$\neg\mathrm{B}_{\neg\mathrm{OR}}$	76.3	73.2	73.2
$\mathrm{B}_{\mathrm{AND}}, N=6$	81.1	77.3	73.3
$\neg\mathrm{B}_{\neg\mathcal{P}}, N_q=1$	81.3	73.4	73.7
$\mathrm{B}_{\mathcal{P}}, N_q=1$	80.0	81.8	80.5
$\mathrm{B}_{\mathcal{P}}, N_q$ by BOATS	81.7		

Average test F_1-score over sixfold cross-validation sets in context change detection task with BOA combinations (17) trained with different algorithms. The used operating point of the BOA is the one with highest F_1-score on train data separately for each CV-fold

where the conjunction lists of $\mathrm{B}_{\mathcal{P}}$ and $\mathrm{B}_{\neg\mathcal{P}}$ are $z_1=[1], z_2=[2], z_3=[3], z_4=[1,2], z_5=[1,3], z_6=[2,3], z_7=[1,2,3]$.

For the disjunctive BOA B_{OR}, the operating points found by the three algorithms are very similar. The IES algorithm finds the best operating point for this BOA with its exhaustive search. The proposed BOATS algorithm does not leave far behind, nor does the Boolean algebra for ROC curves. The assumption of the BAROC algorithm about the detector independence, which does not hold with these detectors, does not impair its performance in training the BOA B_{OR}, where only disjunctive **OR** -operator is used.

The conjunctive BOA $\mathrm{B}_{\mathrm{AND}}$ with $N=1$ and the disjunctive $\neg\mathrm{B}_{\neg\mathrm{OR}}$ have equally expressive decision boundaries. Ideally they would result in identical classifiers, but due to characteristics of the training algorithms they result in having different thresholds. Similarly the ideal decision boundaries of $\mathrm{B}_{\mathrm{AND}}$ with $N=6$ and $\neg\mathrm{B}_{\neg\mathcal{P}}$ coincide. Results with those pairs of BOAs trained with the BOATS and BAROC algorithms are similar, which was expected because of the similarity of decision boundaries. The iterative exhaustive search does not find as good operating points for the BOA combinations when negative scores $-l_m$ are used. This is due to the selection of the threshold values to test, which in this case of using negative detector scores is done based on "non-target" class samples, as explained in Section 3.3.

For the BOA $\mathrm{B}_{\mathcal{P}}$ with $N_q=1, q=1\ldots Q$, the IES algorithm is able to find the best performing operating point. The iterative exhaustive search is thus effective in finding good thresholds for BOAs with different conjunctions. IES was not run with $N_q>1$, because of its extremely long computation time for such a long BOA. BAROC algorithm finds a comparable operating point for the BOA $\mathrm{B}_{\mathcal{P}}$ with $N_q=1, q=1\ldots Q$. This is probably due to the abundance of different conjunctions in the BOA to be combined disjunctively, where the inaccurate independence assumption of BAROC does not matter so much. Also for the BAROC -algorithm, the result with the BOA $\mathrm{B}_{\mathcal{P}}$ with $N_q>1$ is not reported, because it is the same as with $N_q=1$ by definition. The proposed BOATS algorithm leaves slightly behind IES and BAROC for the BOA $\mathrm{B}_{\mathcal{P}}$ with $N_q=1\,\forall q$. However, when the conjunction multiplicities N_q are unlimited, BOATS finds an operating point with similar performance with the best one with $N_q=1\,\forall q$ by IES.

4.4 Computational efficiency of BOA

In this section, we report performance of different BOA cascades in terms of both F_1-score and the average computational load of classification in respect to real time processing. The BOA cascades are trained with the proposed BOATS algorithm for the video context change detection task which has highly unbalanced class distribution, the "no change" class being the prevalent one.

The BOA cascades $\mathrm{B}_{\mathrm{AND}}$, $\neg\mathrm{B}_{\neg\mathrm{OR}}$, and $\neg\mathrm{B}_{\neg\mathcal{P}}$ of (17) correspond to one-sided cascades with early classification opportunity to the "no change" class. They are assumed to be computationally efficient with this data, where samples of "no change" form the large majority of data. The BOA cascades of B_{OR} and $\mathrm{B}_{\mathcal{P}}$ are one-sided, having the early classification opportunity solely to the "target" class. They are likely to be slow with this data.

For this comparison we use, in addition to the BOA cascades of (17), symmetrical cascades realizing of

$$
\begin{aligned}
\mathrm{B}_{\mathrm{C2}} &= \left(l_1 \geq \theta_1^1\right) \vee \bigvee_{n=1..N} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right)\right] \\
\mathrm{B}_{\mathrm{C3}} &= \left(l_1 \geq \theta_1^1\right) \vee \bigvee_{n=1..N_2} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right)\right] \\
&\quad \vee \bigvee_{n=1..N_3} \left[\left(l_1 \geq \theta_1^{2,n}\right) \wedge \left(l_2 \geq \theta_2^{2,n}\right) \wedge \left(l_3 \geq \theta_3^{2,n}\right)\right].
\end{aligned}
\tag{18}
$$

The BOA cascade of B_{C2} is similar to the laughter detection cascade of Fig. 7 with $l_A=l_1$ and $l_V=l_2$. The cascade of B_{C3} with $N_2=N_3=1$ is illustrated in Fig. 9.

