Robust Speech Recognition for German and Dialectal Broadcast Programmes

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Diplom-Ingenieur
Michael Stadtschnitzer

aus

Köflach, Österreich

Bonn 2018

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr.-Ing. Christian Bauckhage
2. Gutachter:	Prof. Dr. Stefan Wrobel
Tag der Promotion:	17. Oktober 2018
Erscheinungsjahr:	2018

Zusammenfassung

Audio-Mining-Systeme analysieren automatisch große Mengen heterogener Mediendateien wie Fernseh- und Radioprogramme, so dass der analysierte Audioinhalt effizient nach gesprochenen Wörtern durchsucht werden kann. Typischerweise bestehen Systeme wie das Audio-Mining-System des Fraunhofer IAIS aus mehreren Modulen zur Strukturierung und Analyse der Daten. Das wichtigste Modul ist hierbei das Modul für die automatische und kontinuierliche Spracherkennung mit großem Vokabular, das das Audiosignal in geschriebenen Text umwandelt. Aufgrund der enormen Entwicklung auf dem Gebiet der Spracherkennung und um den Kunden ein leistungsfähiges Audio-Mining-System zur Verfügung zu stellen, muss das Spracherkennungsmodul mit großen Trainingsdatenmengen regelmäßig mit den neuesten State-of-the-Art-Algorithmen trainiert und aktualisiert werden, die von der Forschungsgemeinschaft bereitgestellt werden. Heutzutage arbeiten Spracherkennungssysteme normalerweise unter sauberen Bedingungen sehr gut, wenn jedoch Geräusche, Nachhall oder dialektale Sprecher vorhanden sind, verschlechtert sich die Leistung dieser Systeme beträchtlich. In Rundfunkmedien sind typischerweise eine große Anzahl verschiedener Sprecher mit hoher Variabilität vorhanden, wie etwa Moderatoren, Interviewer, Befragte, mit oder ohne Umgangssprache, mit oder ohne Dialekt oder sogar mit Voice-Over. Insbesondere in regionalen Programmen der öffentlichen Rundfunkübertragung spricht ein beträchtlicher Teil der Sprecher mit einem Akzent oder einem Dialekt. Außerdem tritt eine große Menge verschiedener Hintergrundgeräusche in den Daten auf, wie Hintergrundsprache oder Hintergrundmusik. Nachbearbeitungsalgorithmen wie Kompression, Expansion und Stereo-Effekt-Verarbeitung, die in Rundfunkmedien großzügig verwendet werden, manipulieren die Audiodaten noch zusätzlich. All diese Probleme machen die Spracherkennung in der Rundfunkdomäne zu einer herausfordernden Aufgabe.

Diese Arbeit konzentriert sich auf die langfristige Entwicklung und Optimierung des deutschen Spracherkennungssystems, das Teil des Audio-Mining-Systems des Fraunhofer IAIS ist, und behandelt Probleme der Robustheit, die in deutschen Rundfunkprogrammen auftreten können, sowie die Anforderungen an das Spracherkennungssystem für einen produktiven industriellen Einsatz des Audio-Mining-Systems, was Faktoren wie Stabilität, Dekodierzeit und Speicherverbrauch umfasst.

Wir adressieren die folgenden drei Probleme: die kontinuierliche Entwicklung und Optimierung des deutschen Spracherkennungssystems über einen langen Zeitraum, die

schnelle automatische Suche nach den optimalen Spracherkennungsdekodierparametern und den Umgang mit deutschen Dialekten im deutschen Spracherkennungssystem für die Rundfunkdomäne.

Um eine hervorragende Leistung über lange Zeiträume zu gewährleisten, aktualisieren wir das System regelmäßig mit den neuesten Algorithmen und Systemarchitekturen, die von der Forschungsgemeinschaft zur Verfügung gestellt wurden, und evaluieren hierzu die Leistung der Algorithmen im Kontext der deutschen Rundfunkdomäne. Wir erhöhen auch drastisch die Trainingsdaten, indem wir einen großen und neuartigen Sprachkorpus der deutschen Rundfunkdomäne annotieren, der in Deutschland einzigartig ist.

Nach dem Training eines automatischen Spracherkennungssystems ist ein Spracherkennungsdekoder dafür verantwortlich, die wahrscheinlichste Texthypothese für ein bestimmtes Audiosignal zu dekodieren. Typischerweise benötigt der Spracherkennungsdekoder eine große Anzahl von Hyperparametern, die normalerweise auf Standardwerte gesetzt oder manuell optimiert werden. Diese Parameter sind oft weit von dem Optimum in Bezug auf die Genauigkeit und die Dekodiergeschwindigkeit entfernt. Moderne Optimierungsalgorithmen für Dekoderparameter benötigen allerdings eine lange Zeit, um zu konvergieren. Daher nähern wir uns in dieser Arbeit der automatischen Dekoderparameteroptimierung im Kontext der deutschen Spracherkennung in der Rundfunkdomäne in dieser Arbeit an, sowohl für die uneingeschränkte als auch für die eingeschränkte Dekodierung (in Bezug auf die Dekodiergeschwindigkeit), indem ein Optimierungsalgorithmus für den Einsatz in der Spracherkennung eingeführt und erweitert wird, der noch nie zuvor im Kontext der Spracherkennung verwendet wurde.

In Deutschland gibt es eine große Vielfalt an Dialekten, die oft in den Rundfunkmedien, vor allem in regionalen Programmen, vorhanden sind. Dialektale Sprache verursacht eine stark verschlechterte Leistungsfähigkeit des Spracherkennungssystems aufgrund der Nichtübereinstimmung von Phonetik und Grammatik. In dieser Arbeit beziehen wir die große Vielfalt deutscher Dialekte ein, indem wir ein Dialektidentifizierungssystem einführen, um den Dialekt des Sprechers abzuleiten, und um nachfolgend angepasste dialektale Spracherkennungsmodelle zu verwenden, um den gesprochenen Text zu erhalten. Für das Training des Dialektidentifizierungssystems wurde eine neuartige Datenbank gesammelt und annotiert.

Indem wir uns mit diesen drei Themen befassen, gelangen wir zu einem Audio-Mining-System, das ein leistungsstarkes Spracherkennungssystem beinhaltet, das in der Lage ist, dialektale Sprecher zu bewältigen und mit optimalen Dekoderparametern, die schnell berechnet werden können.

Abstract

Audio mining systems automatically analyse large amounts of heterogeneous media files such as television and radio programmes so that the analysed audio content can be efficiently searched for spoken words. Typically audio mining systems such as the Fraunhofer IAIS audio mining system consist of several modules to structure and analyse the data.

The most important module is the large vocabulary continuous speech recognition (LVCSR) module, which is responsible to transform the audio signal into written text. Because of the tremendous developments in the field of speech recognition and to provide the customers with a high-performance audio mining system, the LVCSR module has to be trained and updated regularly by using the latest state-of-the-art algorithms provided by the research community and also by employing large amounts of training data. Today speech recognition systems usually perform very well in clean conditions, however when noise, reverberation or dialectal speakers are present, the performance of these systems degrade considerably. In broadcast media typically a large number of different speakers with high variability are present, like anchormen, interviewers, interviewees, speaking colloquial or planned speech, with or without dialect, or even with voice-overs. Especially in regional programmes of public broadcast, a considerable fraction of the speakers speak with an accent or a dialect. Also, a large amount of different background noises appears in the data, like background speech, or background music. Post-processing algorithms like compression, expansion, and stereo effect processing, which are generously used in broadcast media, further manipulate the audio data. All these issues make speech recognition in the broadcast domain a challenging task.

This thesis focuses on the development and the optimisation of the German broadcast LVCSR system, which is part of the Fraunhofer IAIS audio mining system, over the course of several years, dealing with robustness related problems that arise for German broadcast media and also dealing with the requirements for the employment of the ASR system in a productive audiomining system for the industrial use including stability, decoding time and memory consumption.

We approach the following three problems: the continuous development and optimisation of the German broadcast LVCSR system over a long period, rapidly finding the optimal ASR decoder parameters automatically and dealing with German dialects in the German broadcast LVCSR system.

To guarantee superb performance over long periods of time, we regularly re-train

the system using the latest algorithms and system architectures that became available by the research community, and evaluate the performance of the algorithms on German broadcast speech. We also drastically increase the training data by annotating a large and novel German broadcast speech corpus, which is unique in Germany.

After training an automatic speech recognition (ASR) system, a speech recognition decoder is responsible to decode the most likely text hypothesis for a certain audio signal given the ASR model. Typically the ASR decoder comes with a large number of hyperparameters, which are usually set to default values or manually optimised. These parameters are often far from the optimum in terms of accuracy and decoding speed. State-of-the-art decoder parameter optimisation algorithms take a long time to converge. Hence, we approach the automatic decoder parameter optimisation in the context of German broadcast speech recognition in this thesis for both unconstrained and constrained (in terms of decoding speed) decoding, by introducing and extending an optimisation algorithm that has not been used for the task of speech recognitinon before to ASR decoder parameter optimisation.

Germany has a large variety of dialects that are also often present in broadcast media especially in regional programmes. Dialectal speakers cause severely degraded performance of the speech recognition system due to the mismatch in phonetics and grammar. In this thesis, we approach the large variety of German dialects by introducing a dialect identification system to infer the dialect of the speaker in order to use adapted dialectal speech recognition models to retrieve the spoken text. To train the dialect identification system, a novel database was collected and annotated.

By approaching the three issues we arrive at an audio mining system that includes a high-performance speech recognition system, which is able to cope with dialectal speakers and with optimal decoder parameters that can be inferred quickly.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Dr. Daniel Stein and Dr. Christoph Schmidt for their continuous support of my studies and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing this thesis.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Dr.-Ing. Christian Bauckhage, Prof. Dr. Stefan Wrobel, for their insightful comments and encouragement.

My sincere thanks also go to Dr.-Ing. Joachim Köhler, who provided me the opportunity to join his team, and who gave me access to the department facilities, and also for his precious pieces of advice regarding this thesis.

Also, I would like to express my gratitude to *Bayerischer Rundfunk* and *Schweizer Rundfunk und Fernsehen* for the close collaboration and who provided us precious data for research purposes. Without their help, this research would not have been possible to conduct.

I thank my fellow mates in the department for stimulating discussions and for all the fun we had in the last few years.

Last but not least, I thank my family and friends for their motivation and their patience through this intense phase of my life.

Contents

Chapter 1

Introduction

1.1 Audio Mining

Digital media archives are composed of a vast amount of heterogeneous media content files, which are typically annotated only scarcely, manually and inconsistently. Searching in the data is often a challenging task and retrieving the seeked information is considered to be a lucky strike in the majority of cases.

Audio mining systems solve this problem by automatically analysing vast amounts of heterogeneous media content files. After processing the data, the database can be efficiently searched based on the analysis results. A typical audio mining system like the Fraunhofer IAIS audio mining system is composed of several modules (e.g. speaker segmentation, gender detection, automatic speech recognition, speaker diarisation, speaker identification, keyword generation) that employ sophisticated algorithms and models which are trained on large amounts of training data. In order to guarantee a successful audio mining system for long periods of time, the modules have to be updated regularly by using the latest state-of-the-art algorithms and by the usage of sufficient amounts of training data. One of the most important modules of an audio mining system is the automatic speech recognition module, which is responsible to convert the audio speech signal into the written text and to provide the time boundaries (start and end time) of the spoken words. The analysis results of the speech recognition module are also often used as the input for subsequent modules like the keyword extraction module and therefore, highly performant and robust algorithms have to be used.

1.2 Robust Speech Recognition

Automatic speech recognition (ASR) is the technique to automatically transform an audio speech signal into written text. Speech recognition systems typically consist of an acoustic model, a pronunciation lexicon and a language model. A graph search algorithm like the Viterbi algorithm [1] decodes the most likely text hypothesis from the audio signal given the model. The acoustic model represents the relationship between the audio signal and the linguistic units that make up speech (usually phonemes,

syllables, senones or whole words) and is built by modelling statistical representations (e.g. Hidden-Markov-Models [2]) of the sound units by using audio recordings of speech and their corresponding text transcriptions. The pronunciation lexicon is a mapping between the vocabulary words and the corresponding units e.g. a sequence of phonemes. The language model calculates the probability distributions over sequences of words. Usually speech recognition systems perform very well in conditions similar to the training data. However, if there is a mismatch between the training condition and the testing condition, these systems typically degrade. Mismatches can occur e.g. due to background noises, reverberation, or due to speaker variabilities like accents and dialects. In the last few decades tremendous efforts have been made to improve the speech recognition algorithms. In the last few years neural network based architectures superseded the classical approach based on Gaussian mixture models. Within very short periods of time different types of neural network architectures became state-of-the-art in the automatic speech recognition research community. Typically the algorithms are developed by the exploitation of broadly used standard datasets from a certain domain, e.g. the Switchboard corpus [3], which is a corpus containing English telephone speech. It is unclear whether the advances reported for a certain language and domain directly translate to another specific language in a different domain.

Hence, and in order to guarantee a successful Fraunhofer IAIS audiomining system which relies on constant development of the speech recognition system, we, amongst other things, approach the continuous development and optimisation of the large-vocabulary German broadcast speech recognition system over a long period of time in this thesis, where we investigate and evaluate different state-of-the-art speech recognition algorithms for their employment for German broadcast speech in a productive audio mining system. We also extend the training corpus by a large quantity and evaluate the improvements.

After an automatic speech recognition system is trained, a speech recognition decoding algorithm is employed to decode the most likely text hypothesis from the speech signal. Speech recognition decoder typically have a large set of hyperparameters, which are commonly left to default values or which are manually set. These parameter values are most often far from the optimum value in terms of accuracy and decoding time. Automatic decoder parameter optimisation algorithms approach this issue, however state-of-the-art algorithms tend to need a large amount of training iterations for the training to converge. In this thesis we approach the issues related to speech recognition parameter optimisation by introducing a parameter optimisation algorithm that has never been used in the context of speech recognition before to ASR decoder parameter optimisation in the German broadcast domain. We investigate and evaluate its use for both unconstrained and constraint optimisation and compare the results to state-of-the-art methods.

1.3 Dialects in Speech Recognition

Germany has a large variety of different dialects. Dialectal speakers are often present in broadcast media, especially in regional programmes, and can cause impaired performance of the audio mining and speech recognition systems due to the phonological, semantical and syntactical differences that appear in dialectal speech compared to the standard language. One way to cope with dialects in speech recognition is to apply a dialect identification system beforehand and then to use specialised dialectal speech recognition models to decode the text. This is why in this thesis we approach the dialectal robustness of German broadcast speech recognition system. However, the way to write down dialectal text is most often not standardised and hence, transcribed dialectal speech resources are especially rare. That is why in this work a close cooperation with regional broadcasters is built up to sight dialectal resources in their archives which are then exploited to build a German dialect identification system and to improve the speech recognition system.

1.4 About This Thesis

In this thesis, we discuss the long-term development and optimisation of a German broadcast speech recognition system, which is part of a productive audiomining system, namely the Fraunhofer IAIS audio mining system. We evaluate a large number of state-of-the-art speech recognition architectures which became available in the course of this thesis for the employment in the German broadcast domain. Furthermore, we efficiently optimise the parameters of the speech recognition decoder, which is part of the speech recognition system, both in the unconstrained and in a constrained setting, with proper evaluation. We also approach the dialectal robustness of the German speech recognition system, with the help of a close cooperation to regional broadcasters, by the collection of a dialect database and the creation of a dialect identification system and the use of subsequent dialectal speech recognition models.

This thesis is structured as follows: Chapter 2 concisely summarises the scientific goals that are pursued in this work. Chapter 3 introduces the basics of speech processing, machine learning, speech recognition, and dialect identification. The main chapters of this work address the above mentioned goals: the long-term development and optimisation of the German broadcast speech recognition system including the creation and exploitation of a large German broadcast speech database is discussed in Chapter 4. The fast and efficient speech recognition decoder parameter optimisation approach for both constrained and unconstrained optimisation is described in Chapter 5. The issue of dialectal robustness in German speech recognition is dealt with in Chapter 6.

A conclusion and a summary of the scientific achievements of this thesis are given in Chapter 7.

Chapter 2

Scientific Goals

In this chapter, we discuss the topics which will be covered in this work and specify the scientific goals of this thesis.

2.1 Goals

The following scientific goals were defined at the beginning and adjusted in the course of the work:

Related to the long-term development of the German broadcast speech recognition system:

- investigate and evaluate state-of-the-art speech recognition systems in the context of German broadcast speech
- investigate the algorithms for their applicability in a productive audio mining system
- extend the amount of training data and exploit the data for training the speech recognition system

Related to the automatic speech recognition decoder parameter optimisation:

- apply and adapt methods for fast and efficient decoder parameter optimisation in the context of German broadcast speech
- extend the algorithm for the usage in an constrained setting when decoding time is an issue as it is in a productive system

Related to dialectal robustness of the speech recognition system:

- sight and prepare resources in cooperation with regional broadcasters to facilitate the improvements
- deal with the manifold of dialects in German broadcast speech

Chapter 3

Preliminaries

In this chapter the fundamentals needed to comprehend the techniques discovered and developed in this thesis are described. In Section 3.1, a short introduction to human speech is presented, including the human speech production system in Section 3.1.1 and the human speech perception system in Section 3.1.2. The chapter then advances with the transition from the physical domain to the digital domain and a short introduction to digital signal processing in Section 3.2 and a short introduction to pattern recognition in Section 3.3. After that, the chapter then advances with an introduction and the state-of-the-art to the most important techniques covered in this thesis, namely automatic speech recognition (Section 3.4) and dialect recognition (Section 3.5).

3.1 Speech

Speech is the most important means of human communication. In speech, information is encoded by the vocalisation of a syntactic combination of words derived from a vocabulary that is very large (usually more than thousand words). Each vocalised word is build from a combination of a limited set of phonemes. Phonemes are the smallest units of speech and can be divided into vowel and consonant phonemes. A language is then made of a vocabulary, a set of phonemes and the word ordering (i.e. the syntax or grammar). In written language on the other hand the text is usually made of a set of graphemes (i.e. the smallest unit of text) and again a vocabulary and the syntax. Graphemes can also be divided into vowel and consonant graphemes for languages like English or German.

3.1.1 Speech Production

Speech production is the process of the translation of thoughts into speech. After the selection of the words to be uttered, the vocal apparatus is activated. By taking a breath supported by the diaphragm muscle, air pressure from the lungs is built up and

then released. Air is flowing through the larynx, or more precisely, through the glottis, which is the interspace between the vocal cords. The airflow causes an excitation of the vocal cords. The excitation signal of the glottis can be described as an impulse chain, in case the vocal chords are vibrating (voiced excitation) or as a band-filtered noise if the vocal chords are not moving (unvoiced excitation). The frequency of the occurrences of the impulses is often referred to as the fundamental frequency f_0 or pitch. The fundamental frequency is typically lower for male speakers and higher for female speakers. Finally the excitation signal is shaped by the articulators, i.e. nose, mouth, lips and tongue. Depending on the position of the articulators, different sounds are produced. Words are usually pronounced by shaping the excitation signal by a sequence of different articulator positions. When the pronounced words exit the speech production system, the information is propagating as longitudinal air pressure waves through the air with the speed of sound (343 m/s at 20° Celsius air temperature). The organs involved in the task of speech production are depicted in Figure 3.1.

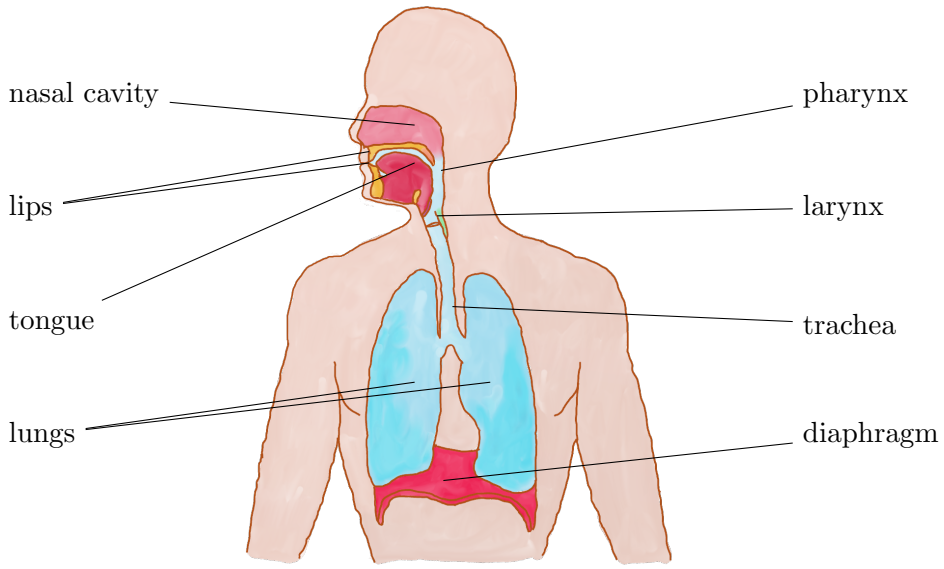


Figure 3.1: The speech production system

3.1.2 Speech Perception

Sound waves propagate through the air as fluctuations of air pressure and enter the outer ear of the human. The sound travels through the auditory channel to the ear drum, which separates the outer ear from the middle ear. The movements of the ear drum travel along the auditory ossicles (the malleus, incus and stapes) in the middle ear to the oval window in the cochlea. The oval window separates the middle ear from the inner ear. The cochlea is filled with fluid and is a spiral-shaped organ. Along

the spiral lies the basilar membrane and the organ of Corti on which sensory hair cells are situated. When the oval window is moving through the activation of the middle ear, the waves propagate in the fluid of the cochlea. Because of the movements the sensory hair cells are activated and send nerve impulses to the brain. In the cochlea a frequency transformation is happening due to the shape of the cochlea. High frequencies activate sensory hair cells near the oval window and low frequencies activate hair cells near the apex (the top of the cochlea). Finally the nerve impulses from the sensory hair cells are turned into a perception of sound in the brain. Healthy humans are able to perceive sounds in a frequency range of about 20 Hz to 20 kHz. However the frequency range is decaying in the upper bound with about 1 kHz per decade of years. The organs involved in speech perception are depicted in Figure 3.2.