In Table 3, we show for the different BOA cascades their best F_1-scores as well as their computation times (CT) of classification using a desktop PC in respect to real time processing. The individual detectors $d_m=(l_m \geq \theta_m)$, $m=1,2,3$ have very different computational loads. Compared to real time processing, the detectors d_1 and d_2 are very fast, d_3 being extremely slow.

The BOA cascade of B_{C2} with $N=1$ has a computational cost of only a fraction of real time, while achieving an outstanding improvement of classification performance over individual detectors $d_m=(l_m \geq \theta_m)$, $m=1,2,3$. It requires a tiny fraction of the computational load of d_3 and less than 5% of the computational load of d_2 while only doubling the time of the fastest detector d_1.

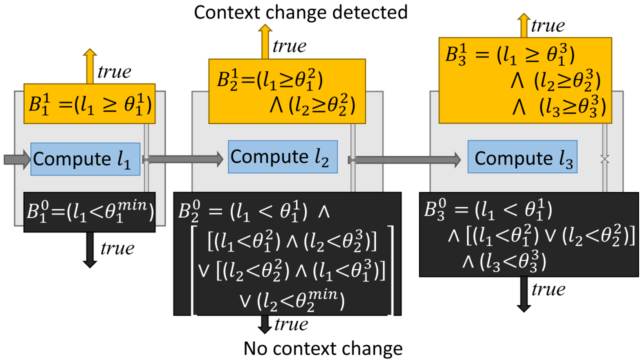

Context change detected
true
$B_1^1=(l_1 \geq \theta_1^1)$
Compute l_1
$B_1^0=(l_1<\theta_1^{min})$
true
true
$B_2^1=(l_1\geq\theta_1^2) \wedge (l_2\geq\theta_2^2)$
Compute l_2
$B_2^0 = (l_1 < \theta_1^1) \wedge [(l_1<\theta_1^2) \wedge (l_2<\theta_2^3)] \vee [(l_2<\theta_2^2) \wedge (l_1<\theta_1^3)] \vee (l_2<\theta_2^{min})$
true
No context change
true
$B_3^1 = (l_1 \geq \theta_1^3) \wedge (l_2\geq\theta_2^3) \wedge (l_3\geq\theta_3^3)$
Compute l_3
$B_3^0 = (l_1 < \theta_1^1) \wedge [(l_1<\theta_1^2) \vee (l_2<\theta_2^2)] \wedge (l_3<\theta_3^3)$
true

Fig. 9 BOA cascade example. The BOA cascade of B_{C3} with $N_1=N_2=N_3=1$. Threshold θ_m^{min} means $\theta_m^{min} = \min\left(\theta_m^2, \theta_m^3\right)$

At the same time it reaches $F_1 = .764$, which is about 9 percent units higher than .674 of d_1, 14 percent units higher than .525 of d_2 and 11 percent units higher than .553 of d_3. With N of B_{C2} not restricted, the F_1-score further improves to .778, but the computational benefit over always computing both l_1 and l_2 is lost.

The BOA cascade of B_{C3} utilizes all the three available detectors. Thus, the F_1-scores obtained with it are all the more improved from those obtained with B_{C2}. Real time processing is compromised by incorporating the extremely slow computation of l_3. However, with the cascade processing, the total computational load of B_{C3} with $N_2 = N_3 = 1$ is reduced to less than 2% of that of always computing all the scores l_1, l_2, and l_3. At the same time the F_1-score is improved to 0.774. With N_2 and N_3 unrestricted and selected by the proposed BOA training algorithm, F_1-score improves further to 0.813, the average computation time being still less than 4% of the time of always computing all the scores l_1, l_2 and l_3.

Table 3 Performance comparison of different BOA cascades

BOA	F_1	CT
$d_1 = (l_1 \geq \theta_1)$	.674	0.01
$d_2 = (l_2 \geq \theta_2)$	.525	0.43
$d_3 = (l_3 \geq \theta_3)$	.553	184
$\mathrm{B}_{\mathrm{C2}}, N_q = 1\ \forall q$	.764	0.02
$\mathrm{B}_{\mathrm{C2}}, N_q$ by BOATS	.778	0.44
$\mathrm{B}_{\mathrm{C3}}, N_q = 1\ \forall q$	.774	2.5
$\mathrm{B}_{\mathrm{C3}}, N_q$ by BOATS	.813	6.9
$\neg\mathrm{B}_{\neg\mathrm{OR}}$	.763	4.1
$\neg\mathrm{B}_{\neg\mathcal{P}}, N_q = 1\ \forall q$	.813	7.0
$\mathrm{B}_{\mathrm{AND}}, N$ by BOATS	.814	8.5
B_{OR}	.683	182.0
$\mathrm{B}_{\mathcal{P}}, N_q = 1\ \forall q$	.798	153.3
$\mathrm{B}_{\mathcal{P}}, N_q$ by BOATS	.817	124.3

Results in terms of F_1-score and computation time (CT) in respect to video time in scene detection task with detectors d_1, d_2, d_3 and BOA combinations of (17) and (18). The BOA thresholds are selected with the proposed BOATS algorithm. The results are test averages over sixfold cross validation sets. The used operating point of each BOA is the one with highest F_1-score on train data separately for each CV-fold