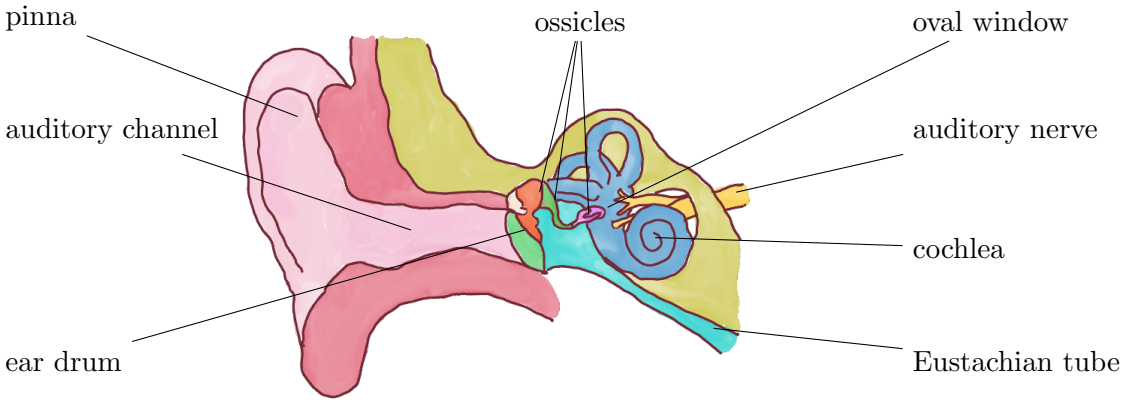


Figure 3.2: The speech perception system

3.2 Digital Signal Processing

To perform digital signal processing with speech or other audio signals on a computer, the sounds (i.e. fluctuations of air pressure) have to be captured first and then converted into the digital domain. To perform this, a microphone and a soundcard is required. The simplest form of a microphone is a dynamic microphone. A dynamical microphone is built by a diaphragm with a coil attached and a magnet. When the air pressure is changing due to incoming air waves, the diaphragm with the attached coil is moving in the magnetic field of the magnet. Due to the electromagnetic principle a current is generated in the wire of the coil, which is the analogous transformed audio signal. The same principle in the opposite direction is used for the transformation of an electrical signal into sound. In the case of a dynamic loudspeaker, the electrical signal makes the coil with the attached diaphragm move back and forth in the magnetic field of the magnet and causes the air pressure to fluctuate. Now that the

sound is converted into an electric current or respectively in an electric voltage, an analog-to-digital converter (ADC) is used to transform the analog signal into the digital domain. The ADC discretises and quantises a time-continuous signal into individual discrete samples at a given sample rate F_s. This means that every $T_s = 1/F_s$ seconds the analog signal is measured and then quantised within a discrete range of values determined by the bit depth. Audio signals are typically recorded at 8-,16- and 24-bit bit depth. However for calculations in the digital domain the signals are usually converted into 32-bit float values or 64-bit double values to have a higher precision for the calculations. A continuous (analog) signal and its digitally transformation is depicted in Figure 3.3.

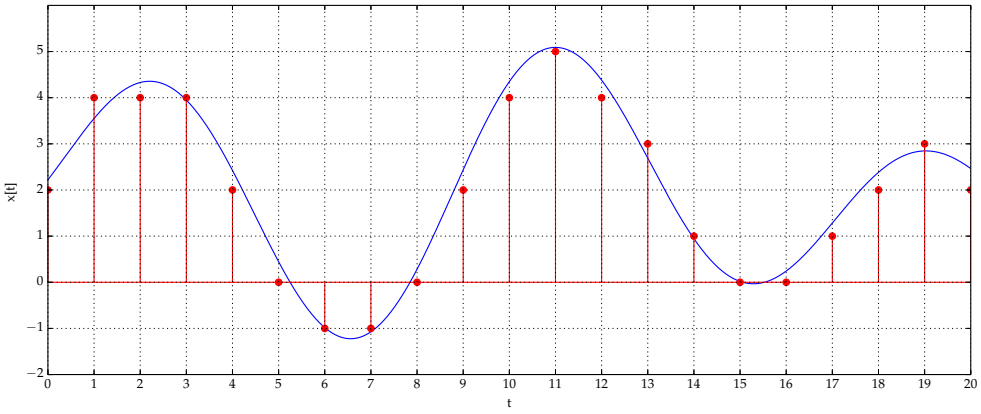


Figure 3.3: Digital signal (red) after sampling and quantising of an analog signal (blue)

It is worth noting that according to the sampling theorem of Nyquist-Shannon, the sampling rate F_s has to be twice the highest frequency of the input signal. Otherwise an effect called aliasing (i.e. undersampling) occurs, which mirrors frequencies higher than $f = F_s/2 + \Delta$ to $f = F_s/2 - \Delta$. That is why a lowpass filter ($f_{high} = F_s/2$) is necessary to prevent frequency components higher than $F_s/2$. However modern sound cards perform this lowpass filtering automatically and the user does not have to deal with aliasing. It is also worth noting that due to the quantisation of an analog signal a quantisation error occurs due to the mismatch between the analog and the quantised signal sample. This error signal is also referred to as quantisation noise and is dependent of the bit depth of the quantised signal. However this quantisation noise is usually neglectable for common bit depths used in audio signal processing. After analog to digital conversion, the discrete signal is ready to either be processed directly or stored. However many computations on digital signals do not take place in the time domain, but rather in the frequency domain.

3.2.1 Discrete Fourier transform

One of the most fundamental transforms in digital signal processing is the discrete Fourier transform (DFT). The DFT transforms a sequence of N complex numbers $x_0, x_1, ..., x_{N-1}$ into an N-periodic sequence of complex numbers:

$$X_k \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi ikn/N}, k \in \mathbb{Z} \tag{3.1}$$

Due to the periodicity attribute, the DFT is usually just computed for k in the interval $[0, N-1]$. Applying the transform to (real-valued) time domain data (e.g. audio signals, speech signals) the transform is also often referred to as discrete time Fourier transform (DTFT). The signal $x[n]$ is transformed into a complex valued spectrum X_k. The parameter k then refers to the so-called frequency bin. From the complex-valued spectrum X_k the magnitude spectrum $|X_k|$ can be derived for each bin k by:

$$|X_k| = \sqrt{\mathrm{Re}(X_k)^2 + \mathrm{Im}(X_k)^2} \tag{3.2}$$

The argument (or phase) of the complex-valued spectrum X_k can be derived by:

$$\arg(X_k) = \arctan\left(\frac{\mathrm{Re}(X_k)}{\mathrm{Im}(X_k)}\right) \tag{3.3}$$

From the complex-valued spectrum (or the magnitude spectrum and the phase) the time domain signal can be perfectly reconstructed by the inverse DFT (IDFT):

$$x_n = \frac{1}{n} \sum_{k=0}^{N-1} X_k \cdot e^{2\pi ikn/N}, n \in \mathbb{Z} \tag{3.4}$$

The fast Fourier transform (FFT) computes the DFT or its inverse. However it reduces the complexity of the algorithm from $O(N^2)$ to $O(N\log N)$ and is able to speed up calculations especially for large N. The most commonly used FFT algorithm is the Cooley-Tukey algorithm [4].

When considering long signals the frequency resolution of the DFT is getting high, however the time resolution is low. In fact the time resolution is just one spectrum for the whole signal length. This is why usually the signal is truncated in overlapping and short frames. The length of a frame is called frame size and the number of samples that overlap between subsequent frames is called overlap. The number of samples that advance between subsequent frames is called hopsize. If the DFT is calculated for each signal frame, this transformation is often called short-time Fourier transform (STFT). The frequency resolution is determined by the frame length, while the time resolution is determined by the hop size. Usually a windowing function is applied to the signal frames (typically a Hann window) before calculating the DFT to minimise effects incurred by discontinuities regarding the periodicity assumption at the frame boundaries. The magnitude spectrum of the STFT of a speech signal is depicted in Figure 3.4.

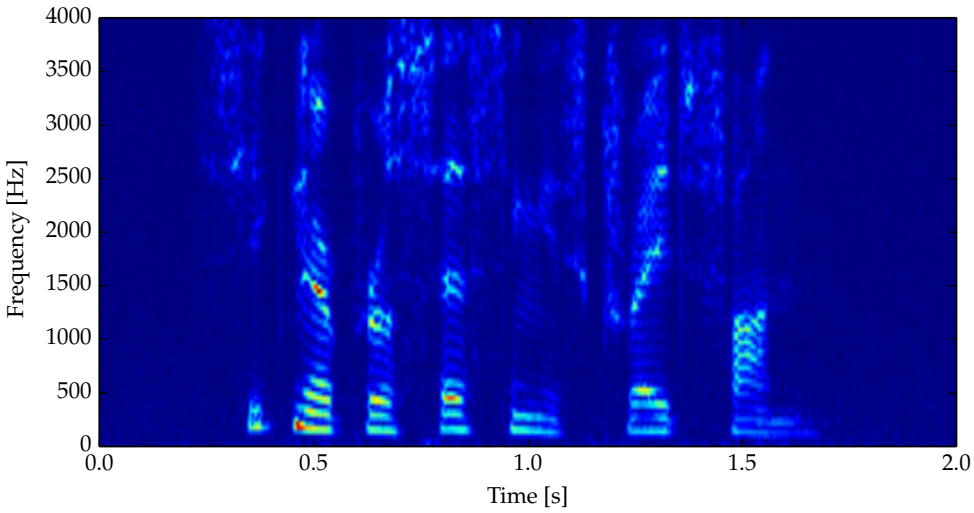


Figure 3.4: STFT (magnitude spectrum) of the speech signal "Signal processing is great fun!"

3.3 Pattern Recognition

Pattern recognition algorithms try to find regularities in data. Typically a class label or a class membership score is assigned to observations. In order for a model with free model parameters to provide accurate results, the model has to be trained on a set of training data. In supervised training this model is trained by an algorithm that learns patterns from labelled training data. Depending on the task, the model can either be a classifier in classification problems (i.e. the task of assigning a class label to an observation, e.g. recognising digits) or a regression classifier (i.e. the output can take continuous values, e.g. predicting house prices from a set of attributes). After training, the model is able to make a prediction on unseen data. Common examples of pattern recognition in speech processing are automatic speech recognition (i.e. the translation of spoken language into text), speaker recognition or language identification. To train an automatic speech recognition system, a training set containing speech signals and the underlying text is needed.

3.3.1 Feature Extraction

Learning patterns from the raw data directly is often difficult. This is why usually discriminative features are extracted before training the model. What constitutes a discriminative feature depends on the task. Features that are discriminative for one task might not be discriminative for another. However nowadays, with the increase of computational power and the availability of deep learning algorithms, pattern recognition systems try to learn discriminative features automatically and to avoid exhaustive feature engineering. This is why nowadays often high-dimensional filterbank features

are often preferred compared to low-dimensional Mel-Frequency Cepstral Coefficients, which were the preferred audio features for decades in speech recognition. Both feature types are explained in the following.

Mel-Frequency Cepstral Coefficients

In speech processing amongst the most prominent features are the Mel-Frequency Cepstral Coefficients (MFCC). To derive the MFCCs of an audio signal, the signal is first filtered with a preemphasis filter. The preemphasis filter boosts the high frequencies of the signal and is implemented by:

$$y_t = x_t - \alpha x_{t-1}, \tag{3.5}$$

where $\alpha = 0.97$. Then the signal is usually first split into frames (typically 25 ms, 10 ms hop size) and a Hamming window is applied to the signal frames. The Hamming window is defined as:

$$w_{hamming}(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \tag{3.6}$$

The MFCCs are calculated for each frame and then stacked to a matrix. To calculate the MFCCs for a signal frame, the DTFT is calculated and the magnitude spectrum is obtained. The phase is usually neglected. Then the power spectrum is calculated from the magnitude response:

$$S_{xx}(k) = |X_k|^2 \tag{3.7}$$

The powers of the spectrum are then mapped onto the mel scale using a set of l triangular overlapping windows (typically $l = 23$ for 16 kHz sampling frequency and $l = 15$ for 8 kHz sampling frequency). The mel scale [5] is a perceptual scale of pitches of equal distances. A commonly used formula [6] to convert the frequency f into mel m is:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.8}$$

A filterbank with triangular filters set in equal distance along the mel scale is depicted in Figure 3.5.

Then the logarithm is calculated on each of the obtained mel powers and the discrete cosine transform (DCT) is performed upon them to decorrelate the data and to retrieve the MFCCs. Optionally, the coefficients c_i are processed with a cepstral lifter according to:

$$\hat{c_i} = \left(1 + \frac{L}{2} \cdot \sin(\pi \frac{i}{L})\right) c_i, \tag{3.9}$$

where c_i is the MFCC at index i, L is the liftering factor (usually 22). The intention of cepstral liftering is to scale the MFCCs so they have a similar range of values.

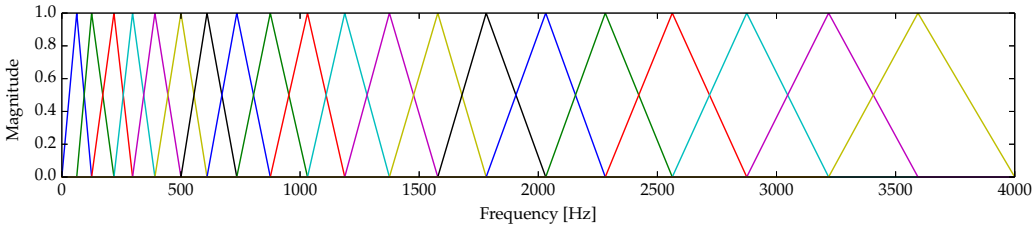


Figure 3.5: Mel filterbank with triangular filters

Typically only the first 13 coefficients are kept, since they contain the most information of the underlying signal spectrum. The other coefficients are discarded because they only contain little information about the spectrum. That is why MFCCs can be considered a compact representation of the spectrum. To cover temporal variations between subsequent frames dynamic features (e.g. delta and delta-delta coefficients) are often calculated on the MFCCs and then stacked. The authors in [7] showed that the calculation of the temporal derivations of the feature vectors have a positive influence on the recognition accuracy. The delta coefficients Δ are calculated by:

$$\Delta x(t) = x(t) - x(t-3) \tag{3.10}$$

The delta-delta coefficients $\Delta\Delta$ are calculated by:

$$\Delta\Delta x(t) = x(t) - \Delta x(t-3) = x(t) - 2x(t-3) + x(t-6) \tag{3.11}$$

In Figure 3.6 the MFCCs of a speech signal is depicted (without delta coefficients).

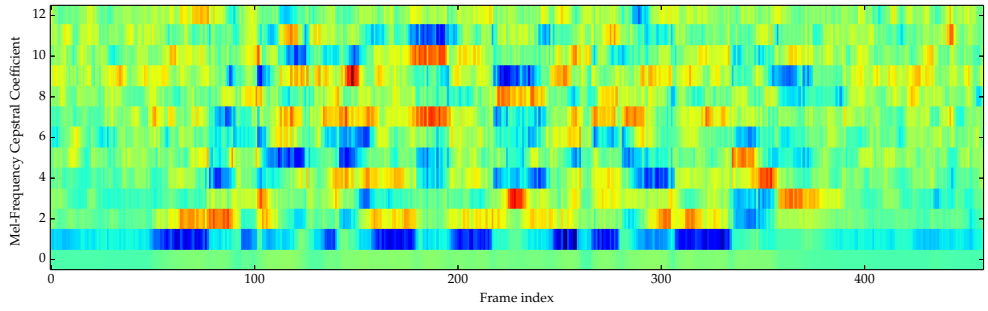


Figure 3.6: Mel-frequency cepstral coefficients of the speech signal “Signal processing is great fun!” (c.f. Figure 3.4)

Filterbank Coefficients

Filterbank coefficients are, similar to MFCCs, spectral descriptors of the audio signal. Filterbank coefficients are typically derived by calculating the Mel-filterbank coefficients

(Section 3.3.1) without the subsequent usage of the DCT. Taking the logarithm after the Mel-filterbank coefficients' calculation, is optional. An advantage compared to the MFCCs is, that no coefficients are discarded. Typically 23 filters are used for 16 kHz sampling rate and 15 filters for 8 kHz sampling rate.

3.3.2 Hidden Markov Models

Hidden Markov models (HMM) are often successfully used in temporal pattern recognition problems, e.g. speech recognition, handwriting recognition or gesture recognition, where the information can be modelled as a temporal sequence of states (e.g. phonemes, graphemes, gestures or subdivisions of those). HMMs were developed in [8]. In HMMs, only the outputs, i.e the observations, are directly visible, the states on the other hand are not directly visible, that is why they are called hidden. An HMM typically consists of a set of hidden states $S = \{s_1, ..., s_n\}$, a set of of possible observations $Y = \{y_1, ..., y_m\}$, the state transition matrix $A \in \mathbb{R}^{n \times n}$, the emission probability matrix $B \in \mathbb{R}^{n \times m}$ and the initial state distribution $\pi \in \mathbb{R}^n$. Stationary HMMs are HMM where the state transition probabilities A and the emission probabilities B do not change over time, an assumption that often holds true. In Figure 3.7 an exemplary HMM is depicted. Only adjacent states can be reached from a specific state. Also the state can remain the same. Training of the HMM is usually performed by the expectation maximisation (EM) algorithm [9]. Hidden Markov models are used in acoustic modelling for speech recognition in Section 3.4.4.

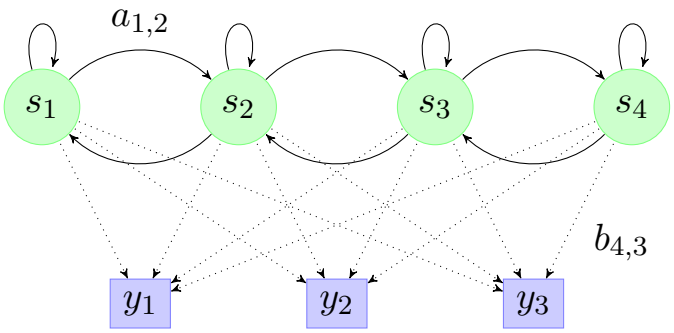


Figure 3.7: An HMM with 4 states. It can emit 3 discrete symbols y_1, y_2 or y_3. $a_{i,j}$ is the probability to transition from state s_i to state s_j. $b_{j,k}$ is the probability to emit symbol y_k in state s_j. In this exemplary HMM, states can only reach the adjacent states or themselves.

3.3.3 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a probabilistic model which assumes that the observations $D = \{x_1, ..., x_i, ..., x_N\}$ are generated from an underlying probability den-

sity $p(x)$. This density $p(x)$ is defined as a linear combination of a finite number of weighted Gaussian probability density functions:

$$p(x_i|\Theta) = \sum_{j=1}^{J} \omega_j \mathcal{N}(x_i|\mu_j, \Sigma_j) \tag{3.12}$$

where x_i is the observation at index i, j is the component index, J is the total number of components, ω_j is the weight, μ_j is the mean vector, Σ_j is the covariance matrix of component j respectively, and $\mathcal{N}$ is the Gaussian probability density function. $\Theta = \{\omega_j, \mu_j, \Sigma_j\}_{\forall j}$ are the model parameters of the GMM. The weights ω_j of a GMM represent probabilities with $0 \leq \omega_j \leq 1$, $\sum_{j=1}^{J} \omega_j = 1$. The univariate (i.e. one-dimensional, $d = 1$) probability density function of the Gaussian distribution is defined as:

$$\mathcal{N}_1(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.13}$$

where μ is the mean and σ is the standard deviation with its variance σ^2.

In Figure 3.8 an exemplary GMM composed of $J = 3$ univariate Gaussian distributions with different means and standard deviations is depicted. The solid red curve shows the probability function of the GMM, the dashed curves show the probability density functions of the components of the GMM.

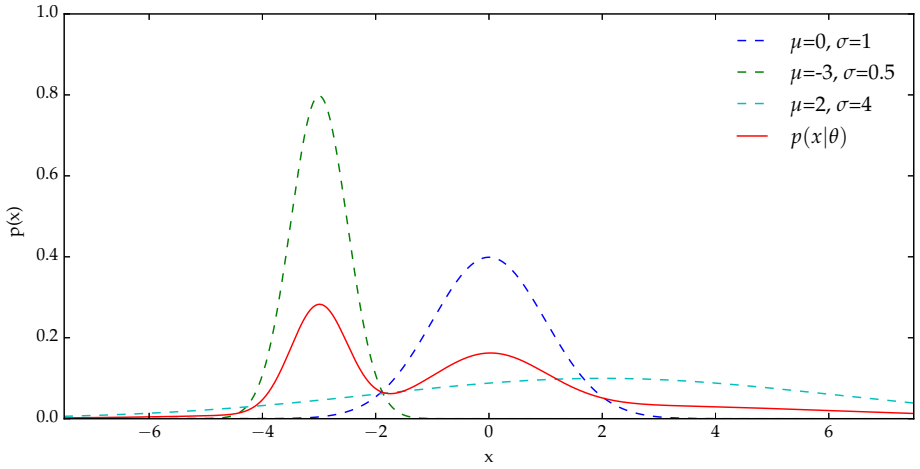


Figure 3.8: Example of a univariate GMM (solid curve) composed of three components with different means and standard deviations (dashed curves)

The multivariate probability density function $\mathcal{N}_d(x|\mu, \Sigma)$ of the d-dimensional Gaus-

sian distribution is calculated as:

$$\mathcal{N}_d(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-\frac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)} \tag{3.14}$$

where $|\Sigma|$ is the determinant of the covariance matrix Σ.

The fitting of a Gaussian Mixture Model to a set of training points is usually done with the expectation-maximisation algorithm [8].

After the fit, the component membership of a data point, i.e. the probability of a data point x being from component k, can be obtained by:

$$p(k|x) = \frac{w_k \mathcal{N}(x|\mu_k,\Sigma_k)}{\sum_{j=1}^{J} w_j \mathcal{N}(x|\mu_j,\Sigma_j)} \tag{3.15}$$

and the component label of a datapoint, i.e. the component which maximises the component membership for a data point, by:

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, p(k|x) \tag{3.16}$$

Multiple GMM can be trained for data sets containing multiple classes. After training, the class label can be obtained for unknown data points by:

$$\hat{c} = \underset{c}{\operatorname{argmax}}\, p(x|\Theta_c) \tag{3.17}$$

3.3.4 Artificial Neural Networks

An artificial neural network (ANN) is based on a large collection of artificial neurons that are transforming an input to an output. The approach is motivated by modelling the biological brain which solves problems with a large number of biological neurons that are connected by axons. A common architecture of an artificial neuron is depicted in Figure 3.9.

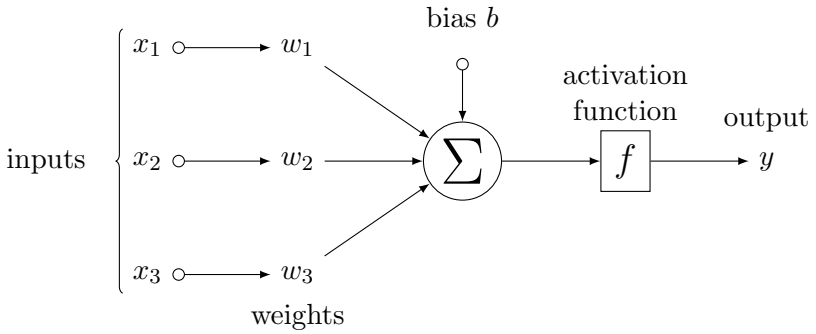


Figure 3.9: Artificial neuron with three inputs

The output of the artificial neuron is derived by:

$$y = f(b + \sum_{i=1}^{N} x_i w_i) \tag{3.18}$$

where N is the number of inputs, w are the weights, x are the inputs, b is the bias and f is the activation function. There are different activation functions used in the literature. The linear function is simply $f(z) = z$. The sigmoid activation function is calculated as:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3.19}$$

A special case of activation function is the softmax function, which is typically used in output layers, as they represent a probability distribution and are dependent on the outputs of the other neurons in the layer. It is calculated as:

$$f(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{3.20}$$

and ensures that $0 \leq y_m \leq 1$ and $\sum_k y_k = 1$. The most prominent activation functions are depicted in Figure 3.10.

Artificial neurons are typically arranged in layers to form an artificial neural network. An ANN with two hidden layers is depicted in Figure 3.11. ANNs with one hidden layer, also called multilayer perceptron, were the prominent type of ANNs in the 90s. However, due to the availability of more computational power and more sophisticated training and initialisation algorithms, it is nowadays feasible to train ANNs with multiple hidden layers. ANNs with multiple hidden layers are also called deep neural networks (DNN). The training of ANNs is typically performed by gradient descent or derivatives of the algorithm to minimise the cost between the network's output and the training target values. The training algorithm also requires a cost function, i.e. a measure that determines the mismatch between the output and the training target. A common cost function is the mean-squared error, which minimises the average squared error. However, there are many more cost functions available that have been proven to be successful in different situations. The selection of the network architecture, the size of the network, the number of hidden layers, the activation function for each layer, the training algorithm and the cost function depend on the problem and on the size of the available training data. Artificial neural networks have been successfully employed in almost every discipline of pattern recognition and have become very popular in the last few years due to the advances in training networks with a large number of hidden layers.

Recurrent Neural Networks

While traditional neural networks assume that all inputs and outputs are independent of each other, recurrent neural networks (RNN) are intended to exploit the sequential

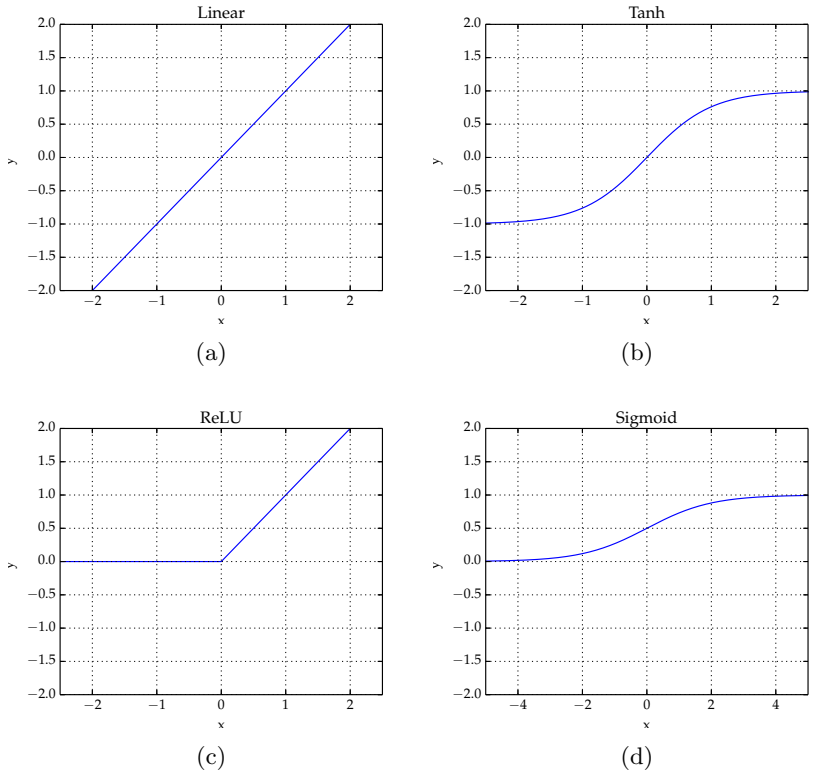


Figure 3.10: Artificial neural network activation functions; a) linear function; b) tangens hyperbolicus function; c) rectified linear unit function; d) sigmoidal function

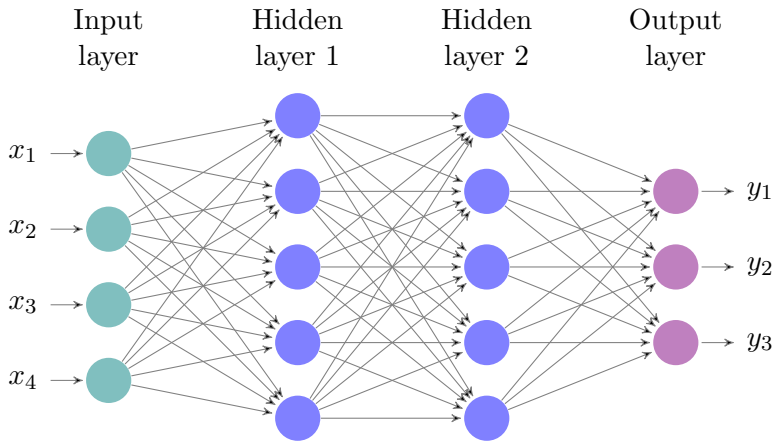


Figure 3.11: An artificial neural network with two hidden layers. It takes four input values and maps the inputs to three output values by employing two hidden layers consisting of 5 neurons each.

information of the data. In recurrent neural networks the output is dependent not only on the input, but also on the previous computations. In Figure 3.12 a diagram of an RNN is depicted and its unfolded equivalent, where x_t is the input at time t, s_t is the hidden state at time t, and y_t is the output at time t.

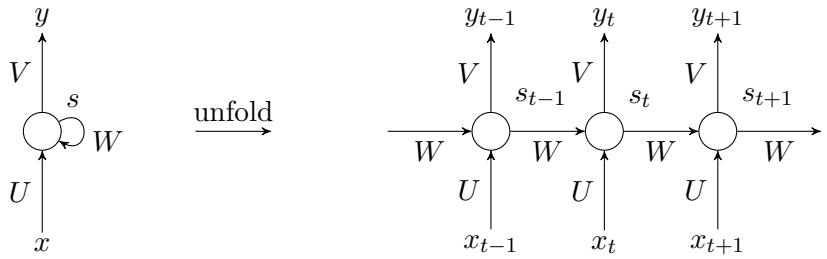


Figure 3.12: A recurrent neural network and the computation's unfolding over time

s_t is calculated based on the previous hidden state s_{t-1} and the current input x_t according to:

$$s_t = f_s(Ux_t + Ws_{t-1}), \tag{3.21}$$

where f_s is the activation function in the recurrent hidden layer (e.g. tanh, ReLU). The output o_t is calculated by:

$$o_t = f_o(Vs_t), \tag{3.22}$$

where f_o is the activation function of the output layer (often a softmax function). An RNN shares the same parameters (U,V,W) across time t. The principal feature of an RNN is the hidden state s, which is able to capture sequential information. The most commonly used type of RNNs are long short-term memory (LSTM), which are an improved and more sophisticated version of the classical (i.e. the so-called vanilla) RNNs. RNN have been employed with great success in natural language processing (NLP) tasks like language modelling, machine translation and speech recognition [10], by exploiting the sequential nature of speech and language.

Convolutional Neural Networks

Convolutional neural networks (CNN) have been used since the 1990s [11] and have been proven to be very effective in areas such as image recognition and classification [12]. They also have been successfully employed to speech tasks like speech recognition [13] and language identification [14, 15] in the last few years. A typical CNN architecture is depicted in Figure 3.13. Convolutional neural networks typically consist of multiple convolutional layers and fully connected layers at the end. The convolutional layers consists of a set of convolutional filters or kernels. These are trained during the training procedure and act like feature detectors. Each convolutional neuron only processes

data for its receptive field, which is limited by the size of the kernels. Convolutional layers apply a convolution operation to the input and then pass the result to the next layer. Usually a pooling layer is added after each convolutional layer, where the dimensionality of the input is reduced by subsampling (e.g. by taking the maximum value or the sum of the inputs). The convolutional neural network can be trained e.g. by the backpropagation algorithm.

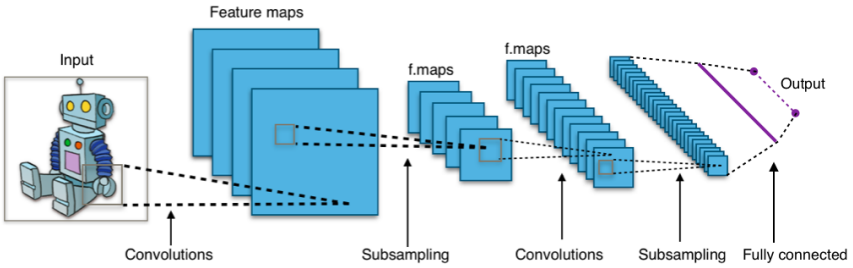


Figure 3.13: Typical CNN architecture [16]

3.4 Automatic Speech Recognition

Automatic speech recognition (ASR), or sometimes called "speech to text" (STT), is the translation of spoken language into text by computers. Speech recognition tasks include tasks with a limited vocabulary and grammar (e.g. the recognition of a limited set of commands, words or numbers) and the recognition of large-scale vocabulary continuous speech (LVCSR). ASR systems can be speaker-dependent (e.g. if trained or fine-tuned to a specific speaker) or speaker independent (e.g. if trained on a large set of speakers). Nowadays LVCSR systems consist of the acoustic model (i.e. modelling the probabilities of phonemes given acoustic features derived from the speech signal), the dictionary (i.e. a lexicon which maps the words to a sequence of phonemes) and a statistical language model (i.e. a probability distribution over sequences of words). During decoding, the ASR system is able to provide the most probable word sequence encoded in the speech signal given the model.

3.4.1 History of Automatic Speech Recognition

A diagram covering the milestones of the history of automatic speech recognition is depicted in Figure 3.14. It is an attempt to summarise and update the diagram in [17]. Early speech recognition systems in the 1950s only considered a vocabulary consisting of a few words or digits. For example, an early automatic speech recognition system was proposed by Davis, Biddulph, and Balashek of Bell Laboratories in 1952 [18]. The system measured formant frequencies (i.e. regions of energy concentration in the speech power spectrum) during vowels regions of digits for single-speaker isolated digit recognition. Formant trajectories of the first and second formant frequencies derived

from labelled digit utterances served as reference pattern for classifying an unknown digit. In 1956 Olson and Belar of RCA Laboratories created a recogniser to identify ten syllables of a single speaker [19]. In 1959 Fry and Denes [20] used statistical information about the underlying language to improve the performance of their phoneme recogniser.

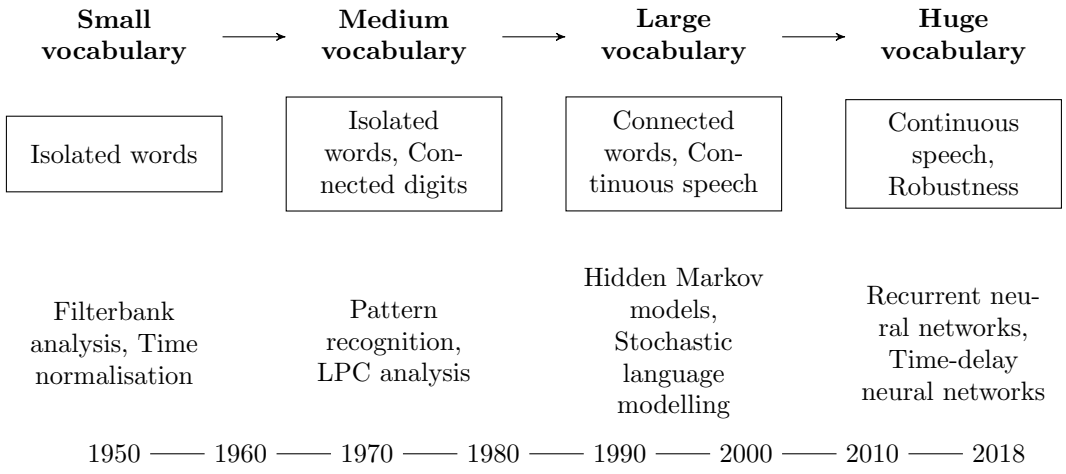


Figure 3.14: Milestones in speech recognition

The number of isolated words in a vocabulary increased up to approximately 100 in the 1960s. In this decade time-normalisation techniques [21] and dynamic programming methods e.g. Dynamic Time Warping (DTW) [22] were introduced inter alia to address the problem of variations in the speaking rate (i.e. temporal variations in repeated speech utterances).

More flexible connected digit and continuous speech systems with medium vocabulary size (up to approx. 1000 words) became feasible in the 1970s (e.g. "Harpy" [23], which introduced a concept of graph search to ASR) with the new upcoming advances in normalisation techniques, dynamic programming (e.g. the application of the Viterbi algorithm [1] for speech recognition [24]), spectral descriptors and pattern recognition. Itakura [25] and Atal [26] independently proposed the concept of Linear Predictive Coding (LPC) for effective estimation of the vocal tract response from speech signals.

Large vocabulary continuous speech recognition (LVCSR) was successfully performed in the 1980s with the advent of Hidden Markov Models (HMMs) [8], and their use in speech recognition [27, 28], and stochastic language models (LMs), most notably the n-gram models [29, 30], enabling systems with a vocabulary of far more than 1000 words. The use of Mel-Frequency Cepstral Coefficients (MFCCs) [31, 32], which were already used for speaker recognition in the 1970s, became popular in ASR. Also the use of dynamic features i.e. first and second order derivatives (also speed and acceleration coefficients) was proposed [7] to improve ASR performance remarkably. ASR and LVCSR systems based on MFCCs, mixture density HMMs and stochastic LMs with some extensions and improvements are still considered to be state-of-the-art speech recognisers.

LVCSR systems were trained on large amounts of training data in the 1990s. It is worth remarking that with increasing computational power and more flexible algorithms, better acoustic and language models for LVCSR could be trained with an increased amount of acoustical and textual training data. While those systems were performing well in clean and controlled conditions, ASR performance dropped significantly in difficult conditions, often rendering them useless for a certain tasks. Additionally ASR systems are typically not robust to mismatched training and test conditions. It is no wonder that much effort was made in the last few decades to improve the robustness of ASR systems. Problems to be tackled in the context of robustness are amongst others background noise, reverberation, channel distortions, casual and disfluent speech, speaker variabilities and mismatched training and testing conditions.

Important early approaches to improve robustness of ASR systems are maximum likelihood linear regression (MLLR) [33] and maximum a posteriori (MAP) [34] adaption. Many other approaches in the context of robust speech recognition have been proposed since then, including novel features such as perceptive linear prediction (PLP) coefficients [35], noise robust features [36] and discriminative features [37], missing feature approaches [38, 39, 40], discriminative training (e.g. maximum mutual information (MMI) estimation [41], minimum classification error (MCE) [42, 43], and minimum phone error (MPE) [44]) and speaker and noise adaptation. Also system combination approaches and advances in language modelling and hypothesis search brought ASR systems to a higher level of maturity.

Since the 1980s mixture density HMMs have been the quasi standard in acoustic modelling. However, in the last few years Deep Neural Network (DNN) acoustic models became relevant [45] due to increased computational power and efficient pretraining algorithms [46] and are reported to outperform discriminatively trained mixture density HMMs [47]. In the last few years, end-to-end speech recognition using recurrent neural networks [48] has been proposed which have a number of significant improvements compared to DNN approaches.

While remarkable improvements could be achieved, the problem of robust speech recognition in difficult conditions is still far from being solved. This section gave a short overview of the history of automatic speech recognition and its most important milestones. An exhaustive summary of the history of ASR can be found in [17].

3.4.2 Statistical Speech Recognition

Today's speech recognition systems are based on statistical approaches. The aim of the statistical approach in automatic speech recognition is to find the sequence of words $w_1^N = x_1, ..., x_n$, that maximises the posterior probability given a sequence of acoustic features $x_1^T = x_1, ..., x_t$. Statistical speech recognition systems are based on the Bayes' theorem [49]. The following equation states the Bayes' theorem mathematically:

$$p(a|b) = \frac{p(b|a) \cdot p(a)}{p(b)}, \tag{3.23}$$

where a and b are events and $p(b) \neq 0$.

In the context of speech recognition the Bayes' theorem can be applied as follows to calculate the probability of a word sequence given the acoustical observations:

$$p(w_1^N|x_1^T) = \frac{p(x_1^T|w_1^N) \cdot p(w_1^N)}{p(x_1^T)} \tag{3.24}$$

To calculate the most probable word sequence on a given acoustical observation, this equation turns into:

$$\begin{aligned} \left[w_1^N\right]_{opt} &= \underset{w_1^N}{\operatorname{argmax}}\{p(w_1^N|x_1^T)\} \\ &= \underset{w_1^N}{\operatorname{argmax}}\{p(x_1^T|w_1^N) \cdot p(w_1^N)\} \end{aligned} \tag{3.25}$$

Note that the probability of the acoustical observations $p(x_1^T)$ vanishes in the argmax calculation, because the value is a constant and does not change for any possible word sequence.

Two stochastic models occur in Equation 3.25, namely the language model (LM, expressed by $p(w_1^N)$) and the acoustical model (AM, expressed by $p(x_1^T|w_1^N)$). The LM assigns a prior probability $p(w_1^N)$ to a sequence of words. The AM assigns the conditional probability $p(x_1^T|w_1^N)$ of observing a sequence of acoustical features for the given word sequence.

In Figure 3.15 the general overview of a statistical speech recognition system is depicted.
A statistical speech recognition system consists of several components:

Feature extraction Acoustical feature extraction aims to extract discriminative features x_1^T from the input speech signal. Feature extraction is discussed in Section 3.3.1.

Acoustical model The acoustical model consists of statistical models for words or subword units, e.g. syllables or phonemes.

Pronunciation dictionary The pronunciation dictionary, which is considered to be part of the AM, defines a mapping between the words and the subword units.

Language model The language model models the probabilities of sentences (including the semantics and the syntax) of the considered language.

Search The search combines the models and finds the most probable word sequence given the acoustical observations according to Equation 3.25.

The components, with exception of feature extraction, which is already discussed in Section 3.3.1, are discussed in the following sections.

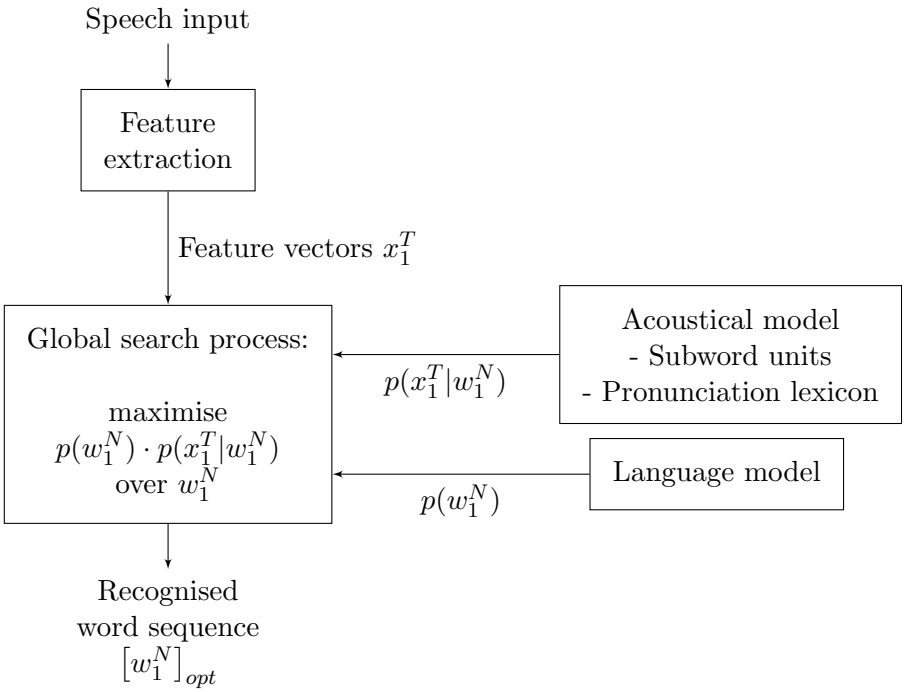


Figure 3.15: Overview of a statistical speech recognition system (c.f. [50])

3.4.3 Pronunciation Dictionary

In most cases, the AM provides models for phonemes instead of modelling whole words directly. A phoneme is the smallest unit of sound that distinguishes one word from another in a particular language. Phonemes are abstractions of phones (the actual speech sounds). Using models on subword units instead of whole words makes the training more reliable, because more training data is available for these small units compared to words, which can often been seen only a few times. Another advantage is that the vocabulary can be gracefully extended independent of the acoustic training data. Even domain changes are possible, e.g. the acoustic model of a speech recognition system trained on news data can be used for sports data by using a language model based on sports data. The words used in the different domains may differ, however the domains share the same set of subword units.

The pronunciation dictionary, which is often called lexicon, defines a mapping of the words to sequences of phonemes. It is technically possible to use multiple pronunciations for a single word. This is reasonable because words can actually have different pronunciations in a certain language. For example, the word “tomato” can be pronounced either by “T AH M EY T OW” or by “T AH M AA T OW”. The example is taken from CMU dictionary [51]. On the other hand different words can have the

same pronunciation, the so-called homophones[1]. For example, the words "cereal" and "serial" have the same pronunciation, namely "S IH R IY AH L". Homophones are only distinguishable in the context of a phrase or sentence, which is modelled by the LM.

The pronunciations of the pronunciation lexicon are either generated manually or automatically. The manual transcription of (a large quantity of) words into phonemes is costly, which is usually performed by expert linguists. For common languages typically large pronunciation dictionaries exists, e.g. CMU dictionary [51] for British English, or Phonolex [52] for German.

Grapheme-to-Phoneme Conversion

From these dictionaries, automatic pronunciation generation models can be trained, the so-called grapheme-to-phoneme (G2P) conversion models, which are then able to offer pronunciations for seen and unseen words with high accuracies. These approaches include rule-based and statistical approaches [53]. A grapheme is the smallest textual unit of a writing system of any given language. Graphemes include alphabetic letters, numerical digits, punctuations and other individual symbols. A grapheme may or may not correspond to a single phoneme of the spoken language. Sometimes, when no pronunciation lexicon is available, the speech recognition system is directly trained on the grapheme sequence, which works quite well for certain languages e.g. German. The de-facto state-of-the-art algorithm over the last few years is the statistical approach presented in [53]. The algorithm is based on joint-sequence models, where the most likely pronunciation $\varphi \in \Phi^*$ for a given orthographic form $g \in G^*$ is defined by:

$$\varphi(g) = \underset{\acute{\varphi} \in \Phi^*}{\operatorname{argmax}}\, p(\acute{\varphi}, g), \tag{3.26}$$

where Φ and G are the sets of phonemes and graphemes respectively. The joint probability distribution $p(\varphi, g)$ is also referred to as the graphonemic joint sequence model. It is assumed that the pronunciation and the orthographic form of each word is composed of a sequence of graphones. Each graphone is a pair q taken from the set of graphones Q, where:

$$q = (g, \varphi) \in Q \subseteq G^* \times \Phi^*. \tag{3.27}$$

Hence, q is a pair of a phoneme sequence and a grapheme sequence, which may have different lengths. A sequence of graphones, as an example, looks like:

"phoneme" fowniym	=	ph / F	o / OW	n / N	e / IY	m / M	e / -

[1] A list of British-English homophones can be found at `http://www.singularis.ltd.uk/bifroest/misc/homophones-list.html`, accessed October 19th, 2017

The joint probability distribution $p(\varphi, g)$ can be reduced to a probability distribution over graphone sequences $p(q)$ and can be modelled by an m-gram model (Section 3.4.5):

$$p(q_1^N) = \prod_{n=1}^{N} p(q_n | q_{n-m+1}, ..., q_{n-1}) \tag{3.28}$$

The graphone size limit L is the maximum number of graphemes or phonemes per graphone allowed by the model. As shown in [53], this model can be trained using Maximum Likelihood (ML) training using the Expectation Maximisation (EM) [9] algorithm. After training the model, the pronunciation for a word, which could be unseen by the model during training, can be derived. The most likely graphone sequence matching the spelling of the word is derived, and projected onto the phonemes, by:

$$\varphi(g) = \varphi(\underset{q \in Q^* | g(q)=g}{\operatorname{argmax}} p(q)) \tag{3.29}$$

3.4.4 Acoustical Model

The acoustical model (AM) is responsible to provide stochastic models that capture both the temporal and static features of the speech signal. The models have to take variations in the acoustic realisations of the speech into account, including the variation of the speaking rate. This is achieved by hidden Markov models (HMMs) [54] in many modern ASR systems [55]. As explained in Section 3.3.2, an HMM consists of a set of hidden states, which cannot be observed directly and hence are hidden. The possible transitions between the states are defined by the topology of the HMM. In automatic speech recognition, often the Bakis topology [56] is used. In the Bakis topology, each state has a forward and a skip forward transition, and a loop, as depicted in Figure 3.16. Loop transitions allow for slow speaking rates, whereas skip forward transitions allow for fast speaking rates.