When observing the computational loads of different BOA cascades in Table 3, we may notice that remarkable computational savings appear whenever the BOA utilizes the computationally heaviest detector function $f_3(\boldsymbol{x}) = l_3$ only by combining it conjunctively with the faster detector functions $f_1(\boldsymbol{x}) = l_1$ and $f_2(\boldsymbol{x}) = l_2$. This is the case in BOA combinations $\mathrm{B}_{\mathrm{C2}}, \mathrm{B}_{\mathrm{C3}}, \mathrm{B}_{\mathrm{AND}}, \neg\mathrm{B}_{\neg\mathrm{OR}}$ and $\neg\mathrm{B}_{\neg\mathcal{P}}$. The BOA cascades of B_{OR} and $\mathrm{B}_{\mathcal{P}}$ utilize a conjunction list $z_3 = [3]$, which means using the threshold comparison $\left(l_3 \geq \theta_3^3\right)$ as an individual conjunction within the BOA function. Because of this these BOA cascades can not avoid computing l_3 unless the input is accepted to the rare "context change detected" class by conjunctions using only scores l_1 and l_2. The BOA cascade of B_{OR} is computationally the most inefficient, as it is able to avoid computing l_3 only if the input is classified as "context change detected" by threshold comparison $(l_1 \geq \theta_1)$ or $(l_2 \geq \theta_2)$. The BOA cascade of $\mathrm{B}_{\mathcal{P}}$ is slightly more efficient due to its conjunctions $\bigvee_{n=1}^{N_q}\left(l_1 \geq \theta_1^{q,n}\right) \wedge \left(l_2 \geq \theta_2^{q,n}\right)$, based on conjunction list $z_q = [1,2]$, capable of classifying the input as "context change detected" with only l_1 and l_2.

The best F_1-score, $F_1 = .814$, among the BOA cascades not utilizing a conjunction list $z_q = [3]$ is achieved with $\mathrm{B}_{\mathrm{AND}}$. Only slightly higher score, $F_1 = .817$, was obtained with BOA cascade $\mathrm{B}_{\mathcal{P}}$, but the computational efficiency obtainable with a cascade structure is obstructed by its computationally inefficient BOA design.

Precision vs recall curves of the detectors $d_1 = (l_1 \geq \theta)$, $d_2 = (l_2 \geq \theta)$, and $d_3 = (l_3 \geq \theta)$, and some BOA combinations of them trained with the BOATS algorithm are shown in Fig. 10. We can see that all the BOA combinations improve the precision-recall curve over the curves of the individual detectors remarkably.

5 Conclusions

We proposed to use a monotone Boolean function for combining multiple binary classifiers and showed how to implement it as a computationally efficient binary classification cascade. The proposed Boolean OR of ANDs (BOA) cascade is defined by a BF over multiple detector scores, and it is implemented as a classification cascade for computational efficiency. We also presented an algorithm, BOA threshold search (BOATS), for learning the thresholds of a BOA cascade.

We showed experimentally that the BOA cascade achieves the state-of-the-art performance in laughter detection task with MAHNOB laughter dataset while requiring much less computational power than the other solutions found in the literature. We also showed that the proposed algorithm suits best for learning thresholds of a BOA combination, compared to other learning strategies for Boolean combinations found in the literature. Finally, we explored the detection performance of different BOA cascades in terms of their F_1-scores and computational loads of detection. We showed that a BOA cascade improves the classification accuracy remarkably over the individual detectors while mostly requiring only a fraction of their combined computation time.

Endnote

[1] Demonstration available at http://arg.cs.tut.fi/demo/CASAbrowser/

Appendix 1

Reference algorithms for BOA training

The Algorithm 2 contains the functionality for training a Boolean BOA combination iteratively, by fusing two elements at a time. The symbol $\boldsymbol{\Theta}$ denotes a matrix of thresholds. Each row of $\boldsymbol{\Theta}$ contains one threshold setting for the corresponding Boolean classifier. The boldface symbols **tp** and **fp** are used to denote vectors of true positives and false positives resulting with different threshold settings in $\boldsymbol{\Theta}$ of a corresponding Boolean classifier, respectively.

One conjunction (q, n) of a BOA is built on lines 8–22 within the loop starting from line 7. On lines 23–28, the newly trained conjunction is combined with the conjunctions trained already.

The algorithm returns thresholds for found operating points α of the BOA in matrix $\boldsymbol{\Theta}_B$. The corresponding true positive rates and false positive rates on training data are returned in vectors $\mathbf{tp}_B$ $\mathbf{fp}_B$ We use this framework for training a BOA with either Boolean algebra of ROC curves (BAROC) by [10] or iterative exhaustive search (IES) by [11].

The training algorithm to be used is selected by a variable ALG. If ALG = IES, the combining is performed with Algorithm 3, and if ALG = BAROC, the combination of two sets of thresholds is done by Algorithm 4.