An HMM is trained for each word or sub-word unit (phoneme, triphones, senones). A triphone is a phone considered in context with its left and right phones. For example, the t-**i**-n in the word "tin" sounds a bit different than the b-**i**-n in the word "bin". A detector for a part of a triphone (e.g. the beginning, the middle, or the end) is called a senone, and may be shared across many triphones (state tying).

Each sub-word unit is represented by an HMM consisting of a set of hidden states (typically 3 to 6). Recently however, the ASR systems are trained on context-dependent (CD) triphones. These sub-word HMM can be concatenated to form word models. In a similar fashion, the word HMMs can be concatenated to form HMMs for word sequences. For a given feature sequence x_1^T the states of an HMM can be traversed in different ways, considering the possible forward, loop and skip forward transitions. The probability of observing the feature sequence x_1^T given the word sequence w_1^N is

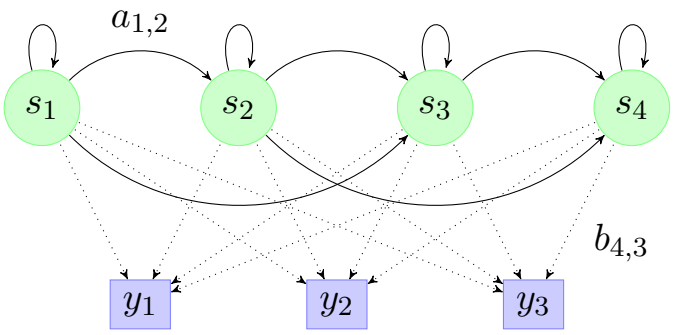


Figure 3.16: An HMM with 4 states following the Bakis topology [56]. It can emit 3 discrete symbols y_1, y_2 or y_3. $a_{i,j}$ is the probability to transition from state s_i to state s_j. $b_{j,k}$ is the probability to emit symbol y_k in state s_j. The HMM allows for loops, forward and skip forward transition.

defined as the sum over all the possible state sequences s_1^T:

$$\begin{aligned} p(x_1^T|w_1^N) &= \sum_{s_1^T} p(x_1^T, s_1^T|w_1^N) \\ &= \sum_{s_1^T} \prod_{t=1}^{T} p(x_1^T, s_1^T|x_1^{t-1}, s_1^{t-1}, w_1^N) \\ &= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|x_1^{t-1}, s_1^t, w_1^N) \cdot p(s_t|x_1^{t-1}, s_1^{t-1}, w_1^N) \end{aligned} \tag{3.30}$$

Equation 3.30 can be rewritten as:

$$p(x_1^T|w_1^N) = \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t, w_1^N) \cdot p(s_t|s_{t-1}, w_1^N) \tag{3.31}$$

by using the model assumption that $p(s_t|x_1^{t-1}, s_1^{t-1}, w_1^N)$ depends only on the identity of the direct predecessor state (first-order Markov assumption) and that the probability of observing the feature vector x_t depends only on the model state s_t.

The factor $p(x_t|s_t, w_1^N)$ of Equation 3.31 refers to the *emission probability*. It is the probability of observing the feature vector x_t while being in state s_t. The *transition probability* $p(s_t|s_{t-1}, w_1^N)$ is the probability for a transition from the state s_{t-1} to state s_t.

The sum in Equation 3.31 can be approximated by the maximum over all state

sequences, resulting in the maximum or *Viterbi* approximation [57]:

$$p(x_1^T|w_1^N) \approx \max_{s_1^T}\Big\{\prod_{t=1}^{T} p(x_t|s_t, w_1^N) \cdot p(s_t|s_{t-1}, w_1^N)\Big\} \tag{3.32}$$

The Viterbi approximation allows for an efficient search procedure. Both Equation 3.31 and Equation 3.32 can be calculated efficiently using dynamic programming algorithms [1, 2].

For decades the emission probabilities of the HMMs for speech recognition were modelled by Gaussian mixture models (Section 3.3.3). In this case the emission probability for a state s is defined by a set of Gaussian densities and corresponding weights:

$$p(x|s) = \sum_{j=1}^{J} \omega_j \mathcal{N}(x|\mu_j, \Sigma_j) \tag{3.33}$$

The speech recognition acoustical model architecture that incorporates HMM with GMM emission probabilities is referred to as GMM-HMM in this thesis. These GMM-HMMs are typically trained to maximise the likelihood (ML) of generating the observed features.

3.4.5 Language Model

The language model (LM) models the probability $p(w_1^N)$ of a word sequence w_1^N, thus covering aspects such as the semantics (i.e. the meaning of words, sentences and texts) and the syntax (i.e. the grammar) of a language. It is used with great success in automatic speech recognition. The most commonly used LM in ASR over the last decades is the m-gram model [30].

m-gram Language Models

m-gram models, also known as n-gram models, are based on the chain rule of probability theory (decomposition rule). The total probability of a word sequence can be formulated as a product of conditional probabilities:

$$P(w_1^N) = P(w_1 ... w_N) = \prod_{n=1}^{N} p(w_n|w_{n-1}, ..., w_1) \tag{3.34}$$

m-gram models only consider the $m-1$ predecessors (Markov assumption of order $m-1$). It is reasonable to restrict the size of m, because during model training with increasing m it is more unlikely to see an m-gram phrase in the training text. Moreover it is intuitive that the words (e.g. the words w_i and w_j) get increasingly independent from each other, with increased distance $(j-i)$ from each other.

With m-gram models only considering $m-1$ predecessors, Equation 3.34 gets reduced to:

$$P(w_1^N) \approx \prod_{n=1}^{N} p(w_n|w_{n-1}, ..., w_{n-m+1}) \tag{3.35}$$

For unigram, bigram, and trigram models, Equation 3.35 can be rewritten as:

Unigram ($m = 1$):

$$P(w_1^N) = \prod_{n=1}^{N} p(w_n) \tag{3.36}$$

Bigram ($m = 2$):

$$P(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-1}) \tag{3.37}$$

Trigram ($m = 3$):

$$P(w_1^N) = \prod_{n=1}^{N} p(w_n|w_{n-2}, w_{n-1}) \tag{3.38}$$

During training, the probabilities are estimated from typically a large amount of example text. But as already mentioned, with increasing m it is increasingly likely that not all possible m-gram phrases are seen during training. Unseen m-grams would obtain a probability of zero, and could not be detected. Language model smoothing algorithms, which deal with the problem of insufficient data, include Katz smoothing [58], Witten-Bell smoothing and the popular modified Kneser-Ney smoothing [59] algorithm.

The performance of a language model is either evaluated directly in the application (e.g. for speech recognition, by comparing the word error rate (WER, Section 3.4.9) of different configurations on a common test set, which is costly, or by calculation of the perplexity (Section 3.4.9) on a common test text.

3.4.6 Search

Automatic speech recognition systems take a speech data segment as input and process a list of recognition hypotheses as output. The search algorithm is responsible to efficiently retrieve the most probable word sequence $\left[w_1^N\right]_{opt}$ by evaluating the possible word sequences according to their probability derived from the language and acoustic model (Equation 3.25). Usually many of the competing hypotheses in the search procedure have common subsequences. The Viterbi approximation in Equation 3.32 can be solved efficiently by the Viterbi decoding algorithm [1]. The forward-backward or Baum-Welch algorithm [60] is a generalised approach for solving Equation 3.32 or Equation 3.31. Both algorithms are examples of dynamic programming. Pruning techniques during search can be applied to alleviate computational extensive recognition tasks. The removal of a correct hypothesis from the search space, which could occur

while pruning, is referred to as a search error. Search errors can be distinguished from model errors, which occur due to false model assumptions.

3.4.7 Decoder Parameter Optimisation

Speech decoders (e.g., HTK [61], Julius [62] or Kaldi [55]) typically provide default values for their parameters, which often appear to be set arbitrarily. The Sphinx [63] Wiki offers detailed ways how to improve the decoding speed or hypothesis quality, but these recommendations have to be transferred manually to the task, usually by a grid search. Because these strategies to select the decoder parameter values are often suboptimal, recently, the task of optimising ASR decoder parameters automatically was approached by several research groups. Some approaches optimise the word error rate (WER, see Section 3.4.9) or similar metrics directly, e.g. with evolutional algorithms [64], or by employing large-margin iterative programming [65], others also take the decoding speed, with the real-time factor (RTF, see Section 3.4.9) as a measure, into account. The authors in [66] evaluate the curve of all possible WER for a given RTF and hence, they are able to use the optimal configuration for any speed constraints. The authors in [67] use an approach very similar to the Kiefer-Wolfowitz finite-difference stochastic approximation (FDSA) [68], where the influence of each parameter on the WER and the RTF is evaluated separately for a given configuration. In the field of machine translation (MT), the parameters of recent decoders are often optimised either with Och's Minimum Error Rate Training [69] or the Downhill Simplex method [70]. Simultaneous Perturbation Stochastic Approximation (SPSA) has been employed for machine translation as well and has been shown to converge faster than Downhill simplex, while maintaining comparable results [71].

3.4.8 Weighted Finite State Transducer

A finite-state transducer (FST) is a finite automaton whose state transitions are labelled with both input and output symbols. Hence, a path through the transducer provides a mapping between an input symbol sequence and an output symbol sequence. A weighted FST (WFST) puts additional weights on its transitions from the input to the output symbols. The weights can encode probabilities, penalties, durations or any other measure. The weights get accumulated along the path to calculate the overall weight for mapping the input sequence into the output sequence. In Figure 3.17 an exemplary WFST is depicted, covering a (weighted) mapping between the word "tomato" and its multiple pronunciations "T AH M AA T O" and "T AH M EY T O".

In Figure 3.18 a WFST is depicted representing a small language model, containing three phrases. In this case the input symbols, namely words, are the same as the output symbols, however the transitions, and therefore the phrases as well, are weighted. FSTs can be optimised by determinisation and minimisation. The result of the determinisation is that no state has two transitions with the same input label. In Figure 3.17, the transition from node 4 to 5 has two transitions with the same input label, and hence the FST can be optimised. Minimisation produces the minimal FST that is equivalent

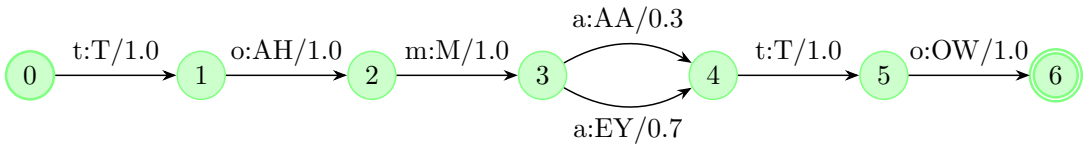


Figure 3.17: Exemplary weighted finite state transducer representing a pronunciation lexicon

to the original FST, i.e. the FST with a minimum number of states. FSTs can be cascaded using finite-state composition. And the optimal results can be derived by shortest path algorithms.

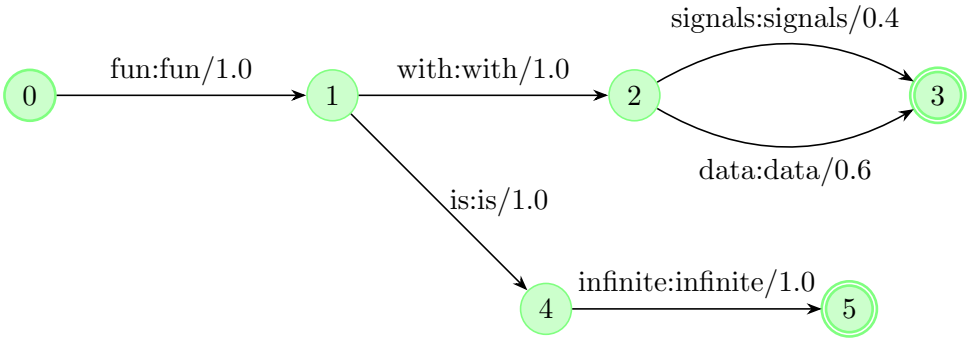


Figure 3.18: Exemplary weighted finite state transducer representing a language model

FSTs have been applied with great success in language and speech processing [72], speech recognition [55] and synthesis, optical character recognition, machine translation, and other machine learning applications. In the Kaldi [55] speech recognition toolkit the WFST based algorithms are implemented based on the OpenFst toolkit[2]. In the Kaldi toolkit, the pronunciation lexicon, the language model and the HMMs of the acoustic models are implemented as WFSTs, which are then composed into a large WFST for the ASR decoding algorithm.

3.4.9 Evaluation and Performance Measures

This section explains the evaluation and performance measures related to automatic speech recognition. The word error rate (WER), the phoneme error rate (PER), the real-time factor (RTF), the out-of-vocabulary rate (OOV), and the perplexity are discussed.

[2] `http://www.openfst.org`, accessed November 1, 2017

Word Error Rate

The word error rate (WER) is a measure to evaluate the performance of speech recognition and machine translation systems. The WER can be calculated from the reference word sequence and a hypothesis word sequence (e.g. the output of the ASR system). Reference and hypothesis can have different lengths. The WER is derived from the Levenshtein distance [73] calculated on a sequence of words rather than on a sequence of characters. The Levenshtein distance is the smallest number of insertions, deletions, and substitutions of words required to change the hypothesis sentence into the reference sentence. It is calculated using the Levenshtein algorithm, which is a dynamic programming algorithm. An example of the Levenshtein distance calculation, which in this case compares a sequence of characters, is depicted in Table 3.1.

		s	**u**	**c**	**c**	**e**	**s**	**s**
	0	1	2	3	4	5	6	7
s	1	0	1	2	3	4	5	6
u	2	1	0	1	2	3	4	5
p	3	2	1	1	2	3	4	5
e	4	3	2	2	2	2	3	4
r	5	4	3	3	3	3	3	4
b	6	5	4	4	4	4	4	4

Table 3.1: Example of the Levenshtein distance calculation on the character level

The word error rate (WER) is calculated as follows:

$$WER = \frac{S + D + I}{N} = \frac{d_L}{N} \tag{3.39}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions and N is the number of words in the reference text and d_L is the Levenshtein distance. The phoneme error rate (PER) can be derived by calculating the Levenshtein distance on sequences of phonemes.

Real Time Factor

The real-time factor (RTF) is used to evaluate the time consumption of an algorithm. For example, in speech recognition, if we use a small language model, the decoding procedure will be fast compared to the decoding of a ASR system with a large language model. In that case the RTF will be smaller than the RTF for the system with the large language model. However, the model with the larger language model usually will have a lower WER and is hence performing more accurately. Often, we want the model with the best accuracy (i.e. lowest WER), but sometimes speed is an issue too, and we want to choose a model that is both accurate and fast. In this thesis, the RTF is calculated by the ratio of the time elapsed by the decoding algorithm divided by the

length of the audio signals:

$$RTF = \frac{t_{elapsed}}{t_{decoded}} \tag{3.40}$$

Since, the computational speed and power varies on different machines, we only compare the RTF of different models or configurations on the same machine.

Out-Of-Vocabulary Rate

The out-of-vocabulary (OOV) rate, typically a percentage, is a metric to measure the number of unknown words in a vocabulary or a text. A vocabulary, or dictionary, is a unique word list of a given text.

The vocabulary OOV rate is defined as the ratio of the number of unknown words in the vocabulary of a test text to the total number of words in the vocabulary of a test text:

$$OOV_{voc} = \frac{n_{voc,unknown}}{n_{voc,total}} \tag{3.41}$$

The running OOV rate is defined as the ratio of the number of unknown running words in a text to the total number of words in a text:

$$OOV_{run} = \frac{n_{run,unknown}}{n_{run,total}} \tag{3.42}$$

It is sometimes beneficial to consider both vocabulary and running OOV rate. It can happen, that the vocabulary OOV rate is quite high, however the unknown words occur only very seldomly so that the running OOV rate is low. The lower a OOV rate is, the better.

Perplexity

The best language model is the one that predicts an unseen test set with the highest probability. Maximising the probability is the same as minimising the perplexity. Perplexity is the probability estimate assigned to a word sequence w_1^N by a language model, normalised by the number of words N:

$$PP := P(w_1^N)^{-\frac{1}{N}} = \Big[\prod_{n=1}^{N} p(w_n|w_{n-1},...,w_1)\Big]^{-\frac{1}{N}} \tag{3.43}$$

For m-gram models this formula reduces to:

$$PP_n = \Big[\prod_{n-1}^{N} p(w_n|w_{n-1},...,w_{n-m+1})\Big]^{-\frac{1}{N}} \tag{3.44}$$

Without a language model, the perplexity would correspond to the vocabulary size, since the model could choose from any arbitrary word from the vocabulary with the

same probability. By training a language model, i.e. learning the typical word sequences of a language, the perplexity is reduced for a similar text.

To obtain a low perplexity (and respectively a high probability) on a given test text, the language model must be trained on a similar training text, where the words and phrases occur likely. For example, if we want to train a language model for the broadcast domain (news, interviews, documentations, etc.), it is beneficial to use training data coming from the broadcast domain as well. In contrast, a language model trained on weather reports will perform badly if used for sport reports and vice versa, because first of all, many words will remain unseen by the language model, and many others will we seen with a probability either too high, or too low. Hence, a language model will have a high perplexity on out-of-domain data. Sometimes it is desired to create a general purpose language model for all possible domains, but usually a domain specific LM will optimise the perplexity for a given domain. A common way to achieve lower word error rates in speech recognition is, amongst others, to reduce the perplexity of the language model on a withheld development set [74].

3.5 Dialect Identification

In speech there is, besides the spoken text, paralinguistic information encoded. The paralinguistic information includes the gender, the age, the health condition, emphasis and regional accent and dialect of the speaker. The accent refers to variations in the pronunciation (phone sequences and realisations) and the speaking style (rhythm, variation in pitch) [75]. It can be differentiated between regional accents and foreign accents. Regional accents are typically harder to identify and show subtler differences compared to foreign accents. Dialects on the other hand refer to differences in the word selection and the use of the grammar [75]. Usually speakers that speak dialect also have an accent of the same region. Accent/dialect identification refers to the task of recognising the speaker's regional accent/dialect, within a predetermined language, given a sample of his/her speech [76]. Dialect identification systems have been used successfully for different tasks. Dialect identification systems allow ASR engines to adapt their acoustic, pronunciation and language models to improve the recognition accuracy [76]. Dialect identification systems also allow text-to-speech synthesis to produce regional speech [76]. They also have been used for targeted advertising, service customisation and audio forensics tasks [77]. Usually dialect identification methods are similar to language identification methods, which is reasonable. In general it is easier to differentiate between languages than between dialect (or even accents). Dialect identification methods can be divided into acoustic methods, phonotactic methods and combinations of both acoustic and phonotactic methods. Acoustic methods exploit differences in the acoustic space (e.g. spectral or prosodic features), while phonotactic methods exploit differences in sequences of phonemes.

3.5.1 Phonotactic Methods

Phonotactic methods exploit differences in the phoneme sequence of an utterance and include the phone recogniser followed by a language model (PRLM) [78] and the parallel PRLM approach [78].

Phone Recogniser followed by Language Model

A phone recogniser (PR) decodes the speech utterance and provides a phoneme sequence. A phone recogniser is either trained on phonetically transcribed audio data, or by using a speech recogniser which uses a phoneme LM instead of a word LM. This can be achieved by training the phoneme LM on a text, where each word is substituted by the phoneme hypothesis derived from a grapheme-to-phoneme converter. The derived phoneme sequence from the phone recogniser is then scored by a (dialectal or language) LM. During training, a language model is trained on the decoded phoneme sequences of each dialect (or language). The language model which minimises the perplexity on a given test utterance is the dialect (or language) hypothesis. The workflow of the PRLM is depicted in Figure 3.19.

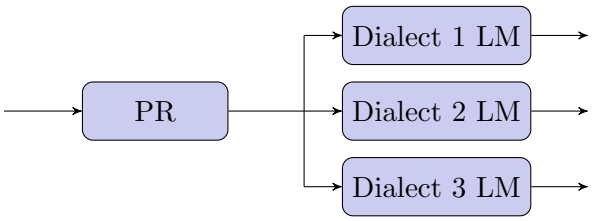


Figure 3.19: Phone recogniser followed by language model (PRLM)

Parallel Phone Recogniser followed by Language Model

Multiple parallel phone recognisers are employed in the parallel PRLM approach [78]. It can be beneficial that the phone recognisers have different phoneme sets and were trained for different languages or dialects. For example, the authors in [76] used phone recognisers trained for the languages English, German, Hindi, Mandarin, Spanish, Modern Standard Arabic to recognise between Modern Standard Arabic and four Arabic dialects. A backend classifier (e.g. logistic regression classifier, SVM, neural network) combines the perplexities to determine the hypothesised dialect (or language). The workflow of the PPRLM is depicted in Figure 3.20.

3.5.2 Acoustic Methods

Acoustic methods exploit differences in the acoustic space (e.g. spectral or prosodic features). Utterance modelling approaches, i.e. approaches that try to model the acoustic features of an utterance as a whole, include the Gaussian Posterior Probability Supervector (GPPS), the Gaussian Mean Supervector (GMS) and the popular i-Vectors

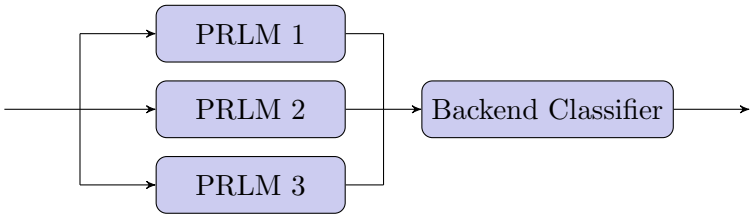


Figure 3.20: Parallel phone recogniser followed by language model (PPRLM)

[79], which originate from speaker identification. The major advantage of the utterance modelling approaches, is that they deliver a utterance feature vector of fixed size, independently of the size of the utterance. Hence, the utterance feature vectors of fixed size can easily be classified by a subsequent classifier e.g. a Naive Bayesian Classifier (NBC) [80] or a Support Vector Machines (SVM) [81].

Universal Background Model

Consider a GMM (Section 3.3.3) that has been trained on a large quantity of speech data, ideally covering all the dialects that we want to be able to differentiate and the standard language itself. The training data should actually cover all the variations, subtleties and nuances of a language. It should also cover a large quantity of speakers of both female and male speakers. However, it should not contain utterances of the test data, that we want to classify, because good results on training sentences might be due to overfitting. The GMM that is trained on all the aspects of a language is considered to be a universal background model (UBM), and is required for the utterance modelling techniques GPPS, GMS and i-Vectors.

Gaussian Posterior Probability Supervector

Consider a GMM-UBM with the likelihood function which is given in Equation 3.12. The occupance posterior probability [77] for the j^{th} mixture component is calculated by:

$$\kappa_j = \frac{1}{T}\sum_{t=1}^{T}\frac{\omega_j \mathcal{N}(\mathbf{o}_t|\mu_j, \mathbf{\Sigma}_j)}{\sum_{j'=1}^{J}\omega_{j'}\mathcal{N}(\mathbf{o}_t|\mu_{j'}, \mathbf{\Sigma}_{j'})}, \tag{3.45}$$

where T is the total number of feature frames in the utterance.

The Gaussian Posterior Probability Supervector (GPPS) is then the vector of the stacked occupancy posterior probabilities for all J mixture components:

$$\kappa = [\kappa_1, ..., \kappa_j, ..., \kappa_J] \tag{3.46}$$

Gaussian Mean Supervector

To calculate the Gaussian Mean Supervector (GMS), the UBM is first adapted to the speech characteristics of the new speaker e.g. by maximum-a-posteriori (MAP) adaptation [82]. In this procedure, the means of the UBM are shifted according to the new speaker data. Then the Gaussian means of the adapted GMM are extracted and stacked to form the Gaussian mean supervector M.

i-Vectors

Total variability modelling has been developed as an alternative method of modelling GMM supervectors and which provides superior performance in speaker recognition [79]. This method extracts so-called i-vectors from the GMM mean supervector M and the mean supervector of the UBM μ. Total variability modelling assumes that M can be decomposed as:

$$M = \mu + Tv, \tag{3.47}$$

where T is the total variability matrix, which transforms the high-dimensional feature space into a low-dimensional feature subspace. The total variability matrix T is estimated via factor analysis. It represents a transformation that is optimised to separate different speech recordings taken from the training set, optimally. The authors in [83] proposed an efficient procedure for training T and for MAP adaption to retrieve the i-vectors v. The i-vectors are a low-dimensional representation of fixed size (typically 400) of audio recordings that can be used for classification purposes.

3.5.3 Evaluation Metrics and Performance

This section covers the evaluation and performance metrics that are used for binary and multi-class classification problems.

Binary Classification

Several expressions are used in the terminology for discussing binary classification problems, such as detection problems, e.g. speech/non-speech detection.

- **True positive (TP)** refers to the number of correctly hypothesised positive cases in the data.
- **True negative (TN)** refers to the number of correctly hypothesised negative cases in the data.
- **False positive (FP)** refers to the number of actual negative cases falsely hypothesised as positive.
- **False negative (FN)** refers to the number of actual positive cases falsely hypothesised as negative.

The recall is calculated as:

$$RC = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{3.48}$$

The precision is calculated as:

$$PR = \frac{TP}{TP + FP} \tag{3.49}$$

The specificity is calculated as:

$$SP = \frac{TN}{N} = \frac{TN}{TN + TP} \tag{3.50}$$

The accuracy is calculated as:

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.51}$$

The F_1 score, which is the harmonic mean of precision and sensitivity, is calculated as:

$$F_1 = 2 \cdot \frac{PR \cdot RC}{PR + RC} \tag{3.52}$$

Multi-Class Classification

In multi-class classification problems (e.g. dialect identification) often the confusion matrix is used for the visualisation of the performance of multi-class classifiers. Each column of the matrix represents the predicted class while each row represents the actual class. The confusion matrix allows to see if the model is confusing two or more classes. An exemplary confusion matrix is depicted in Figure 3.21.

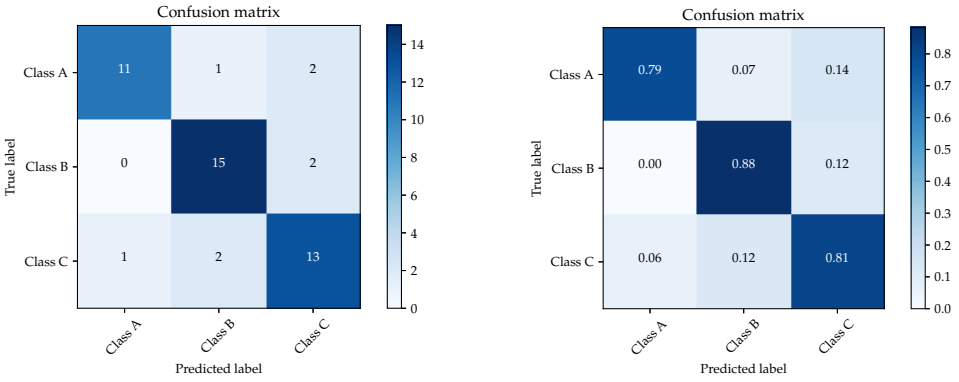


Figure 3.21: Exemplary confusion matrix of size 3×3; (left) unnormalised; (right) normalised

For confusion matrices of size $n \times n$ the accuracy is calculated as:

$$ACC = \frac{\sum_{i=1}^{n} a_{ii}}{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}} \tag{3.53}$$

where a_{ij} is the matrix element of the i-th row and the j-th column.

Chapter 4

Long-Term Development of a German Broadcast Speech Recognition System

Archivists, journalists and content hosters often have the problem of dealing with huge amounts of heterogeneous audio-visual media data. The media objects are usually only accompanied with few metadata such as the title and probably a few keywords, and search algorithms can often only search on this sparse metadata. Hence, finding the requested information is often considered a lucky strike or even impossible. Audio mining systems, like the Fraunhofer IAIS audio mining system [84], solve this issue by automatically analysing the media data, and then providing efficient search tools that support the users to find the information that they seek. The most important component of an audio mining system is the speech recognition system, because it automatically translates speech in audio signals into text, which is the key result for subsequent search and recommendation engines. The field of speech recognition is of broad interest for both the scientific and the industry community and typically many developments are achieved in this field in short periods of time. Typically the advances of speech recognition systems are reported on standard English datasets, e.g. the Switchboard corpus [3] which contains utterances of English telephone speech. Since it is unclear how well these techniques perform in the German broadcast domain, and in order to maintain a competitive audio mining system, this chapter deals with the long-term development and optimisation of a German broadcast speech recognition system in the context of the Fraunhofer IAIS audio mining system, where we employ the latest state-of-the-art algorithms in the context of German broadcast speech recognition.

We first introduce the Fraunhofer IAIS audio mining system in Section 4.1. Then, in Section 4.2 the baseline speech recognition system component itself, and the resources necessary to train and evaluate it, are described. The baseline speech recognition configuration serves as a starting point of research for this thesis. In Section 4.3 we expand the training corpus dramatically which is used to train the ASR system, and explore different ASR architectures that became available in the course of this thesis

to improve the speech recognition system. Section 4.4 summarises this chapter.

4.1 The Fraunhofer IAIS Audio Mining System

The architecture of the Fraunhofer IAIS audio mining system is depicted in Figure 4.1.

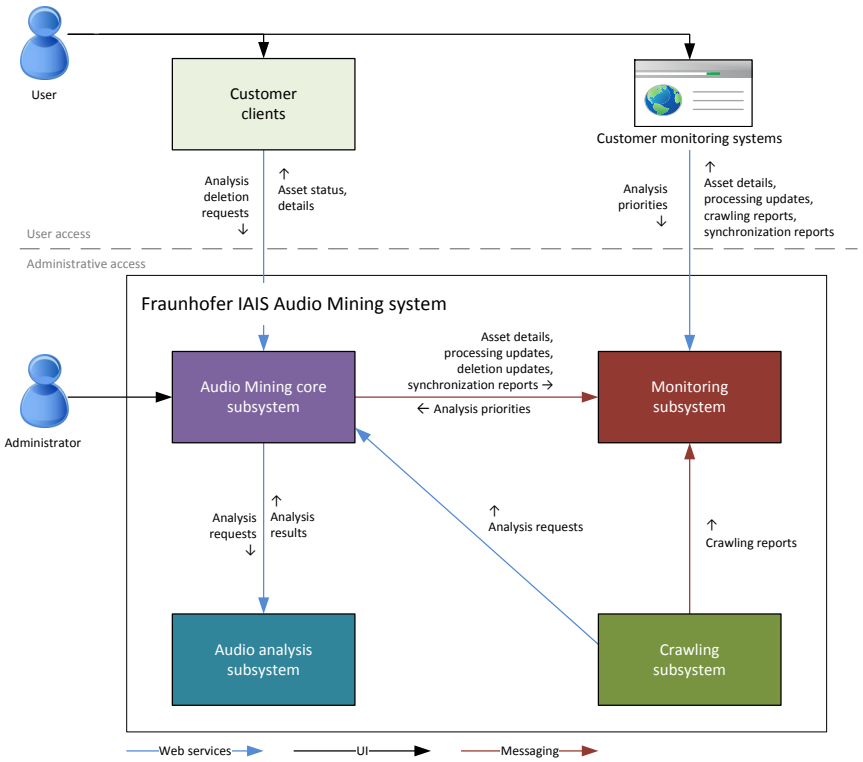


Figure 4.1: The Fraunhofer IAIS Audio Mining system architecture

The Fraunhofer IAIS audio mining system is composed of four subsystems:

- The audio analysis subsystem (Figure 4.2) contains services that perform a detailed audio analysis on the media objects, namely audio segmentation, speech/non-speech detection, gender detection, speaker clustering and recognition, automatic speech recognition and keyword extraction.
- The audio mining core subsystem combines a media asset archive, the search functionality and a recommendation engine based on the audio analysis results.
- The audio mining monitor subsystem provides a way to track the progress of the analysis and to perform administrative tasks.

- The crawling subsystem gathers assets from a variety of sources that needs to be analysed.

Figure 4.2: Workflow of the audio analysis subsystem

By using a webservice-oriented architecture and message-based communication, the system offers a high degree of flexibility. It can be integrated into a customer's archive by using it as a metadata enrichment service, or it can be used as a stand-alone media archive. The graphical user interface of the stand-alone media archive is depicted in Figure 4.3. The system is able to cope with huge amounts of heterogeneous audiovisual media data. The system has been employed with great success for several public German broadcasters and in many commercial and research projects. We have published an overview paper of the Fraunhofer IAIS audio mining system in [84], where more detailed information about the system can be found.

The most important subsystem of the Fraunhofer IAIS audio mining system is the audio analysis subsystem, where the actual audio mining takes place. One of the key components of the audio analysis subsystem is the automatic speech recognition component that transforms the audio signal in text, which can be further analysed by subsequent analysis, search and recommendation algorithms. The speech recognition system that has been employed in the Fraunhofer IAIS audio mining system, and which serves as a baseline configuration in this thesis, is described in the following section.

4.2 Baseline Speech Recognition System and Resources

In this section the speech recognition system is described, which is used as a baseline configuration in this thesis. It was employed within the Fraunhofer IAIS audio mining system at the start of this thesis. Also, the resources necessary to train and evaluate this baseline configuration are described in this section. In Section 4.2.1 the training corpus and a development corpus, which are necessary to train the baseline ASR system, are described. In Section 4.2.2 the difficult speech corpus, which is used for evaluation purposes throughout this thesis, is described. In Section 4.2.3 another speech recognition evaluation corpus is described, namely the LinkedTV corpus. In Section 4.2.4 the baseline speech recognition system is described and evaluated.

4.2.1 Audio Mining Corpus

The audio mining (AM) corpus is a collection news, interviews, talk shows and documentaries from German broadcast. The training set (AM-train) consists of 105 hours of training data. Hence, the corpus covers a mixture of planned speech, as found in news shows, and spontaneous speech, as found in interviews and talk

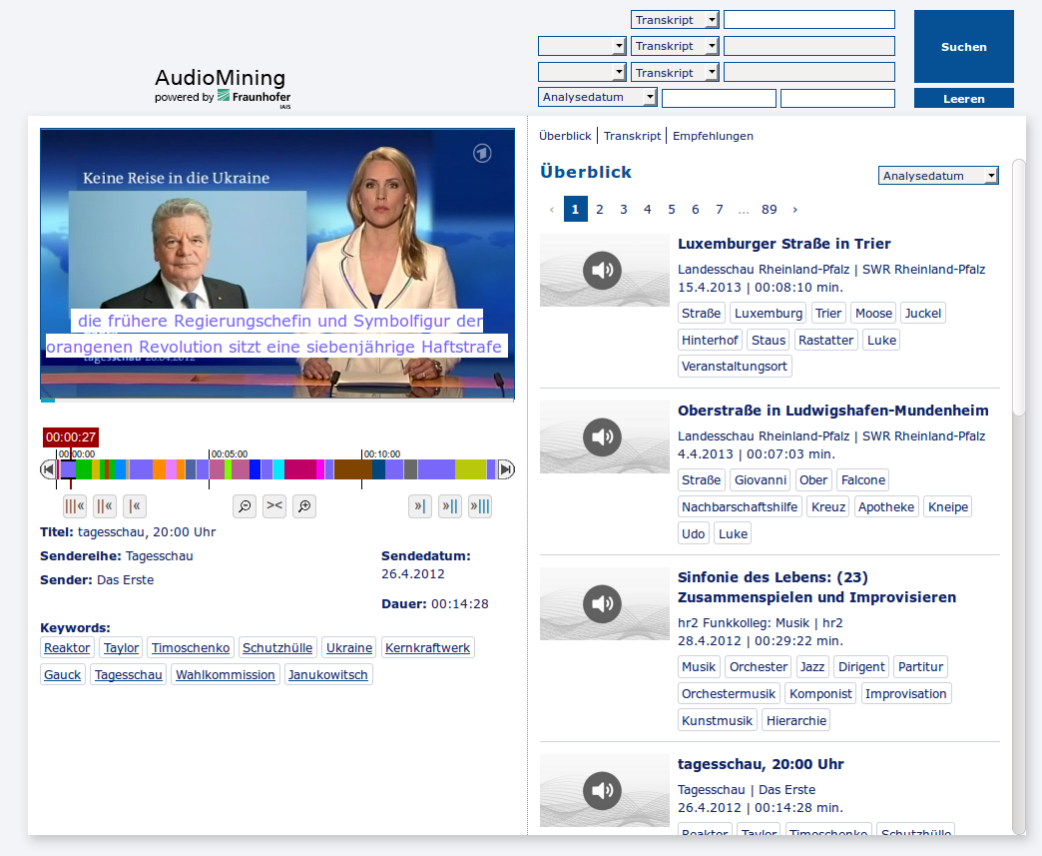


Figure 4.3: Graphical Web User Interface of the Fraunhofer IAIS Audio Mining system

shows. Accompanied to the corpus is a development corpus (AM-dev) with a total of 2,348 utterances (33,744 words), with a similar composition of planned and spontaneous speech as the training set. Detailed corpus statistics are listed in Table 4.1.

Corpus	Size (hh:mm)	Number of Utterances	Avg. length (seconds)	Word count	Avg. Words
AM-train	105:24	119,386	3.18	997,996	8.36
AM-dev	3:29	2,348	5.33	33,748	14.37

Table 4.1: Audio mining (AM) corpus statistics

4.2.2 Difficult Speech Corpus

The Difficult Speech Corpus (DiSCo) is presented in [85]. This heterogeneous German broadcast corpus covers a variety of serious programmes including several challenging conditions for speech analysis like background noise, cross-talk situations, spontaneous

speech and dialects. The programmes covered in this corpus are news, political interview shows, sport commentaries, science shows, political talk shows, regional reports, foreign affairs reports and television news magazines. The data covers a total of 18 hours of video material. The data was segmented and transcribed by a professional typist. The segments are first annotated by the labels: 'non-speech', 'unintelligible', 'cross-talk' and 'speech'. Speech segments are orthographically transcribed. Special markers were used for hesitations and mispronounced words. The type of speech is labelled for each segment by choosing one of the markers: 'spontaneous', 'planned', or 'undecided'. The noise type of each segment is described by one of the markers: 'none', 'music', 'background speech', 'applause', or 'other'. The presence of a dialect is annotated by the markers: 'yes' or 'no'. Also, the speaker is annotated by providing the name ('firstname_lastname'), and the gender ('male' or 'female'). Table 4.2 shows the statistics of the DiSCo subsets. Further information about the corpus can be found in [85].

Speech Type	Noise Type	Size (hh:mm)	Number of Utterances	Avg. length (seconds)	Number of Words	Avg. Words
planned	clean	00:55	1,364	2.43	9,184	6.73
planned	music	01:11	1,780	2.38	10,354	5.79
planned	other	01:46	2,633	2.43	16,711	6.35
planned	mix	01:27	2,200	2.37	13,698	6.23
planned	speech	00:29	727	2.39	5,054	6.95
planned	applause	00:06	115	3.09	994	8.64
planned	dialect	00:12	318	2.36	2,179	6.85
spontaneous	clean	01:55	2,861	2.41	20,740	7.25
spontaneous	music	00:05	120	2.36	850	7.08
spontaneous	other	01:03	1,379	2.74	11,741	8.51
spontaneous	mix	01:06	1,650	2.40	12,071	7.32
spontaneous	speech	00:10	727	2.34	2,067	8.30
spontaneous	applause	00:04	100	2.66	782	7.82
spontaneous	dialect	01:11	1,647	2.60	13,123	7.97

Table 4.2: Difficult Speech Corpus (DiSCo) subset statistics

4.2.3 The LinkedTV Evaluation Corpus

Radio Berlin-Brandenburg (RBB) provided speech datasets to the LinkedTV[1] project, in which Fraunhofer IAIS was part of, again separated into a planned set (1:08h, 787 utterances) and a spontaneous set (0:44h, 596 utterances). While during the annotation of the DiSCo corpus a strong emphasis was put on selecting only segments with clean acoustics (i.e. no background music, high quality) and without dialectal speech, this was not feasible for the LinkedTV corpus. Therefore, especially the spontaneous set

[1]EU-Project: LinkedTV - Television Linked To The Web, Project-ID: 287911, funded under FP7-ICT programme, October 2011 - March 2015, `https://www.linkedtv.eu`

contains utterances from street interviews, partly with strong Berlin dialect (e.g. “dit Jahr war ja nich berauschend bei Hertha wa” instead of “das Jahr war nicht sehr berauschend bei Hertha”), partly with moderate background noise. The statistics of the LinkedTV datasets are depicted in Table 4.3.

Corpus	Speaking style	Size (hh:mm)	Number of Utterances	Avg. length (seconds)	Word count	Avg. Words
LinkedTV	planned	1:08	787	5.18	10,984	13.96
LinkedTV	spontaneous	0:44	596	4.42	8,869	14.88

Table 4.3: LinkedTV evaluation corpus statistics

4.2.4 Baseline Speech Recognition System

The baseline configuration has been proposed in [86] and evaluated in [85]. The system uses the HTK-Toolkit [61], which was considered to be state-of-the-art when this thesis was launched in the year 2012. For decoding, the Julius [62] decoder was used, which has the advantage, that it can be used commercially without a special license. The system is based on hidden Markov models (HMM) with Gaussian mixture models (GMM) (Section 3.4.4) to model the emission probabilities of the states. It employed context-dependent tied crossword triphones using 32 mixtures per state in 3-state HMMs, leading to 240k Gaussians in 7.5k states. The audio mining training set (“AM-train”) (Section 4.2.1) was used for training the model. The system has a vocabulary size of 200,000 words and a trigram language model, which has been trained on texts gathered from broadcast domain covering a total of 75 million running words.

The system has been evaluated on various evaluation sets. The audio mining development set (“AM-dev”) is described in Section 4.2.1. The DiSCo evaluation subsets are described in detail in Section 4.2.2. The LinkedTV evaluation sets are described in Section 4.2.3. The evaluation results are listed in Table 4.4. It can be seen that naturally the ASR system performs better on planned speech than on spontaneous speech, which is more difficult to model. Also, the LinkedTV corpora are typically more difficult than the clean sets of the DiSCo corpus, since they contain noises of different types as they appear normally in broadcast programmes, in contrast to the clean sets of the DiSCo corpus, which cover only the speech segments with low or no background noise.

Configuration	AM-dev	DiSCo planned clean	DiSCo spontaneous clean	LinkedTV planned	LinkedTV spontaneous
Baseline HMM-GMM [85]	30.2	26.4	33.5	27.0	52.5

Table 4.4: WER [%] results on various corpora for the baseline configuration

4.3 Improvements to the Speech Recognition System

This section covers the advances of the Fraunhofer IAIS speech recognition system, which were developed as part of this thesis. They cover the collection and exploitation of a large-scale German broadcast speech recognition corpus, namely the GerTV1000h corpus (Section 4.3.1), and the evaluation of different speech recognition system architectures, which emerged throughout the years by the scientific community, for their employment in the German broadcast domain.

4.3.1 Large-Scale German Broadcast Speech Corpus

In this section we describe the large-scale German broadcast corpus, which we proposed in [87] and which we call the "GerTV1000h" or the "GerTV" corpus. We collected and manually transcribed a huge and novel training corpus of German broadcast video material, containing 2,705 recordings with a volume of just over 900 h. The new corpus is segmented into utterances with a mean duration of approximately 5 seconds, yielding 662,170 utterances, and is transcribed manually on the word level. The total number of running words is 7,773,971 without taking additional annotation into account. Individual speakers are not annotated, but speaker changes within an utterance are marked and allow for a rough speaker adaptive training scheme. The recorded data covers a broad selection of news, interviews, talk shows and documentaries, both from television and radio content across several stations. In addition to the verbatim transcript, the tags in Table 4.5 were used to further describe the recordings. The tags denote the occurrences of audible background noise, speaker noise, hesitations, speaker changes within an utterance, cross-talking speakers, foreign words, mispronounced words, untranscribable utterances, unintelligible words or word fragments.

Together with the audio mining corpus (Section 4.2.1), which is from now on considered to be included in the GerTV corpus, the corpus has grown to over 1,000 hours of transcribed German broadcast data. We also adopt the development subset of the audio mining dataset as the development corpus for the GerTV corpus. All audio is recorded and stored in 16-bit PCM waveform files, with 16 kHz sampling frequency and a single mono channel.

The corpus was not fully available throughout this thesis, since the annotation process was time-consuming. Hence we used different amounts of training data for the creation of the speech recognition systems. However, we always report the size (in hours) of the training amount when necessary. The different training sets that we used in this thesis and the development corpus related to the GerTV corpus are listed in Table 4.6. We discarded utterances where mispronunciations or unintelligible words or utterances occurred for the training of the ASR system. That is why the full training set has 992 hours of speech data.

Label	Description
⟨int⟩	If an utterance contains clearly audible background noises, it is tagged with ⟨int⟩. The type and the volume of the noise was not differentiated in this annotation sequence.
⟨spk⟩	This tag denotes various speaker noises, such as breathing, throat clearing or coughing.
⟨fil⟩	All kinds of hesitations are labelled with this tag.
⟨spk_change⟩	If the speaker changes during the utterance, ⟨spk_change⟩ is inserted at the corresponding position. Using this annotation, speaker turns can be inferred and then used for speaker-adaptive training schemes in later steps.
⟨overlap⟩	If more than one speaker is talking at the same time, the utterance is marked with this tag.
⟨foreign⟩	One or more foreign words, sometimes proper names but most of the time original material with a voice-over.
⟨mispron⟩WORD⟨mispron⟩	Clearly mispronounced words are enclosed in this tag.
⟨reject⟩	If a whole utterance can not be transcribed, it is marked with this tag.
**	If one or more words are unintelligible (e.g. due to background noise), they are transcribed with **.
=	Word fragments are transcribed and end with =, marking them as incomplete.

Table 4.5: Labels used for the annotation of the GerTV corpus

Training Set	Duration (h)	# Utterances	# Words total	# Words unique
TS I	105.0	119,386	997,996	62.206
TS II	322.0	292,133	3,204,599	118,891
TS III	636.0	529,207	5,940,193	181,638
TS IV (full)	992.0	773,631	9,406,119	243,313
dev	3.5	2,348	33,748	6,376

Table 4.6: Training and development datasets of the GerTV corpus

4.3.2 Extension and Optimisation of the Baseline System

The baseline ASR system has been described in Section 4.2.4. At the timepoint of the experiments, we were in the process of collecting and transcribing a huge amount of German broadcast speech for the GerTV1000h corpus, which we described in Section 4.3.1. In this setup, we extended the training material of the baseline system

and hence, used TS II (322 hours of speech data, c.f. Table 4.6) for training the ASR system. In order to use the additional data, the training setup using HTK [61] was heavily modified. The initial configuration with approx. 105 h training data (TS I, c.f. Table 4.6) used tied crossword triphones using 32 mixtures per state in 3-state HMMs, leading to 240k Gaussians in 7.5k states. Since simply increasing the number of mixtures led to unsatisfactory results, the state tying configuration was adapted. Using fewer states and therefore more training data per state solved the problem of overfitting infrequent states and leads to significant improvements in WER, especially in acoustically challenging situations like spontaneous speech. The extended model uses 575k Gaussians for 6k states and generalises well to previously unseen material, as can be seen from Table 4.7. A consistent improvement is noticeable for all the regarded evaluation corpora due to the increased training amount and the increased complexity of the model. For evaluation, we use the GerTV1000h development corpus (Section 4.3.1), the clean datasets of the DiSCo corpus (Section 4.2.2, planned and spontaneous) and the LinkedTV datasets (Section 4.2.3).