Appendix 2

Boolean decision makers at BOA cascade stages

At each stage $s = 1 \dots S$ of a BOA cascade, one target likelihood score $f_m(\boldsymbol{x}) = l_m = l_s, m \in \{1 \dots M\}$ is computed. All the scores l_i, $i = 1..s$ are thus available at cascade stage s to make the classification or the decision to enter the next cascade stage. BFs $B_1^1(l_1)$, $B_1^0(l_1)$, $B_2^1(l_1, l_2)$, $B_2^0(l_1, l_2)$,..., $B_S^1(l_1, l_2, \dots, l_S)$ and $B_S^1(l_1, l_2, \dots, l_S)$ are set to make these internal decisions of the cascade. As illustrated in Fig. 3, at each stage s, after computing the predefined target likelihood score l_s, a classification to the "target" class is made if $B_s^1(l_1, l_2, \dots, l_s) = true$ and a classification to the "non-target" class is made if $B_s(l_1, l_2, \dots, l_s)^0 = true$. If both the functions, B_s^0 and B_s^1, output *false*, the next cascade stage is entered. The functions B_S^0 and B_S^1 at the last cascade stage S are negations of each other ensuring the classification to be made.

The decision makers B_s^1, $s = 1 \dots S$ are partitions of the BOA function (7), and the functions B_s^0, $s = 1 \dots S$ are partitions of the negation (8) of the BOA function. This ensures that the decision makers B_s^1, B_s^0, $s = 1 \dots S$ are

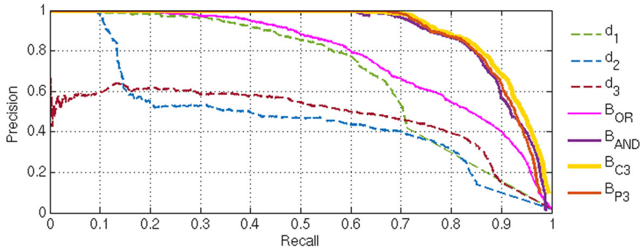


Fig. 10 P-R curves of BOA cascades. Precision vs. recall curves of detectors d_1, d_2, and d_3 and some BOA combinations of them

Algorithm 2 An algorithm to find thresholds for a BOA combination with IES or BAROC procedure

```
procedure ITERATIVEBC
    input: $\mathcal{Z}$, N, , $\mathcal{F}$, $X$, ALG
      # $\mathcal{Z}$ contains $z_1, z_2, \ldots, z_Q$.
      # N contains $N_1, N_1, \ldots, N_Q$.
      # $\mathcal{F}$ contains the detector functions $f_m$, $m = 1 \ldots M$.
      # $X = X^0 \cup X^1$ contains the training data.
      # ALG is either IES or BAROC

    for $m = 1 \ldots M$ do
        Set $\vartheta_m = \{f_m(\boldsymbol{x}) \mid \boldsymbol{x} \in X^1\}$
    end for
    Set $B(\boldsymbol{x};\boldsymbol{\theta}) = false$,  $\boldsymbol{\Theta}_B = \emptyset$,  $\mathbf{tp}_B = 0$,  $\mathbf{fp}_B = 0$
    for $q = 1 \ldots Q$ and $n = 1 \ldots N_q$ do
        Set $m = z_q(1)$
        Set $B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta}) = \left(f_m(\boldsymbol{x}) \geq \theta\right)$
        Set $\boldsymbol{\Theta}_{\text{conj}} = \vartheta_m$
        $\mathbf{tp}_{\text{conj}} = \left|\{\ \boldsymbol{x} \mid \boldsymbol{x} \in X^1, B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\Theta}_{\text{conj}}) = true\ \}\right|$
        $\mathbf{fp}_{\text{conj}} = \left|\{\ \boldsymbol{x} \mid \boldsymbol{x} \in X^1, B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\Theta}_{\text{conj}}) = false\ \}\right|$
        for $i = 2, 3, \ldots, M_q$ do
            Set $n = z_q(i)$
            Set $\boldsymbol{\Theta}_n = \vartheta_n$
            if ALG=IES then
                Set new $\boldsymbol{\Theta}_{\text{conj}}$, $\mathbf{tp}_{\text{conj}}$, $\mathbf{fp}_{\text{conj}}$ according to
                    ES2($B_{\text{conj}}$, $\boldsymbol{\Theta}_{\text{conj}}$, $\wedge$, $(l_n \geq \theta)$, $\boldsymbol{\Theta}_n$, DATA)
            else if ALG = BAROC then
                Set new $\boldsymbol{\Theta}_{\text{conj}}$, $\mathbf{tp}_{\text{conj}}$, $\mathbf{fp}_{\text{conj}}$ according to
                    BA2(($|X^1|\mathbf{tp}_{\text{conj}}$, $|X^0|\mathbf{fp}_{\text{conj}}$), $\boldsymbol{\Theta}_{\text{conj}}$, $\wedge$, $\left(|X^1|\mathbf{tp}_n, |X^0|\mathbf{fp}_n\right)$, $\boldsymbol{\Theta}_n$)
            end if
            Update $B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta}) = B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta}) \wedge (f_n(\boldsymbol{x}) \geq \theta)$
        end for
        if ALG=IES then
            Set new $\boldsymbol{\Theta}_B$, $\mathbf{tp}_B$, $\mathbf{fp}_B$ according to
                ES2($B$, $\boldsymbol{\Theta}_B$, $\vee$, $B_{\text{conj}}$, $\boldsymbol{\Theta}_{\text{conj}}$, DATA)
        else if ALG=BAROC then
            Set new $\boldsymbol{\Theta}_B$, $\mathbf{tp}_B$, $\mathbf{fp}_B$ according to
                BA2($\left(|X^1|\mathbf{tp}_B, |X^0|\mathbf{fp}_B\right)$, $\boldsymbol{\Theta}_B$, $\vee$, $\left(|X^1|\mathbf{tp}_{\text{conj}}, |X^0|\mathbf{fp}_{\text{conj}}\right)$, $\boldsymbol{\Theta}_{\text{conj}}$)
        end if
        Update $B(\boldsymbol{x};\boldsymbol{\theta}) = B(\boldsymbol{x};\boldsymbol{\theta}) \vee B_{\text{conj}}(\boldsymbol{x};\boldsymbol{\theta})$
    end for

    return: $\boldsymbol{\Theta}_B$, $\mathbf{tp}_B$, $\mathbf{fp}_B$.
end procedure
```