By applying the techniques developed in Chapter 5, we optimised the ASR speech recognition decoder parameters of the extended GMM-HMM system by the employment of the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [88] (Section 5.1.1) by the use of the development set and we were able to further improve the performance of the system by a large margin as evaluated on the evaluation corpora (see Table 4.7).

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.	**WER** LinkedTV planned	**WER** LinkedTV spont.
GMM-HMM [85]	105	30.2	26.4	33.5	27.0	52.5
GMM-HMM	322	29.6	24.0	31.1	26.4	50.0
GMM-HMM-SPSA	322	27.7	22.6	28.4	24.5	45.6

Table 4.7: WER results on several corpora for the baseline configuration, the extended configuration and the extended configuration with the optimised decoder hyperparameters by the employment of the SPSA algorithm [88]

4.3.3 Subspace Gaussian Mixture Models

[89] describes an acoustic modelling approach which is called Subspace Gaussian Mixture Model (SGMM). In this model all phonetic states share a common Gaussian Mixture Model structure, and the means and mixture weights vary in a subspace of the total parameter space. Parameters which are shared globally define the subspace. As the results derived from the experiments of the authors indicate, the SGMM gives better results than a conventional modelling approach, particularly with smaller amounts of training data. This style of acoustic model allows for a much more compact representation. The basic form of the model can be expressed by:

$$p(x|j) = \sum_{i=1}^{I} w_{ji} \mathcal{N}(x; \mu_{ji}, \Sigma_i) \quad (4.1)$$

$$\mu_{ji} = M_i v_j \quad (4.2)$$

$$w_{ji} = \frac{\exp(u_i^T v_j)}{\sum_{i'=1}^{I} \exp(u_{i'}^T v_j)} \quad (4.3)$$

where $x \in \mathcal{R}^D$ is the feature, j is the speech state, $v_j \in \mathcal{R}^S$ is the state vector with $S \simeq D$ being the subspace dimension. The model in each state is a simple GMM with I Gaussians, and mixture weights w_{ji}, means μ_{ji} and covariances Σ_i which are shared between states. The means and mixture weights are not parameters of the model. Instead they are derived from a state-specific vector v_j with the subspace dimension S typically being around the same as the feature dimension D. The parameters M_i and u_{ji} are globally shared parameters. The reason why it is a subspace model is that the state-specific parameters v_j determine the means μ_{ji} and the weights w_{ji} for all i. Hence, the subspace dimension of S is typically much lower than $I(D+1)$ in the conventional approach. The model is trained by an expectation-maximisation (EM) [9] procedure, like the normal HMM training. However, the training involves a training of a universal background model (GMM-UBM, c.f. Section 3.5.2), to estimate M and v using a similar approach as in speaker adaptive training (SAT) [90]. For further information about the approach see [89].

Experiments

In order to evaluate if the SGMM approach also yields improvements in the context of German broadcast speech we trained a speech recognition system based on the implementation of the SGMM approach in the Kaldi toolkit [55]. The GMM-UBM is trained with 700 Gaussian mixtures. The SGMM model was trained with 30,000 substates and 9,000 probability density functions. The model was trained with 25 iterations. We trained the speech recognition on the GerTV corpus (Section 4.3.1) training set TS II (322 hours) and TS III (636 hours). The results and a comparison with the previous models are listed in Table 4.8. It can be seen that both SGMM configurations clearly outperform the previous models. The SGMM with the additional training data only slightly improved the results compared SGMM trained on 322 hours. However, the authors noted that the model especially performs well already with a small amount of training data. On the other hand, a more complex model, e.g. by increasing the number of substates or probability density functions could possibly further improve the results for higher amounts of training data.

4.3.4 Hybrid Deep Neural Network Hidden Markov Models

Major advances have been reported in training densely connected directed neural networks with many hidden layers. The result is a deep belief network [46] that is able

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.	**WER** LinkedTV planned	**WER** LinkedTV spont.
GMM-HMM [85]	105	30.2	26.4	33.5	27.0	52.5
GMM-HMM	322	29.6	24.0	31.1	26.4	50.0
GMM-HMM-SPSA	322	27.7	22.6	28.4	24.5	45.6
SGMM-HMM	322	23.5	18.1	22.5	21.0	36.6
SGMM-HMM	636	23.3	18.1	22.4	20.5	35.9

Table 4.8: WER results on several corpora for the SGMM based ASR models in comparison to the previous models

to learn a large set of nonlinear feature detectors which can capture sophisticated statistical patterns in the data. First the algorithm initialises the weights of each layer individually with the use of the acoustical training data. However, the labels are not used during the initialisation step. After the initialisation procedure the entire network is fine-tuned by the use of labelled training data. This semi-supervised approach using deep belief network has proven to be effective in numerous speech, audio, text and image applications [91, 92, 93, 94].

In [45] a context-dependent model for large vocabulary continuous speech recognition is presented which is based on pre-trained hybrid deep neural network hidden Markov model (DNN-HMM) architecture. The algorithm trains the DNN to produce a distribution over senones (tied triphone states) as its output. The hybrid DNN-HMM architecture is depicted in Figure 4.4. The DNN models the observation likelihood of all senones [95], and the HMM models the sequential property of the speech signal. The authors report a relative sentence error reduction of 16.0 % and 23.2 % compared to discriminatively trained GMM-HMM trained on the MPE and ML criteria, respectively.

Experiments

In order to evaluate the approach in the context of German broadcast speech recognition, we train a hybrid DNN-HMM [45] using the implementation from the Kaldi toolkit [55] with the use of the training data of the GerTV corpus (Section 4.3.1). We use 4 hidden layers with 1024 neurons in each layer and the tangens hyperbolicus activation function. The initial learning rate was set to 0.01, and the final learning rate was set to 0.001. The training was performed for a total of 20 epochs, 15 epochs with reducing learning rate, and 5 extra epochs with the final learning rate.

In Table 4.9 the results are listed in comparison to configurations developed in the previous sections. Note that with the end of the LinkedTV project the evaluation datasets became unavailable, which is the reason why we cannot report numbers for some configurations. It can be derived that the employment of the hybrid DNN-HMM architecture drastically reduces the WER on every evaluation dataset. Also, the extension of training material to 992 hours, which is considered to be the full dataset of the GerTV corpus (only few utterances are disregarded due to mispronunciations, or

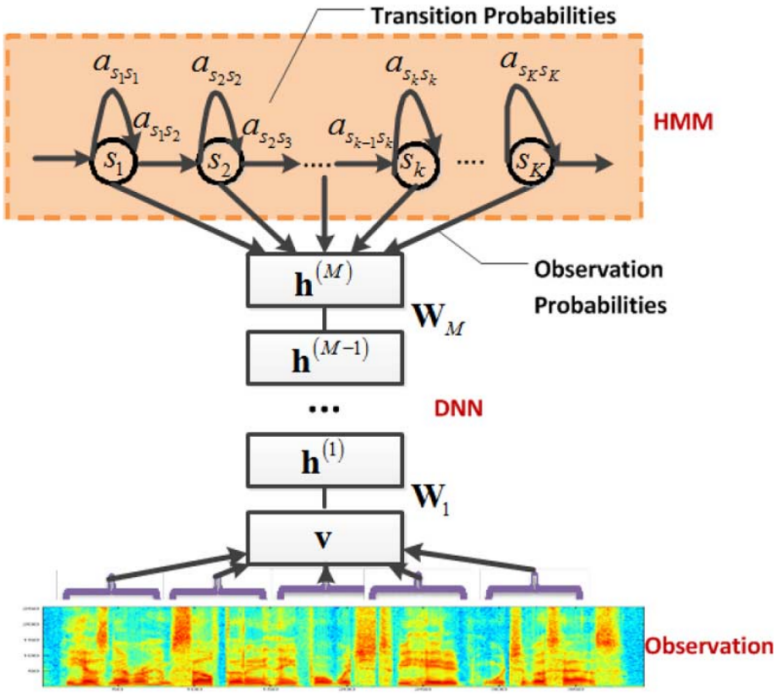


Figure 4.4: Diagram of the hybrid DNN-HMM architecture [45]

foreign language), still improves the accuracy of the system. This is the first configuration that makes use of the complete GerTV corpus. It has been employed in the Fraunhofer IAIS audiomining system for several years.

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.	**WER** LinkedTV planned	**WER** LinkedTV spont.
GMM-HMM [85]	105	30.2	26.4	33.5	27.0	52.5
GMM-HMM	322	29.6	24.0	31.1	26.4	50.0
GMM-HMM-SPSA	322	27.7	22.6	28.4	24.5	45.6
DNN-HMM	322	23.9	18.4	22.6	21.2	37.6
DNN-HMM	636	22.7	17.4	21.5	19.9	35.3
DNN-HMM	992	21.3	15.5	19.7	-	-

Table 4.9: WER results for the hybrid DNN-HMM systems in comparison to the previous models

4.3.5 Recurrent Neural Network Rescoring

The author in [96] proposed a recurrent neural network based language model. The architecture employed is called a simple recurrent neural network or Elman network [97]. The network has an input layer x, and hidden layer s and an output layer y. The input vector $x(t)$ is formed by concatenating the vector w representing the current word and the output from the neurons in the hidden layer s at time $t-1$. Input, hidden and output layers are computed as follows:

$$x(t) = w(t) + s(t-1) \tag{4.4}$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right) \tag{4.5}$$

$$y_k(t) = g\left(\sum_j s_j(t) v_{kj}\right) \tag{4.6}$$

where $f(z)$ is the sigmoid activation function and $g(z)$ is the softmax activation function. The input vector $x(t)$ represents the word at time t encoded using a 1-of-N encoding. Hence, the size of the vector x is equal to the size of the vocabulary (in practice 30k to 500k). The size of the hidden layer s is usually 30 to 500 hidden units, and typically reflects the size of the training text. The output layer $y(t)$ represents the probability distribution of the next word given the previous word $w(t)$ and the context $s(t-1)$. The network is trained for several epochs in which all data is sequentially presented. To train the network, the standard backpropagation algorithm with stochastic gradient descent is employed. The starting learning rate α is 0.1. After each epoch, the validation loss is calculated. If the log-likelihood of the validation set increases, the training continues in a new epoch. Otherwise the learning rate α is halved. If there is again no significant improvement, the training stops. The algorithm usually converges after 10-20 epochs.

Experiments

To evaluate the approach in the context of German broadcast speech recognition, we train an RNN LM on the same text corpus (75 million words from broadcast domain) as for the m-gram language model which is used during decoding. We used the RNN LM implementation of the RNNLM toolkit [98]. The RNN LM is used for rescoring the n-best hypothesis list ($n = 100$) using a mixing value of 0.5 between the scores of the m-gram and the RNN LM. We evaluated the number of hidden neurons (100, 200, 300, 400) for the RNN LM on the development corpus. The best configuration was the one with 300 neurons, which we evaluate on the test corpora. The results and a comparison with the previous models is depicted in Table 4.10. It can be seen that the WER can be reduced for all evaluation corpora, compared to the hybrid DNN-HMM approach without RNN LM rescoring.

Configuration	Size (hours)	WER dev	WER DiSCo planned	WER DiSCo spont.
GMM-HMM [85]	105	30.2	26.4	33.5
DNN-HMM	992	21.3	15.5	19.7
DNN-HMM-RNN	992	20.0	15.3	18.4

Table 4.10: WER results for the hybrid DNN-HMM systems plus subsequent RNN rescoring in comparison to other configuration

4.3.6 Deep Neural Networks with p-Norm Nonlinearities

The developments on pre-trained deep neural networks continued, the sigmoidal activation functions have been replaced by rectified linear units (ReLU) [99], which is a simple activation function $y = \max(0, x)$. Also, the maxout nonlinearity [100], which can be regarded as a generalisation of ReLU, was proposed and employed with great success in speech recognition [101]. In [102] a speech recognition system is proposed based on hybrid DNN-HMM with p-norm non-linearities. p-norm non-linearities are inspired from maxout nonlinearities. In a maxout network, the nonlinearity has a dimension-reducing nature. If we suppose we have K maxout units (e.g. $K = 500$) with a group size of G (e.g. $G = 5$) then the maxout nonlinearity would reduce the dimension from 2500 to 500. For each group of 5 neurons, the output would be the maximum of all the inputs:

$$y = \max_{i=1}^{G} x_i \tag{4.7}$$

The p-norm nonlinearity, as proposed by the authors ([102]), is calculated in a similar fashion as:

$$y = ||x||_p = \left(\sum_{i}^{G} |x_i|^p \right)^{1/p} \tag{4.8}$$

The value of p and G is configurable, however the results favored $p = 2$ and $G = 10$. Normalisation layers are added to stabilise training. For further information on p-norm DNN-HMM see [102].

Experiments

To evaluate the approach in the context of German broadcast speech recognition, we trained a p-norm DNN-HMM model based on the approach published in [102] on the full dataset of the GerTV Corpus (992 hours). The employed p-norm DNN-HMM ($p = 2$) consists of 5 hidden layers each mapping 3000 inputs to 300 outputs ($G = 10$). The input to the network are 13 MFCCs plus first and second derivative with a context window of $\pm$ 4 and using linear discriminant analysis (LDA) to reduce feature size to 250. The network was trained for 15 epochs which a reducing learning rate scheme,

and 5 extra epochs with fixed learning rate. The output of the network using a softmax layer providing emission probabilities of the covering the phoneme states (senones) of the HMM. The results are depicted in Table 4.11 and indicate an improvement over the conventional hybrid DNN-HMM even with RNN rescoring.

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.
GMM-HMM [85]	105	30.2	26.4	33.5
DNN-HMM	992	21.3	15.5	19.7
DNN-HMM-RNN	992	20.0	15.3	18.4
p-norm DNN-HMM	992	18.8	13.3	16.5

Table 4.11: WER results for the hybrid p-norm DNN-HMM system in comparison to other configuration

4.3.7 Recurrent Neural Networks based on Long Short-Term Memory

The authors in [48] propose an end-to-end speech recognition system using deep bidirectional recurrent neural network (RNN) [103] models based on long short-term memory (LSTM) [104, 10] units with weighted finite state transducer based decoding. The acoustic modelling involves the learning of a single RNN, which predicts context-independent targets (e.g. characters or phonemes). The employed RNN is able to learn complex temporal context in sequences. The forward sequence of hidden states $H = (h_1, ..., h_T)$ of a recurrent layer given an input sequence $X = (x_1, ..., x_T)$ is calculated by iterating from $t = 1$ to T:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h), \tag{4.9}$$

where W_{hx} is the input-to-hidden weight matrix and W_{hh} is the hidden-to-hidden weight matrix and σ is the logistic sigmoid nonlinearity. Hence, the hidden outputs of the layer of a given time step h_t is not only dependent on the input x_t, but also on the output of the hidden activation h_{t-1} from the previous time step. In the bi-directional case an additional recurrent layer computes the backward sequence in a similar fashion. Not only the RNN layers contain recurrent connections, also the LSTM units, which are the building block of the RNN, contain recurrent connections. They contain multiplicative gates and memory cells with self-connections to store the temporal states of the network. The architecture of an LSTM unit is depicted in Figure 4.5.

In contrast to previous models, the training of the RNN model is not dependent on bootstrapping (i.e. the training of less complex models, e.g. monophone HMM-GMM to align the data to get pre-generated frame labels). The authors adopt the connectionist temporal classification (CTC) [105] objective function to automatically infer the alignments between the speech data and the labels. CTC aims to maximise $\ln Pr(z|X)$, the log-likelihood of the label sequence z given the input sequence X, by

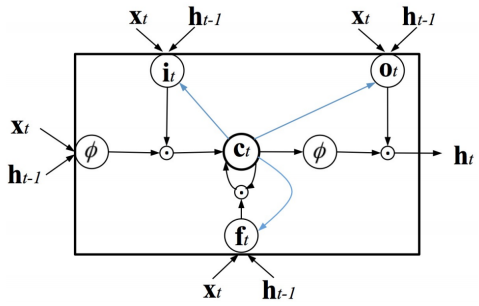


Figure 4.5: Diagram of an LSTM unit [48]

optimising the RNN model parameters. The neural network is trained using the back-propagation through time (BPTT) algorithm [106]. The authors report comparable WER to hybrid DNN-HMM systems, while speeding up decoding and simplifying the training.

Experiments

To evaluate the approach in the context of German broadcast speech recognition, we trained a speech recognition system using deep bidirectional RNNs based on LSTM and WFST as implemented in the Eesen Toolkit [48] with the use of the German broadcast corpus GerTV (Section 4.3.1). The system uses 40 dimensional filter-bank coefficients and first (Δ) and second order ($\Delta\Delta$) derivatives as the input features with a total dimensionality of 120. The output targets of the RNN are the softmax probabilities of the phonemes and the blank character. The features are normalised via variance normalisation and mean subtraction on the speaker level. We used 97 % of the speech data for training the RNN and the remaining 3 % for cross validation. We trained systems with different numbers of layers (4 and 5) and number of neurons (320, 360, 400). The architecture of the best configuration as evaluated on the development corpus is depicted in Figure 4.6.

The results of the optimal configuration on the development and test datasets and a comparison to the previous models is listed in Table 4.12. The RNN model outperforms the hybrid DNN-HMM configuration including the variant with RNN rescoring and the p-norm DNN-HMM by a large margin consistently on the evaluation datasets.

4.3.8 Time Delay Neural Networks

Recently it has been shown that recurrent neural network architectures are able to efficiently model long term temporal dependencies in speech in the context of speech recognition [48]. However the training time is higher than for feed forward networks due to the sequential learning algorithm. Hence, in [107] the authors propose a time delay neural network (TDNN) architecture for speech recognition that uses sub-sampling and which models the long term temporal dependencies with training times comparable to

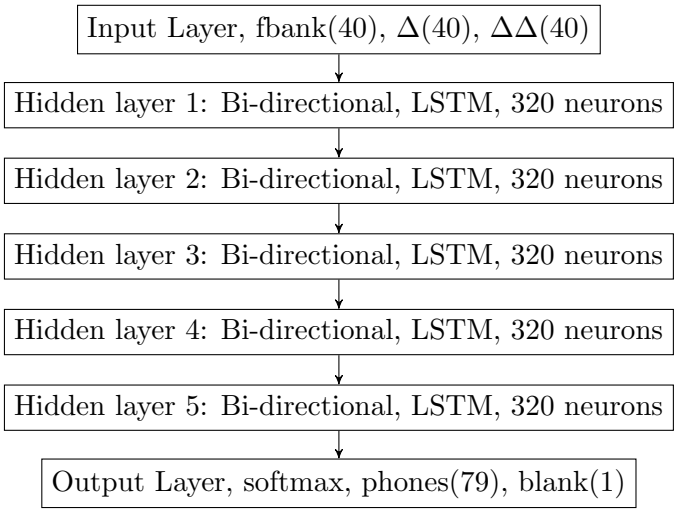


Figure 4.6: RNN architecture

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.
GMM-HMM [85]	105	30.2	26.4	33.5
DNN-HMM	992	21.3	15.5	19.7
DNN-HMM-RNN	992	20.0	15.3	18.4
p-norm DNN-HMM	992	18.8	13.3	16.5
RNN	992	17.2	11.9	14.5

Table 4.12: WER results for the RNN systems in comparison to previous configurations

standard feed forward neural networks. TDNN have been already introduced by [108] in the context of phoneme recognition. Subsampling is used to reduce the computational cost by computing the hidden activations at only few time steps at each level, rather than calculating it for all. However, the activations for input and output layers are computed for all time steps. Through a proper selection of the time steps, at which activations are computed, computational effort can be reduced, while ensuring that information from all time steps in the input with large temporal context is processed by the network. In Figure 4.7 the computation in a TDNN with and without subsampling is depicted. In the TDNN architecture, the lower layers learn the narrow temporal contexts and the deeper layers process the hidden activations from a wider temporal context. Each layer in a TDNN operates at a different temporal resolution, which increases towards the deeper layers. The authors report superior results of the TDNN compared to DNN and RNN approaches.

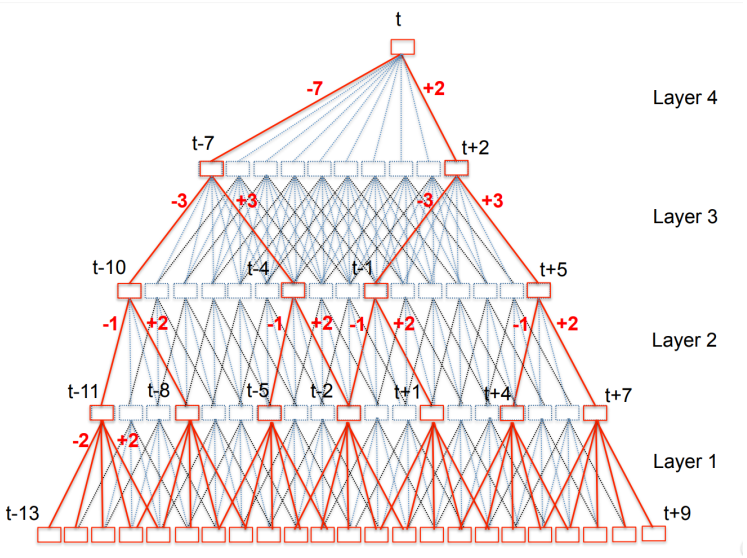


Figure 4.7: Computation in TDNN with (red) and without (blue) sub-sampling [107]

Experiments

To evaluate the approach in the context of German broadcast speech recognition, we train a TDNN on the full dataset of the GerTV corpus following the implementation of the Kaldi toolkit [55]. We used MFCCs without cepstral truncation (40 dimensional) as the input features. In addition to each frame we append the i-vector (Section 3.5.2) of dimension 100, which is calculated over a sliding window of 6 seconds, as suggested by the authors. The MFCC input is not subject to cepstral mean normalisation. The intention is to allow the i-vector to supply the information about any mean offset of the speaker's data, so the neural network can perform the feature normalisation itself. Feature splicing is used to extend the temporal context to $\pm$ 2 frames. Linear discriminant analysis is used to reduce the dimension of the features while preserving the variance of the data. The network is composed of six TDNN layers with different temporal context. The output of the network is a softmax layer which provides outputs that represent the emission probabilities of the senones (phoneme substates). The training is performed with greedy layer-wise supervised training, preconditioned stochastic gradient descent updates, and an exponential learning rate schedule. The architecture of the employed TDNN is depicted in Figure 4.8. The results of the approach are listed in Table 4.13 and indicate that the speech recognition system trained on TDNN outperform the system based on RNNs or p-norm DNN-HMM also for German broadcast speech. They also highlight that TDNNs are capable of exploiting large temporal context speech data.

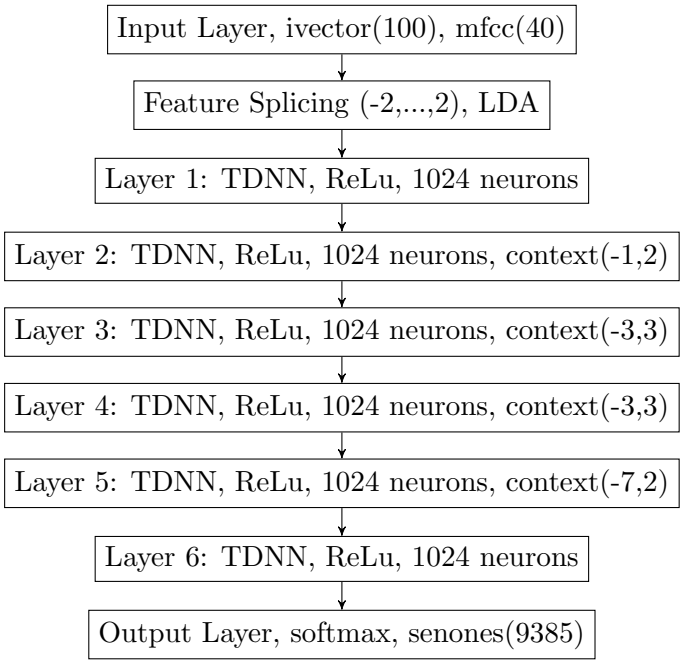


Figure 4.8: TDNN architecture

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.
GMM-HMM [85]	105	30.2	26.4	33.5
p-norm DNN-HMM	992	18.8	13.3	16.5
RNN	992	17.2	11.9	14.5
TDNN	992	15.6	11.1	13.2

Table 4.13: WER results for the TDNN system in comparison to other configuration

4.3.9 Time Delay Neural Networks with Projected Long Short-Time Memory

In [109] the authors undertake an exploration in the context of speech recognition on which is the best way to combine dropout [110] with LSTMs, or more specifically the projected LSTMs (LSTMP, [10]). The authors in [109] propose to use an acoustic model based on a combination of TDNN layers and LSTMP layers, which gave consistent improvements in terms of WER over a large range of datasets, including Switchboard [3], TED-LIUM [111] and AMI [112]. Dropout is an easy way to improve the generalisation of neural networks. The dropout probability p determines what proportion of the mask values are one. Dropout is achieved by multiplying neural net activations by random zero-one masks during training, not during testing. However, in another approach [109] the authors use per-frame dropout instead of the classical per-element

dropout approach, because the per-element approach did not perform very well in the context of speech recognition. In the per-frame dropout, the entire frame vector is multiplied by either zero or one, in contrast to the per-element dropout, in which each element of the dropout mask is multiplied by either zero or one separately. However, the dropout mask is chosen independently over multiple layers or gates. LSTMP have been introduced to address the computational complexity of learning LSTM models [113]. The architecture of an LSTMP unit is depicted in Figure 4.9. By extending the LSTM by a recurrent projection layer, the parameters of the unit can be reduced, as shown in [10].

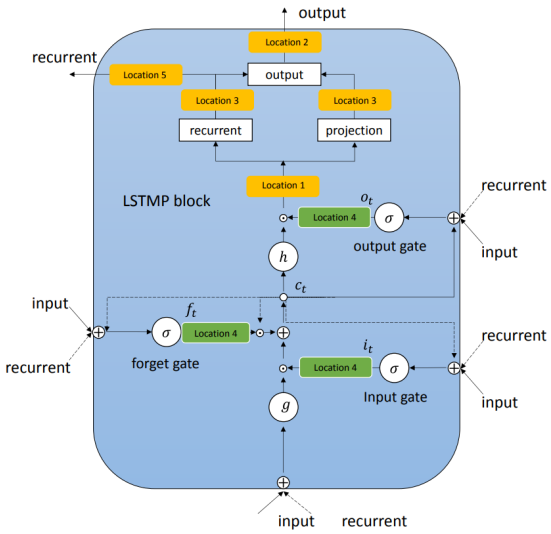


Figure 4.9: Architecture of a projected LSTM block [109]

Experiments

To evaluate the approach in the context of German broadcast speech recognition we trained a model following the approach in [109] using the Kaldi Toolkit [89]. High-resolution MFCCs (40 dimensional, without cepstral truncation) were used as the input features to the neural network. However the features were sliced across ± 2 frames of context and appended by a 100-dimensional i-vector [79]. We again use the full GerTV1000h corpus for training. Speed perturbation is used to augment the data 3-fold [114], so the training data is artificially increased to 3 times 992 hours. Speed perturbation is performed by manipulation the speed of the audio samples to 90 % and 110 % in addition to the unmodified audio data. The sox tool[2] is used to perform the speed perturbation. The architecture of the employed neural network is depicted in Figure 4.10. It uses several TDNN layers with subsampling and LSTMP layers with

[2]http://sox.sourceforge.net/

per-frame dropout. The results of the approach are listed in Table 4.14. Using this approach we could improve the accuracy of our broadcast German LVCSR system by a large margin with a WER of 8.9 % on the DiSCo dataset with planned speech in clean condition. This is the speech recognition system that has recently been employed in the productive Fraunhofer IAIS audio mining system.

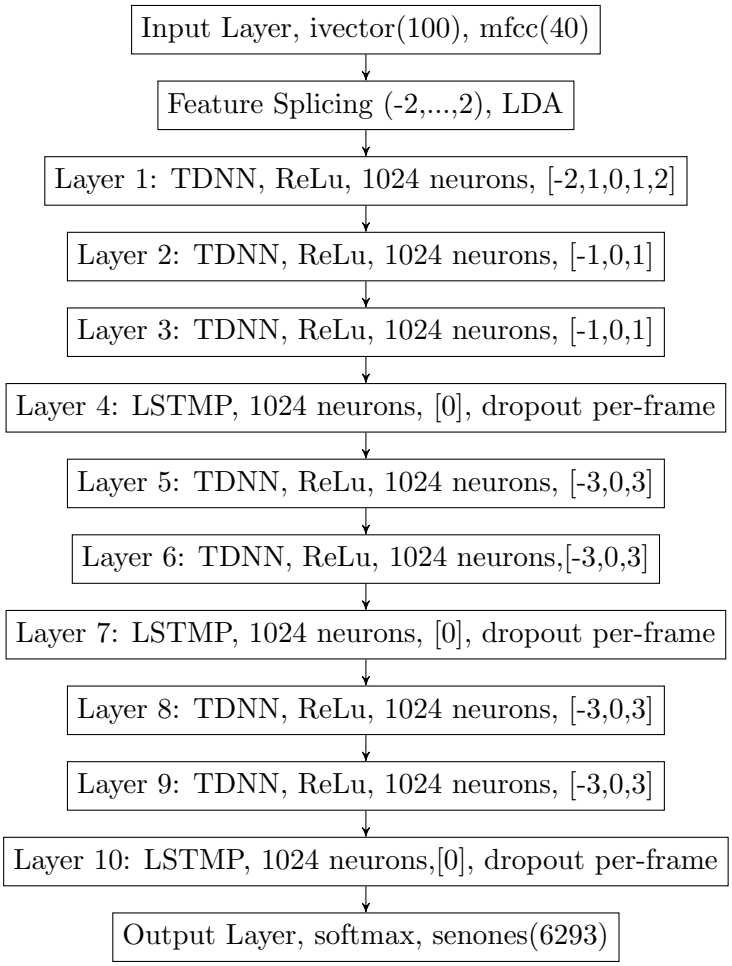


Figure 4.10: TDNN-LSTMP architecture

4.3.10 Language Model Rescoring with Gated Convolutional Neural Networks

Language modelling in applications other than speech recognition is usually performed by neural network architectures. However, in the literature no approach is known for speech recognition to directly employ a neural network language model during the decoding. In speech recognition, language modelling is usually performed by using an m-gram model during decoding. However, one possibility to use neural networks for

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.
GMM-HMM [85]	105	30.2	26.4	33.5
RNN	992	17.2	11.9	14.5
TDNN	992	15.6	11.1	13.2
TDNN-LSTMP	992	13.7	8.9	10.4

Table 4.14: WER results for the TDNN-LSTMP system in comparison to previous configurations

language modelling in speech recognition is to perform a language model rescoring using neural network language models. In [115] a novel method was proposed to perform language modelling with Gated Convolutional Neural Networks (GCNN). In this approach, words are represented by a vector embedding stored in a lookup table $D^{|V|\times e}$, where $|V|$ is the vocabulary size and e is the size of the embedding. A sequence of words $w_0, .., w_N$ is then represented by the word embeddings $E = [D_{w_0}, ..., D_{w_N}]$. The hidden layers $h_0, .., h_L$ are computed as:

$$h_l(X) = (X * W + b) \otimes \sigma(X * V + c) \tag{4.10}$$

where $X \in \mathbb{R}^{N\times m}$ is the input of layer h_l (either the word embeddings or the output of the previous layers), $W \in \mathbb{R}^{k\times m\times n}$, $b \in \mathbb{R}^n$, $V \in \mathbb{R}^{k\times m\times n}$ and $c \in \mathbb{R}^n$ are learned parameters, σ is the sigmoid function and $\otimes$ is the element-wise product between matrices, where m, n are respectively the number of input and output feature maps and k is the patch size. The output of each layer is a linear projection $X * W + b$ modulated by the gates $\sigma(X * V + c)$. The authors call this gating mechanism Gated Linear Units (GLU). Convolution is performed by shifting the convolutional inputs to prevent the kernels to see future context. To obtain model predictions a (hierarchical or adaptive) softmax layer is employed. For further information on gated convolutional neural networks for language modelling see [115].

Experiments

In the context of German broadcast speech recognition, we train an 8-layer bottleneck GCNN as proposed in [115], with our standard German broadcast training text data (75 million words) and the same dictionary (500k words), as used in the previous experiments. The hidden layers are GLU layers with dropout and the output layer is a hierarchical softmax layer. We trained the network with Adagrad optimiser and cross-entropy loss function. The batch size was 32, the initial learning rate was 0.8. The learning rate was halved, when there was no improvement on a left out validation set for 4 validation checks and the network state was set reset to the optimal state. A validation check was performed 25 times per epoch. After training we rescored the n-best hypothesis list ($n = 100$) provided by the TDNN-LSTMP model, trained in

Section 4.3.9, using the 8-layer bottleneck Gated CNN. The results are listed in 4.15 and show that a consistent improvement of about 9 % relative can be achieved on the test sets using GCNN rescoring.

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.
TDNN-LSTMP	992	13.7	8.9	10.4
TDNN-LSTMP-GCNN	992	12.7	8.1	9.3

Table 4.15: WER results for the TDNN-LSTMP system with and without GCNN language model rescoring in comparison

4.4 Summary and Contributions

This chapter covers the long-term development of a German broadcast speech recognition system, which is part of the Fraunhofer IAIS audio mining system, where we investigate and evaluate state-of-the-art speech recognition methods for their employment in the German broadcast domain. We first briefly introduced the Fraunhofer IAIS audio mining system (Section 4.1) and highlighted the importance of the speech recognition component. From the speech recognition decoder output not only the text, but also the time boundaries on the word level can be derived. Text and time information is then exploited by a subsequent search engine, which brings together the analytics also from the other components, e.g. speaker clustering, gender detection, or speaker identification. Also, keywords are extracted from the text, which provide a short summarisation of the underlying media material. The speech recognition module must be improved regularly in order to have a competitive audio mining system, by keeping track with the state-of-the-art and by optimising and adapting the system for its employment in the German broadcast domain. This chapter presents the developments of the author during the course of this thesis (2012-2018) in this context. We described the baseline German broadcast speech recognition system (Section 4.2.4) which was proposed in [85] and which was trained on 105 hours of audio data and which is based on GMM-HMM. We also briefly introduced the corpora and resources involved in creating the baseline system (Section 4.2). Then, we collected and annotated a large quantity (900 hours) of German broadcast data and proposed it together with the already available data (105 hours) as the GerTV1000h corpus in [87] (Section 4.3.1). Of course, the annotation of such a large quantity of data is time consuming, therefore we report the developments of the speech recognition system on the amount of training data which was available at the timepoint of the experiments (and mention the quantity).

The developments, which are summarised in Figure 4.11, Figure 4.12 and Table 4.16, started by an extension of the baseline GMM-HMM configuration to a larger quantity of training material and an optimisation (Section 4.3.2). By introducing a gradient-free parameter optimisation algorithm, namely the Simultaneous Perturbation

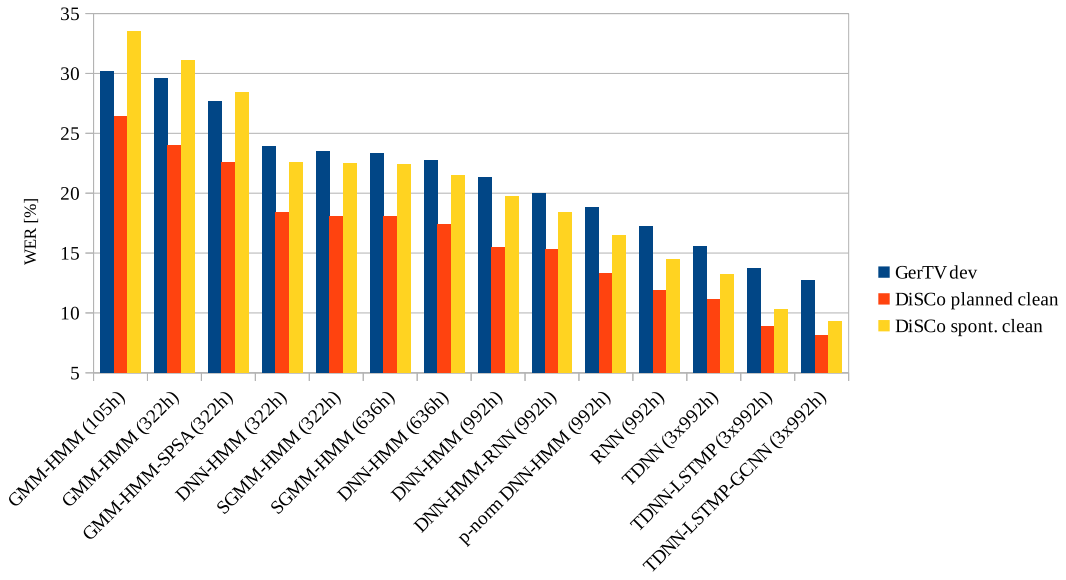


Figure 4.11: Performance of different configurations of the Fraunhofer IAIS speech recognition system, grouped by approach

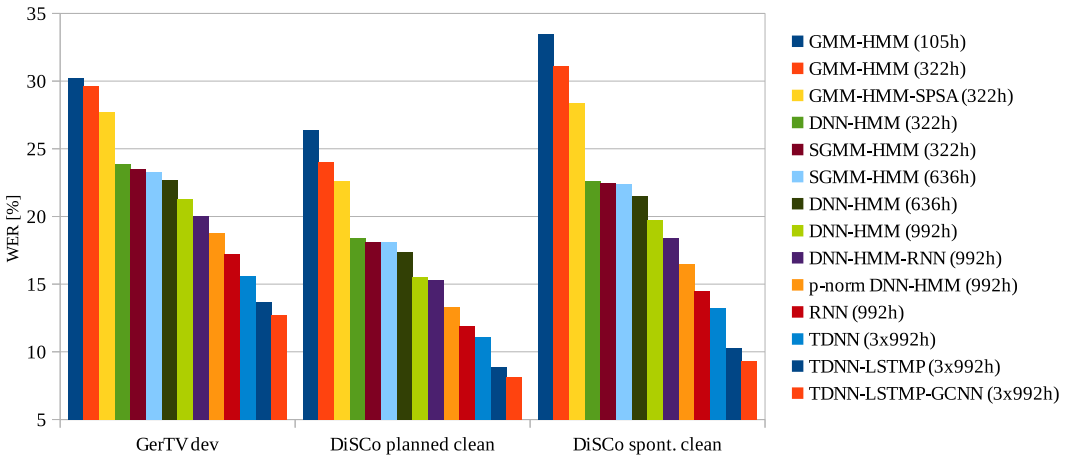


Figure 4.12: Performance of different configurations of the Fraunhofer IAIS speech recognition system, grouped by test set

Stochastic Approximation (SPSA) [88] algorithm, which was not used in the context of speech recognition before, to speech recognition, we were able to further improve this configuration by a large quantity (see Chapter 5). By employing Subspace Gaussian Mixture Models (SGMM) [89], which share common parameters across Gaussian mixtures, we could further improve the German broadcast speech recognition system

(Section 4.3.3). With the advent of deep neural networks (DNN) in speech recognition, we followed the approach in [45] and employed a hybrid DNN-HMM for acoustic modelling in the context of German broadcast speech recognition (Section 4.3.4). This is also the first configuration which employed the full GerTV corpus (TS IV, 992 hours). Note that we excluded some utterances from the training because they included mispronunciations, foreign language or dialect, as indicated by the annotation markers. Results indicated a further improvement of the speech recognition system by using hybrid DNN-HMM. The hybrid DNN-HMM system was again improved by using recurrent neural network language model n-best hypothesis rescoring (Section 4.3.5) by following the approach proposed in [96]. By optimising the non-linearities of the DNN, as proposed in [102], and with the employment of p-norm non-linearities, we could again improve the German broadcast speech recognition system (Section 4.3.6). By following the RNN approach based on LSTM units [48], which are able to exploit the temporal information without the use of HMMs more efficiently, we were able to further improve the German broadcast system (Section 4.3.7). To reduce the training time of RNNs and to improve the accuracy, the authors in [107] proposed an acoustic model for speech recognition which is based on Time Delay Neural Networks (TDNN) with subsampling. Following the approach we were able to improve the accuracy of the German broadcast speech recognition once again (Section 4.3.8). In [109] the author combined the TDNN acoustic model with projected LSTM (LSTMP) recurrent layers. We adapted this approach in the context of German broadcast speech recognition and were able to improve the system (Section 4.3.9). Finally, Gated Convolutional Neural Networks (GCNN) became available for language modelling, which we used for rescoring the hypotheses derived TDNN-LSTMP configuration to improve the system even more (Section 4.3.10).

In the course of this thesis (2012-2018), we were able to improve the performance of the German broadcast speech recognition by a large margin by carrying out numerous experiments and by optimising the necessary parameters for our setup. We reduced the WER of the German broadcast speech recognition system for the clean DiSCo test dataset for planned speech from 26.4 % to 8.1 % WER, which is an improvement of 18.3 % WER absolute or 68.2 % relative. In the case of the clean DiSCo test dataset for spontaneous speech, we were able to improve the system from 33.5 % to 9.3 %, which is an improvement of 24.2 % absolute or 72.3 % relative.

Configuration	**Size** (hours)	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spont.
GMM-HMM [85]	105	30.2	26.4	33.5
HMM-GMM	322	29.6	24.0	31.1
HMM-GMM-SPSA	322	27.7	22.6	28.4
DNN-HMM	322	23.9	18.4	22.6
SGMM-HMM	322	23.5	18.1	22.5
SGMM-HMM	636	23.3	18.1	22.4
DNN-HMM	636	22.7	17.4	21.5
DNN-HMM	992	21.3	15.5	19.7
DNN-HMM-RNN	992	20.0	15.3	18.4
p-norm DNN-HMM	992	18.8	13.3	16.5
RNN	992	17.2	11.9	14.5
TDNN	3x992	15.6	11.1	13.2
TDNN-LSTMP	3x992	13.7	8.9	10.3
TDNN-LSTMP-GCNN	3x992	12.7	8.1	9.3

Table 4.16: Performance of different configurations of the Fraunhofer IAIS speech recognition system

Chapter 5

Gradient-Free Speech Recognition Decoder Parameter Optimisation

As noted in the previous chapters, the speech recognition system (Chapter 4) is the key component of the Fraunhofer IAIS audio mining system, which enables the efficient search of large heterogeneous media archives. In order to be competitive amongst other competitors on the market, this system has to be constantly developed by employing the latest state-of-the-art methods provided by the research community. Also, the system has to be optimised for the employment in the German broadcast speech domain to be successful. Both the optimisation of the acoustic model and the language model in automatic speech recognition are well-established task. The actual decoding process, however, also uses a large set of free parameters that should be optimised for the given task. Some decoding parameters directly weight the models, others affect the size of the search space, where it is hard to foresee the effect on the hypothesis quality and on the decoding time. In praxis, these parameters are often left to default values provided by the toolkit manuals, or set empirically in a rather time consuming task. Both strategies are usually of limited success and end up with a parameter set that is far away from the optimum in terms of performance and decoding speed, without unlocking the full potential of the ASR system. Automatic decoder parameter optimisation algorithms solve this problem by performing a gradient approximation to optimise the parameters. Gradient approximation methods, i.e. gradient-free methods, rely on measurements of the objective function (e.g. WER), not on measurements of the gradient of the objective function and are usually employed when the measurement of the gradient of the objective function is complicated or even impossible, as it is for speech recognition. State-of-the-art decoder parameter optimisation methods usually need a large number of iterations and hence, are slow. This is why we introduce a method for automatically solving multivariate optimisation problems, namely the simultaneous perturbation stochastic approximation (SPSA) algorithm, which was developed in [116] in a general mathematical context, and which has not been employed in the context of

ASR decoder parameter optimisation before, to ASR decoder parameter optimisation.

In Section 3.4.7 we already discussed the related work of automatic parameter optimisation in the field of natural language processing, especially speech recognition. In Section 5.1 we employ the SPSA algorithm for optimising the parameters of the decoder of the German broadcast speech recognition system in an unconstrained setting, i.e. we directly optimise the accuracy of the system in terms of word error rate (WER). In Section 5.2 we extend the SPSA algorithm for optimising the accuracy of the system by constraining the decoding speed in terms of real time factor (RTF). In Section 5.3 we compare the developed methods to state-of-the-art methods with different speech recognition architectures. Section 5.4 summarises this chapter and the achieved contributions.

5.1 Unconstrained Decoder Parameter Optimisation

In this section we employ the simultaneous perturbation stochastic approximation (SPSA) [116] algorithm for automatically optimising the decoder parameters of the speech recognition systems, which we developed in Chapter 4. In the unconstrained setting, the algorithm tries to optimise the values to achieve the best accuracy of the system in terms of word error rate (WER).

In Section 5.1.1 we describe the SPSA algorithm. In Section 5.1.2 we discuss the parameters of the speech recognition decoder which we are trying to optimise. In Section 5.1.3 we employ and evaluate the algorithm in the context of speech recognition and discuss the results.

5.1.1 Simultaneous Perturbation Stochastic Approximation

The simultaneous perturbation stochastic approximation (SPSA) algorithm was introduced in [116] and fully analysed in [88] in a general mathematical context. A simple step-by-step-guide for the implementation of the SPSA algorithm in generic optimisation problems was proposed in [117], where the author also offers some practical suggestions for choosing certain algorithm coefficients.