Algorithm 3 Exhaustive search of ROCCH operating points for a Boolean combination of two detectors

```
procedure ES2
    input: $B_1$, $\boldsymbol{\Theta}_1$, $\star$, $B_2$, $\boldsymbol{\Theta}_2$, DATA
      # $\star$ is the Boolean operator to be used, either AND ($\wedge$) or OR ($\vee$)

    for $p = 1 \ldots$ number of sets $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}_1$ do
        for $q = 1 \ldots$ number of sets $\boldsymbol{\theta}$ in $\boldsymbol{\Theta}_2$ do
            Set $\mathbf{tp}(p,q) = \left|\left\{\boldsymbol{x} \mid \overset{\boldsymbol{x} \in X^1}{B_1(\boldsymbol{x};\boldsymbol{\Theta}_1(p)) \star B_2(\boldsymbol{x};\boldsymbol{\Theta}_2(q)) = true}\right\}\right|$
            Set $\mathbf{fp}(p,q) = \left|\left\{\boldsymbol{x} \mid \overset{\boldsymbol{x} \in X^0}{B_1(\boldsymbol{x};\boldsymbol{\Theta}_1(p)) \star B_2(\boldsymbol{x};\boldsymbol{\Theta}_2(q)) = true}\right\}\right|$
        end for
    end for
    for $fp = 0 \ldots |X^0|$ do
        Find $(p^*, q^*) = \arg\max_{\mathbf{fp}(p,q)=fp}\ \mathbf{tp}(p,q)$
        Set $\mathbf{fp}_{\text{ROCCH}}(fp) = fp$
        Set $\mathbf{tp}_{\text{ROCCH}}(fp) = \mathbf{tp}(p^*, q^*)$
        Set $\boldsymbol{\Theta}_{\text{ROCCH}}(fp) = \left[\boldsymbol{\Theta}_1(p^*), \boldsymbol{\Theta}_2(q^*)\right]$
    end for

    return: $\boldsymbol{\Theta}_{ROCCH}$, $\mathbf{tp}_{\text{ROCCH}}$, $\mathbf{fp}_{\text{ROCCH}}$.
end procedure
```

consistent. This means that both B_s^1 and B_s^0 never output *true* concurrently, i.e. if $B_s^1(\boldsymbol{x}) = true$ then $B_s^0(\boldsymbol{x}) = false$ and similarly if $B_s^0(\boldsymbol{x}) = true$ then $B_s^1(\boldsymbol{x}) = false$. It also means that if classification is made by B_s^1 or B_s^0 at a cascade stage s, the decision makers B_r^1 and B_r^0 of the other stages $r = 1 \ldots S,\ r \neq s$ would not make contradicting classifications. Formally, if $\exists s\ B_s^c = true$ then $B_r^{\neg c} = false \quad \forall r \in 1 \ldots S$.

The internal decision makers at BOA cascade stages $s = 1 \ldots S$ for the "target" class are

$$B_s^1(\boldsymbol{x};\alpha) = \bigvee_{z_q \,\middle|\, \substack{\exists j \quad s=z_q(j), \\ \nexists j m=z_q(j), m>s}} \bigvee_{n=1}^{N_q} \bigwedge_{i=1}^{M_q} \left(f_{z_q(i)}(\boldsymbol{x}) \geq \theta_{z_q(i)}^{q,n}\right). \tag{19}$$

That is, B_s^1 contains the conjunctions (q,n) of the BOA (7) that utilize the newly computed likelihood score l_s and possibly those computed at earlier stages, but naturally none of the scores $l_m,\ m > s$. Examples can be seen in Figs. 7 and 9.

Similarly, the internal decision makers $B_s^0,\ s = 1 \ldots S$ of the BOA cascade for the "non-target" class are partitioned from the negated BOA; B_s^0 contains the conjunctions k of the BOA (8) that utilize the newly computed likelihood score l_s and possibly those computed at earlier stages, but none of the scores $l_m,\ m > s$. The partition of the

Algorithm 4 Combining ROC curves of two detectors using the Boolean algebra of ROC curves by [10]

```
procedure BA2
    input: $(\mathbf{tpr}_1, \mathbf{fpr}_1)$, $\mathbf{\Theta}_1, \star, (\mathbf{tpr}_2, \mathbf{fpr}_2)$, $\mathbf{\Theta}_2$
      # $\star$ is the Boolean operator to be used, either AND ($\wedge$) or OR ($\vee$)

    for $fpr = 0 \ldots 1$ do
        if $\star = \wedge$ then
            Find $(p^*, q^*) = \arg \max_{\mathbf{fpr}_1(p) \cdot \mathbf{fpr}_2(q) = fpr} \mathbf{tpr}_1(p) \cdot \mathbf{tpr}_2(q)$
        else if $\star = \vee$ then
            Find $(p^*, q^*) = \arg \max_{\mathbf{fpr}_1(p) + \mathbf{fpr}_2(q) - \mathbf{fpr}_1(p) \cdot \mathbf{fpr}_2(q) = fpr} \mathbf{tpr}_1(p) + \mathbf{tpr}_2(q) - \mathbf{tpr}_1(p) \cdot \mathbf{tpr}_2(q)$
        end if
        Set $\mathbf{fpr}_{12}(fpr) = fpr$
        Set $\mathbf{tpr}_{12}(fpr) = \mathbf{tpr}_1(p^*) \cdot \mathbf{tpr}_2(q^*)$
        Set $\mathbf{\Theta}_{12}(fpr) = \left\{\mathbf{\Theta}_1(p^*), \mathbf{\Theta}_2(q^*)\right\}$
    end for

    return: $\mathbf{\Theta}_{12}$, $(\mathbf{tpr}_{12}, \mathbf{fpr}_{12})$.
end procedure
```

K conjunctions of $\neg$B of (8) is given by a Boolean variable $\mathbf{c}_s(k)$, which denotes whether the k:th conjunction of the negated BOA (8) is used for decision maker B_s^0. It is recursively defined as

$$\mathbf{c}_0(k) = \mathit{false} \quad \forall\, k = 1 \ldots K$$

$$\mathbf{c}_s(k) = \bigwedge_{r=1}^{s-1} \neg \mathbf{c}_r(k) \wedge \bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \bigwedge_{m=s+1}^{S} \neg \left[z_q(\mathcal{I}(k,q,n)) = m\right], \tag{20}$$

where $\mathcal{I}(k,q,n)$ is given by (9). The first part of the Eq. (20) makes sure that the conjunction k has not been used for B_r^0, $r < s$, while the rest of the equation checks whether detector functions beyond f_s, i.e., any of $f_{s+1}, f_{s+2}, \ldots, f_S$, are used in the conjunction k of (8) and sets $\mathbf{c}_s(k) = \mathit{false}$ if so.

Now, the decision-makers B_s^0 for the "non-target" class are

$$B_s^0 = \bigvee_{k \,\middle|\, \substack{k \in \{1 \ldots K\} \\ \mathbf{c}_s(k) = \mathit{true}}} \left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \left(f_{z_q(\mathcal{I}(k,q,n))}(\boldsymbol{x}) < \theta^{q,n}_{z_q(\mathcal{I}(k,q,n))} \right) \right], \tag{21}$$

where the detector function indicator index $\mathcal{I}(k,q,n)$ is given by (9), $K = \prod_{q=1}^{Q} M_q^{N_q}$ and BOA variables z_q, N_q, for $q = 1 \ldots Q$ are adopted from (8). Using the alternative notation of the $\neg$B (8), the decision makers B_s^0, $s = 1 \ldots S$ for the "non-target" class may be written as

$$B_s^0(\boldsymbol{x};\boldsymbol{\theta}) = \bigvee_{\substack{i_{1,1}=1 \ldots M_1 \\ z_1(i_{1,1}) \le s}} \bigvee_{\substack{i_{1,2}=1 \ldots M_1 \\ z_1(i_{1,2}) \le s}} \cdots \bigvee_{\substack{i_{1,N_1}=1 \ldots M_1 \\ z_1(i_{1,N_1}) \le s}} \bigvee_{\substack{i_{2,1}=1 \ldots M_2 \\ z_2(i_{2,1}) \le s}} \cdots \bigvee_{\substack{i_{2,N_2}=1 \ldots M_2 \\ z_2(i_{2,N_2}) \le s}} \bigvee_{\substack{i_{Q,1}=1 \ldots M_Q \\ z_Q(i_{Q,1}) \le s}} \cdots \bigvee_{\substack{i_{Q,N_Q}=1 \ldots M_Q \\ z_Q(i_{Q,N_Q}) \le s}} \left[\bigwedge_{q=1}^{Q} \bigwedge_{n=1}^{N_q} \left(f_{z_q(i_{q,n})}(\boldsymbol{x}) < \theta^{q,n}_{z_q(i_{q,n})} \right) \right]. \tag{22}$$

This notation, while possibly being more comprehensible, includes all the decision makers B_r^0, $r < s$ in B_s^0, however this redundancy does not affect the functionality.

Abbreviations

BAROC: Boolean algebra of ROC curves; BF: Boolean function; BOA: **OR** of **AND**s function; BOATS: An algorithm to search thresholds for a BOA detector; CNF: Conjunctive normal form; CNF-BF: Boolean function in conjunctive normal form; CPU: Central processing unit; DNF: Disjunctive normal form; DNF-BF: Boolean function in disjunctive normal form; IBC: Iterative Boolean combination; IES: Iterative exhaustive search; LAD: Logical analysis of data; MFCC: Mel-frequency cepstral coefficient; OCAT: One clause at a time; RGB: Red-green-blue color format; ROC: Receiver operating characteristic curve; ROCCH: ROC convex hull; SIFT: Scale invariant feature transform

Acknowledgements

Discussions with Prof. Jiří Matas leveraged the research from implementing ad hoc ideas to doing excogitated research. The authors would also like to thank prof. Bhaskar Rao on useful discussions related to cascades and BFs. Furthermore, the authors want to thank the anonymous reviewers for their well-informed comments for improving the manuscript.

Funding

Funding for this research was provided by the Tampere University of Technology.

Availability of data and materials

Mahnob Laughter dataset is located at https://mahnob-db.eu/laughter/. CASA dataset is located at http://arg.cs.tut.fi/demo/CASAbrowser/.

Authors' contributions
KM has written the manuscript and implemented and executed the experiments. TV has been involved in designing the experiments as a supervisor and helped KM in writing the manuscript in a solid scientific way. JK gave the initial idea of utilizing Boolean functions for combining classifiers. He also provided his help in software related problems. All authors read and approved the final manuscript.

Ethics approval and consent to participate
People appearing in the videos of Mahnob Laughter dataset have given their consent for data usage for research purposes.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 2 May 2017 Accepted: 6 July 2018
Published online: 24 July 2018

References

1. S Yang, P Luo, C-C Loy, X Tang, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Wider face: A face detection benchmark, (2016)
2. S Zhang, R Benenson, M Omran, J Hosang, B Schiele, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. How far are we from solving pedestrian detection? (2016)
3. T Virtanen, A Mesaros, T Heittola, MD Plumbley, P Foster, E Benetos, M Lagrange, *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*. (Tampere University of Technology. Department of Signal Processing, 2016). ISBN (Electronic): 978-952-15-3807-0
4. J Ashbourn, *Biometrics: Advanced Identity Verification: the Complete Guide*. (Springer, 2014)
5. A Courbet, D Endy, E Renard, F Molina, J Bonnet, Detection of pathological biomarkers in human clinical samples via amplifying genetic switches and logic gates. Sci. Transl. Med. **7**(289) (2015)
6. E Boros, PL Hammer, T Ibaraki, A Kogan, Logical analysis of numerical data. Math. Program. **79**(1), 163–190 (1997)
7. S Petridis, B Martinez, M Pantic, The MAHNOB laughter database. Image Vis. Comput. **31**(2), 186–202 (2013)
8. A Mesaros, T Heittola, A Eronen, T Virtanen, in *Signal Processing Conference, 2010 18th European*. Acoustic event detection in real life recordings (IEEE, 2010), pp. 1267–1271
9. J Daugman, *Biometric Decision Landscapes, vol. 482*. (University of Cambridge, Computer Laboratory, 2000)
10. ME Oxley, SN Thorsen, CM Schubert, in *Information Fusion, 2007 10th International Conference On*. A boolean algebra of receiver operating characteristic curves (IEEE, 2007), pp. 1–8
11. Q Tao, R Veldhuis, Threshold-optimized decision-level fusion and its application to biometrics. Pattern Recog. **42**(5), 823–836 (2009)
12. K Venkataramani, BV Kumar, in *Multimedia Content Representation, Classification and Security*. Role of statistical dependence between classifier scores in determining the best decision fusion rule for improved biometric verification (Springer, 2006), pp. 489–496
13. M Barreno, A Cardenas, JD Tygar, in *Advances in Neural Information Processing Systems 20*. Optimal roc curve for a combination of classifiers, (2008), pp. 57–64
14. W Khreich, E Granger, A Miri, R Sabourin, Iterative boolean combination of classifiers in the roc space: an application to anomaly detection with hmms. Pattern Recognit. **43**(8), 2732–2752 (2010)
15. E Granger, W Khreich, R Sabourin, Fusion of biometric systems using boolean combination: an application to iris-based authentication. Int. J. Biometrics. **4**(3), 291–315 (2012)
16. C Shen, On the principles of believe the positive and believe the negative for diagnosis using two continuous tests. J. Data Sci. **6**, 189–205 (2008)
17. Y Crama, PL Hammer, *Boolean Functions: Theory, Algorithms, and Applications. Encyclopedia of Mathematics and its Applications*. (Cambridge University Press, 2011)
18. G Alexe, S Alexe, TO Bonates, A Kogan, Logical analysis of data – the vision of peter l. hammer. Ann. Math. Artif. Intell. **49**(1), 265–312 (2007)
19. I Chikalov, V Lozin, I Lozina, M Moshkov, HS Nguyen, A Skowron, B Zielosko, *Logical Analysis of Data: Theory, Methodology and Applications*. (Springer, Berlin, 2013), pp. 147–192
20. PL Hammer, Partially defined boolean functions and cause-effect relationships. Lecture in International Conference on Multi-attribute Decision Making Via OR-based Expert Systems (1986)
21. RS Michalski. (RS Michalski, JG Carbonell, TM Mitchell, eds.) (Springer, Berlin, Heidelberg, 1983), pp. 83–134
22. AP Kamath, NK Karmarkar, KG Ramakrishnan, MGC Resende, A continuous approach to inductive inference. Math. Program. **57**(1), 215–238 (1992)
23. T Fawcett, An introduction to roc analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
24. P Hess, Dedekind's problem: monotone boolean functions on the lattice of divisors of an integer. Pacific J. Math. **81**(2), 411–415 (1979)
25. RS Michalski, in *V international Symposium on Information Processing (FCIP 69), Vol A3 (Switching Circuits)*. On the quasi-minimal solution of the general covering problem, (1969)
26. AS Deshpande, E Triantaphyllou, A greedy randomized adaptive search procedure (grasp) for inferring logical clauses from examples in polynomial time and some extensions. Math. Comput. Model. **27**(1), 75–99 (1998)
27. F Pawley, A Syder, The one-clause-at-a-time hypothesis. Perspect. Fluen, 163–199 (2000)
28. JR Quinlan, Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
29. JR Quinlan, *C4.5: Programs for Machine Learning*. (Morgan Kaufmann Publishers Inc., San Francisco, 1993)
30. L Breiman, JH Friedman, RA Olshen, CJ Stone, *Classification and Regression Trees*. (Chapman & Hall, New York, 1984)
31. PL Hammer, A Kogan, B Simeone, S Szedmák, Pareto-optimal patterns in logical analysis of data. Discrete Appl. Math. **144**(1-2), 79–102 (2004)
32. S Alexe, PL Hammer, Accelerated algorithm for pattern detection in logical analysis of data. Discret. Appl. Math. **154**(7), 1050–1063 (2006). Discrete Mathematics and Data Mining II (DM and DM II)
33. TO Bonates, PL Hammer, A Kogan, Maximum patterns in datasets. Discret. Appl. Math. **156**(6), 846–861 (2008). Discrete Mathematics and Data Mining II
34. RS Michalski, I Mozetic, J Hong, N Lavrac, in *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence. AAAI'86*. The multi-purpose incremental learning system aq15 and its testing application to three medical domains (AAAI Press, 1986), pp. 1041–1045
35. RE Reinke, in *Machine Intelligence 11*, ed. by JE Hayes, D Michie, and J Richards. Incremental Learning of Concept. Descriptions: A Method and. Experimental Results (Clarendon Press Oxford, 1988)
36. SN Sanchez, E Triantaphyllou, J Chen, TW Liao, An incremental learning algorithm for constructing boolean functions from positive and negative examples. Comput. Oper. Res. **29**(12), 1677–1700 (2002)
37. R Feraund, OJ Bernier, J-E Viallet, M Collobert, A fast and accurate face detector based on neural networks. IEEE Trans. Pattern Anal. Mach. Intell. **23**(1), 42–53 (2001)
38. P Viola, MJ Jones, Robust real-time face detection. Int. J. Comput. Vis. **57**(2) (2001)
39. L Lefakis, F Fleuret, in *NIPS*. Joint cascade optimization using a product of boosted classifiers, (2010)
40. MJ Saberian, N Vasconcelos, Learning optimal embedded cascades. IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 2005–2018 (2012)
41. C Shen, P Wang, S Paisitkriangkrai, A van den Hengel, Training effective node classifiers for cascade classification. Int. J. Comput. Vis. **103**, 326–347 (2013)
42. H Li, Z Lin, X Shen, J Brandt, G Hua, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. A convolutional neural network cascade for face detection, (2015), pp. 5325–5334
43. VC Raykar, B Krishnapuram, S Yu, in *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*. Designing efficient cascaded classifiers: tradeoff between accuracy and cost, (2010)
44. M Chen, Z Xu, KQ Weinberger, O Chapelle, D Kedem, in *AISTATS*. Classifier cascade for minimizing feature evaluation cost, (2012)

45. J Sochman, J Matas, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Waldboost - learning for time constrained sequential detection, (2005)
46. T Wu, S-C Zhu, in *ICCV*. Learning near-optimal cost-sensitive decision policy for object detection, (2013)
47. MM Dundar, J Bi, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Joint optimization of cascaded classifiers for computer aided detection, (2007)
48. C Zhang, P Viola, in *NIPS*. Multiple-instance pruning for learning efficient cascade detectors, (2007)
49. X Zhu, D Ramanan, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Face detection, pose estimation, and landmark localization in the wild, (2012), pp. 2879–2886
50. K Mahkonen, J-K Kämäräinen, T Virtanen, in *Computer Vision-ACCV 2014 Workshops*. Lifelog scene change detection using cascades of audio and video detectors (Springer, 2014), pp. 434–444
51. J Lankinen, J-K Kämäräinen, in *VISAPP (1)*. Video shot boundary detection using visual bag-of-words, (2013), pp. 788–791
52. R Research, C5.0. http://rulequest.com/download.html. Accessed 2018
53. O Rudovic, S Petridis, M Pantic, in *Proceedings of the 21st ACM International Conference on Multimedia*. Bimodal log-linear regression for fusion of audio and visual features (ACM, 2013), pp. 789–792
54. S Petridis, V Rajgarhia, M Pantic, Comparison of Single-model and Multiple-model Prediction-based Audiovisual Fusion, ISCA Speech Organisation (2015)
55. H Rao, Z Ye, Y Li, MA Clements, A Rozga, JM Rehg, in *Joint Conference on Facial Analysis, Animation and Audio-Visual Speech Processing (FAAVSP)*. Combining acoustic and visual features to detect laughter in adults' speech, (2015), pp. 153–156

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-4229-9

ISSN 1459-2045