The SPSA algorithm minimises a loss function $L(\cdot)$ for the optimisation of a p-tuple of free parameters θ as follows:

Let $\hat{\theta}_k$ denote the estimate for θ in the k-th iteration. Then, for a gain sequence denoted as a_k, and an estimate of the gradient of the loss function at a certain position denoted as $\hat{g}_k(\cdot)$, the algorithm has the form:

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}\left(\hat{\theta}_k\right) \tag{5.1}$$

In order to estimate $\hat{g}_k(\cdot)$, we perturbate each $\hat{\theta}_k$ with a vector of length p containing mutually independent, zero-mean random variables Δ_k, multiplied by a positive scalar c_k, to obtain two new parameter tuples:

$$\hat{\theta}_k^+ = \hat{\theta}_k + c_k\Delta_k \tag{5.2}$$
$$\hat{\theta}_k^- = \hat{\theta}_k - c_k\Delta_k \tag{5.3}$$

For a loss function $L(\cdot)$, we then estimate $\hat{g}\left(\hat{\theta}_k\right)$ as:

$$\hat{g}\left(\hat{\theta}_k\right) = \frac{L\left(\hat{\theta}_k^+\right) - L\left(\hat{\theta}_k^-\right)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix} \tag{5.4}$$

We follow the implementation suggestions in [117] and use a ± 1 Bernoulli distribution for Δ_k, and further set:

$$a_k = \frac{a}{(A+k+1)^\alpha} \quad \text{with } a = 2,\ A = 8,\ \alpha = 0.602 \tag{5.5}$$
$$c_k = \frac{c}{(k+1)^\gamma} \quad \text{with } c = 0.25,\ \gamma = 0.101 \tag{5.6}$$

Using these gain sequences a_k and c_k, SPSA normally converges in a similar number of iterations as the classical Kiefer-Wolfowitz FDSA [68], but requires p times fewer measurements of the loss function, as the decoding performance has to be evaluated only 2 times in each iteration, as shown in [88].

α and γ are set to the lowest values satisfying the theoretical conditions for convergence, as proposed in [88], leading to larger step sizes in the iteration process and therefore faster performance.

Since the measurements of $L(\theta)$ do not contain noise, we choose c to be a small positive number as proposed in [117]. Likewise, we set $A = 8$ to approximately 10 % of the expected iterations and finally $a = 2$ so that the steps in the update have a reasonable size for the first iterations. During our experiments, we did not experience a high sensitivity of SPSA for any of its hyperparameters, only the required number of iterations to reach a stable result changes.

The speech recognition decoder parameters θ which we optimised are discussed in the following section (Section 5.1.2).

5.1.2 GMM-HMM Decoder Parameters

In this section we describe and discuss the speech recognition decoder parameters that have been automatically optimised. The corresponding decoder parameters of the GMM-HMM approach, as implemented by the speech recognition decoder Julius [62], are:

- m-gram language model weight on 1^{st} pass

- Grammar word insertion penalty for the 1^{st} pass
- Beam width for rank beaming on the 1^{st} pass. This value defines the search width on the 1^{st} pass.
- m-gram language model weight on 2^{nd} pass
- Grammar word insertion penalty for the 2^{nd} pass
- Beam width for rank beaming on the 2^{nd} pass. This value defines the search width on the 2^{nd} pass.
- Score envelope width for enveloped scoring
- Stack size, i.e. the maximum number of hypotheses that can be stored on the stack during search.
- Number of expanded hypotheses required to discontinue the search
- Number of sentence hypotheses to be output at the end of the search

Note that Julius speech recognition decoder is based on a two-pass strategy, which is implemented for efficiency reasons. A 2-gram and a 3-gram language model is used on the respective passes. The first pass of the search efficiently reduces the graph to a subgraph consisting of the top n-best paths. The second pass seeks out for the optimal path among those remaining in the subgraph with higher accuracy.

In Table 5.1 the free parameters of the decoding process are listed again. Also, the starting value of the optimisation (i.e. the default values provided by the toolkit), as well as reasonable minimal and maximum values are listed, which are used as upper and lower boundaries of the parameters during optimisation. Some parameters are given individually to the 1^{st} pass or 2^{nd} pass of the Julius decoder, and are marked with (2). Continuous parameters are marked by a trailing ".0".

Parameter	**Start**	**Minimum**	**Maximum**
(2) LM weight	10.0	0.0	20.0
(2) Ins. penalty	-7.0/10.0	-20.0	20.0
(2) Beam width	1,500/250	700/20	3,000/1,000
Score envelope	80.0	50.0	150.0
Stack size	10,000	500	20,000
# expanded hyp.	20,000	2,000	20,000
# sentence hyp.	10	5	1,000

Table 5.1: Free parameters of the decoding process

5.1.3 Experimental Setup and Evaluation

For the experiments we used the best speech recognition system which was available at the timepoint of the experiments, namely the extended HMM-GMM ASR system (Section 4.3.2), which was trained on 322 hours of audio material taken from the GerTV corpus (Section 4.3.1).

We employed the development dataset from the GerTV corpus (Section 4.3.1) for the approximation of the gradient during the optimisation process. The full list of parameters that were optimised is described in Section 5.1.2. Also, the default values of the decoder parameters as well as the minimum value and the maximum value of each parameter, which spans the parameter range that we allow for optimisation, are described in Table 5.1.

Internally, we map the parameter ranges to $[-15 \cdots +15]$ for the SPSA iterations. If the parameters are integers, we store them as floating values internally but truncate them for each evaluation of the loss function.

We optimise the parameters to maximise the accuracy of the speech recognition system in terms of the word error rate (WER), i.e., the number of substitutions, insertions and deletion errors divided by the reference length, using it directly as the loss function (or rather the objective function) in the SPSA algorithm in the unconstrained setting.

The results on the development set are shown in Figure 5.1. The progression of the optimisation over the iterations is visible in this graph. Also, the WER is depicted, for $L(\theta^+)$ and $L(\theta^-)$ for every iteration, which represent the simultaneously perturbated parameter values from which the gradient is approximated in the SPSA algorithm. The algorithm converges after some iterations. In each iteration, the decoding has to take place for the parameters θ^+ and θ^-, and for the optimised parameters after calculating the gradient θ. Hence, a total of three decodings is necessary for each iteration. In total, the hypothesis quality improved by 1.9 % WER absolute (6.4 % relative) during 18 iterations. In a second run, the improvement was similar and it converged after 10 iterations already.

To evaluate whether the performance gain by using the optimised parameters also generalises to unseen data, we evaluate the ASR configurations with the optimised parameters on the clean datasets of the DiSCo corpus (Section 4.2.2, planned speech and spontaneous speech) and the LinkedTV datasets (Section 4.2.3). The results are summarised in Table 5.2.

The results on the test sets are presented in Figure 5.2 for the DiSCo corpora and in Figure 5.3 for the LinkedTV corpora. It can be seen that the optimisation generalises nicely on all four test corpora: 1.2 % WER absolute improvement on the planned speech task, and 2.7 % WER absolute improvement on the spontaneous speech task for the DiSCo datasets (see Figure 5.2).

The same is true for the LinkedTV data, with even higher improvements of 1.9 % and 4.4 % WER respectively. While the resulting error rates for the two "planned" corpora are comparable, the LinkedTV spontaneous dataset performs a lot worse. This can likely be explained by the fact that LinkedTV spontaneous contains partly dialectal

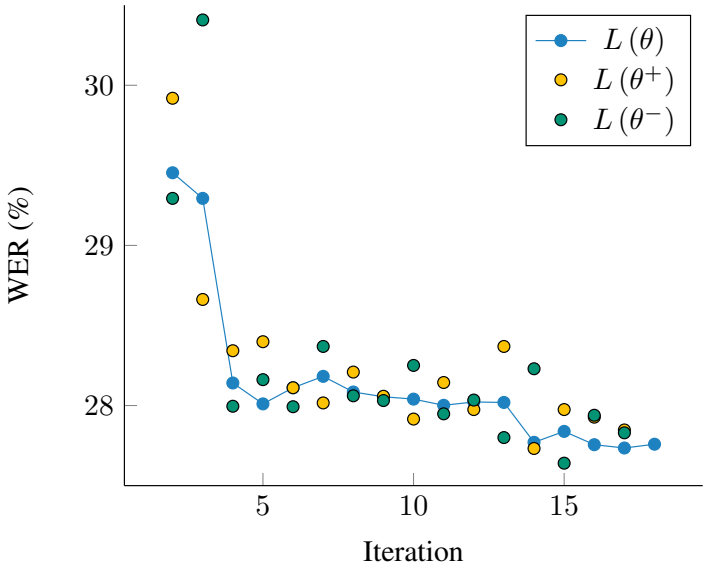


Figure 5.1: First example run of the SPSA and its word error rate progression on the development corpus

Configuration	**WER** dev	**WER** DiSCo planned	**WER** DiSCo spontaneous	**WER** LinkedTV planned	**WER** LinkedTV spontaneous
GMM-HMM	29.6	24.0	31.1	26.4	50.0
SPSA 1st run	27.7	22.8	28.4	24.6	45.7
SPSA 2nd run	27.7	22.6	28.4	24.5	45.6

Table 5.2: WER results on several corpora for two SPSA runs and comparison to the configuration without SPSA optimisation

speech, while DiSCo spontaneous is purely standard German.

In this section we have introduced the SPSA algorithm, which was presented in [116] in a general mathematical context, and which has not been used for the task of speech recognition decoder parameter optimisation before, to the task of speech recognition decoder parameter optimisation. We have shown that it is an efficient method to optimise the free parameters of a speech recognition system. Using the WER as the loss function directly, the optimisation converges after less than 20 iterations, leading to an WER improvement of about 2 % absolute for the employed speech recognition system. In each iteration only three decodings are needed (parameters θ_k^-,θ_k^+ and θ_k). The resulting settings generalise well to different datasets, i.e., both planned and spontaneous speech. Hence, SPSA can be a fast and decent choice for speech recognition decoder parameter optimisation.

However, the improvements come with a cost. In Figure 5.4 the evolution of the

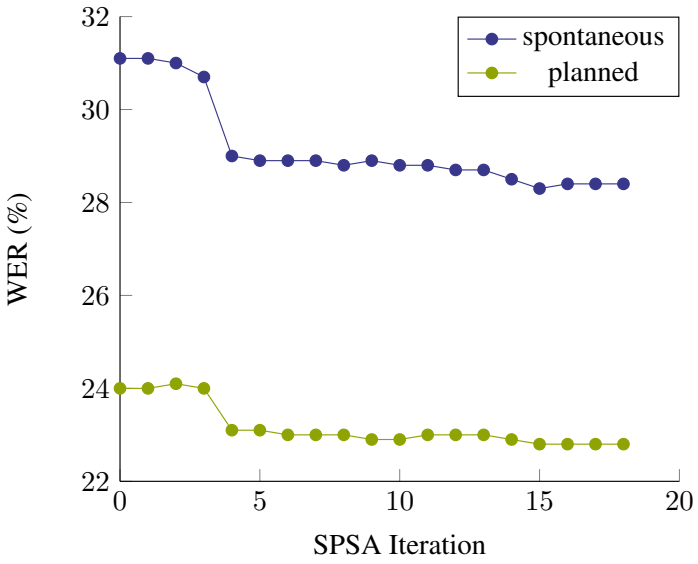


Figure 5.2: WER progression on DiSCo corpora. WER results on planned and spontaneous data, showing the first run of SPSA with 18 iterations. Iteration 0 denotes the employed speech recognition configuration without SPSA parameter optimisation.

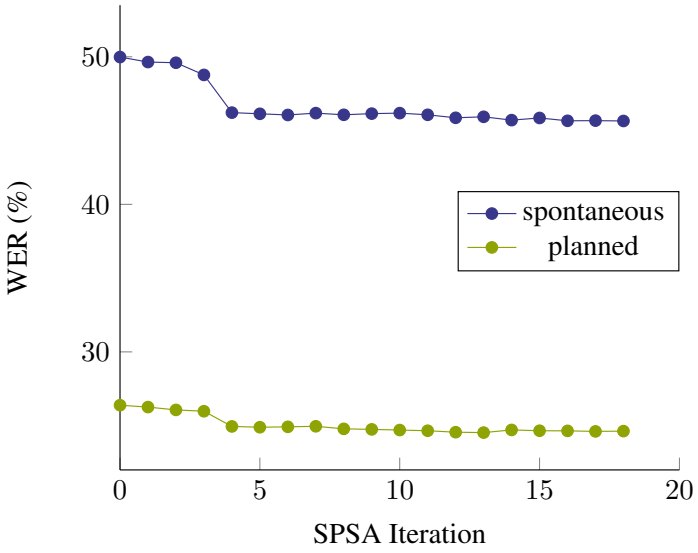


Figure 5.3: WER progression on LinkedTV corpora. WER results on planned and spontaneous data, showing the first run of SPSA with 18 iterations. Iteration 0 denotes the employed speech recognition configuration without SPSA parameter optimisation.

real-time factor (RTF, Section 3.4.9) for the first optimisation run is depicted. It can be seen, that the decoding time (expressed by the RTF) is increased over the iterations.

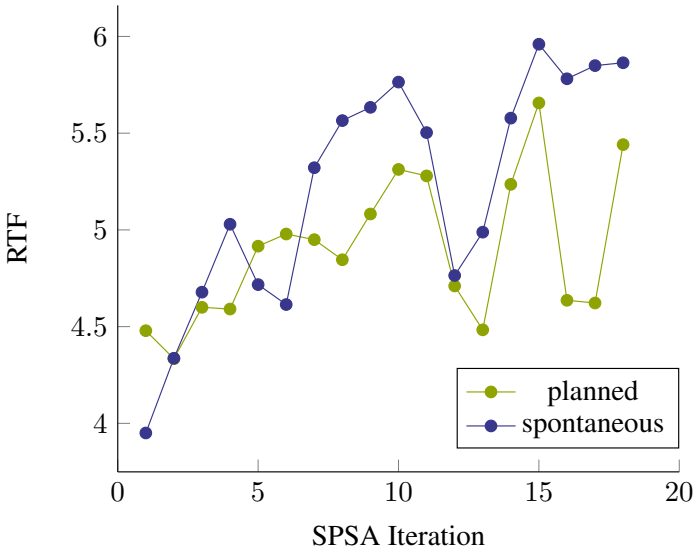


Figure 5.4: RTF development on the DiSCo corpora "clean planned" and "clean spontaneous", for the first optimisation run using the unconstrained optimisation criterion.

This is often the case when optimising the performance, because often the complexity of the algorithm is increased (e.g. by making the decoding graph larger). While for many settings this might not pose a problem, in time-crucial applications, e.g. in productive audio mining systems, this is not desirable. Thus, in the following section (Section 5.2), we take the RTF into account and optimise WER and RTF jointly.

5.2 Time-constrained Decoder Parameter Optimisation

We have shown in the experiments of the previous section (Section 5.1), where we employed the SPSA algorithm to optimise the accuracy of the speech recognition system by using the word error rate directly as the objective function (i.e., the loss or cost function), that the algorithm is capable to optimise the performance of a speech recognition system with a rather small number of iterations and decodings of the development set. However, the improvements in the performance come partially at the expense of the decoding speed in terms of RTF (as shown in Figure 5.4 and increased memory consumption, because some changes in the parameters increase the complexity of the algorithm. Often this is not a problem when we aim for the best performance of the speech recognition system, but in time-crucial applications like in the framework of an audio mining system this is not desirable. Thus, in this section we try to take the RTF into account and optimise WER and RTF jointly in another set of experiments.

In Section 5.2.1 we introduce the RTF penalty term to the general loss function of the SPSA algorithm. In the subsequent sections we define RTF penalty terms for the use of constrained optimisation, namely the exponential penalty (Section 5.2.2, the

delta penalty (Section 5.2.3), and the increasing penalty (Section 5.2.4) and evaluate the methods experimentally. In Section 5.2.5 we compare the RTF constrained configurations to the unconstrained and the baseline configuration without SPSA optimisation.

5.2.1 Time-constrained Word Error Rate Optimisation

As we want to optimise the word error rate by also taking into account the decoding speech in terms of the RTF, we constrain the optimisation. Therefore, we penalise the loss function by a RTF dependent penalty term μ:

$$L\left(\hat{\theta}_k\right) = \mathrm{WER}\left(\hat{\theta}_k\right) + \mu\left(\hat{\theta}_k\right) \tag{5.7}$$

In the following, we introduce several RTF dependent penalty terms for the constrained optimisation:

5.2.2 Exponential RTF penalty

Intuitively, we penalised the RTFs above a given threshold t exponentially:

$$\mu\left(\hat{\theta}_k\right) = \begin{cases} \exp\left(\mathrm{RTF}\left(\hat{\theta}_k\right) - t\right), & \text{for RTF}\left(\hat{\theta}_k\right) > t \\ 0, & \text{else} \end{cases} \tag{5.8}$$

However, this turned out to deteriorate the parameters too much when the initial RTF was already substantially higher than this given threshold t. This was especially a problem for optimisation on a slow machine, where the WER dropped 30 % absolute due to a severe gradient misjudgement in the first iteration. Due to the mentioned problems we faced with the exponential RTF penalty we leave it out from the evaluations.

5.2.3 Delta RTF penalty

Another possibility is to use the delta of the actual RTF and a given threshold t directly with:

$$\mu\left(\hat{\theta}_k\right) = \begin{cases} \mathrm{RTF}\left(\hat{\theta}_k\right) - t, & \text{for RTF}\left(\hat{\theta}_k\right) > t \\ 0, & \text{else} \end{cases} \tag{5.9}$$

which lead to an equilibrium of the RTF while optimising the WER (Figure 5.5).

This trend was also reproducible in a second optimisation run see, c.f. Table 5.3. Also, from this table it can be shown that by using the delta RTF the WER can be improved compared to the baseline configuration without SPSA optimisation while maintaining a similar RTF. In contrast to the unconstrained optimisation where the RTF increases in an undesirable way. As an outcome of the experiments, when comparing the unconstrained method to the constrained method with the delta penalty, the WER results slightly favor the delta penalty configuration, while the RTF results clearly favor the delta penalty configuration, which is the aim of constrained optimisation in this context.

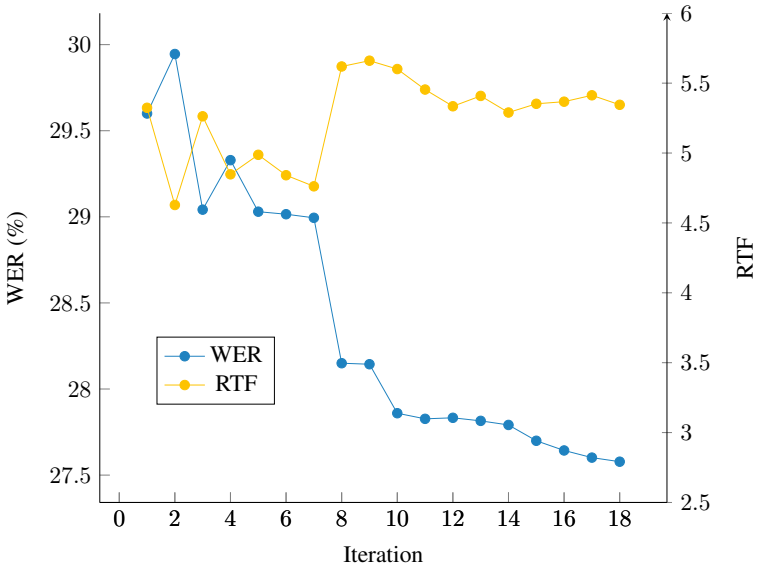


Figure 5.5: Optimisation run on the development set with delta RTF penalty (Equation 5.9), $t = 5.0$

loss function	#iter	dev			test planned			test spontaneous		
		WER	**RTF** 1.6GHz	**ΔRTF**	**WER**	**RTF** 2.6GHz	**ΔRTF**	**WER**	**RTF** 2.6GHz	**ΔRTF**
baseline	0	29.6	5.3	1.00	24.0	4.6	1.00	31.1	4.0	1.00
unconstrained	18	27.7	7.0	1.32	22.8	5.4	1.17	28.4	5.9	1.48
unconstrained	18	27.7	7.3	1.38	22.6	6.1	1.33	28.4	6.1	1.53
delta	18	27.6	5.3	1.00	22.2	4.5	0.98	27.7	4.8	1.20
delta	18	27.6	5.1	0.96	22.5	4.2	0.91	27.9	4.4	1.10
increasing	14	32.5	3.0	0.57	26.1	2.2	0.48	31.9	2.3	0.58
	+28	31.6	2.9	0.55	25.3	2.5	0.54	30.0	2.6	0.65
increasing	12	31.2	3.0	0.57	25.5	2.2	0.48	31.0	2.1	0.53
	+28	33.6	2.9	0.55	27.5	2.3	0.50	32.1	2.4	0.60

Table 5.3: WER and RTF results on all corpora, for the SPSA iterations and their respective loss functions. Each optimisation on a given loss function has been executed two times from scratch to check for convergence. The unconstrained runs (Section 5.1) use the WER directly as loss function, delta uses Equation 5.9 and increasing uses Equation 5.10

5.2.4 Increasing RTF penalty

In another setup we penalise the RTF increasingly with each iteration:

$$\mu\left(\hat{\theta}_k\right) = \left(\text{RTF}\left(\hat{\theta}_k\right) - t\right) \cdot \tilde{k}\,, \text{for RTF}\left(\hat{\theta}_k\right) > t \tag{5.10}$$

with an increasing $\tilde{k} = k$ as long as a RTF threshold t is not reached. For the first iteration where the RTF is equal or below to the threshold t, $\tilde{k}$ is fixed in order to give

the optimisation the ability to converge, thus stabilising the WER.

In our experiments, we arbitrarily set the RTF threshold to $t = 3$, which was reached in iteration 14 an 12, respectively (c.f. Table 5.3). After this, the WER decreased in the the first run another 0.9 % absolute on the development set which maintaining the desired RTF (see Figure 5.6), with the result generalising well to the unseen DiSCo planned and spontaneous test sets.

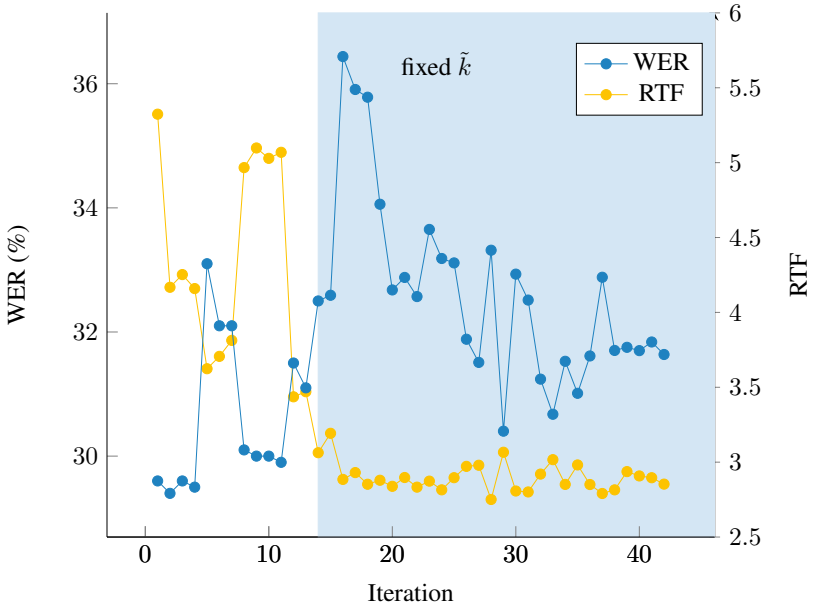


Figure 5.6: Optimisation run on the development set with increasing RTF penalty (Equation. 5.10)

In the second run, SPSA got stuck in a local optimum, leading to faster decoding but with lower quality.

5.2.5 Comparison of the RTF Penalty Functions

In order to see whether our optimisation is a reasonable trade-of between RTF and WER, we collected all results from the iterations and computed their convex hull on the DiSCo datasets (planned clean, Figure 5.7 and spontaneous clean, Figure 5.8).

It can be seen that the final SPSA iteration for each optimisation run is typically part of the convex hull or very near to its border. From our optimisation runs, we could see no gain for the RTF-unconstrained loss function. A delta RTF penalised loss function could result in a configuration that performs better in terms of WER and is generally faster. If the RTF is penalised increasingly in each step, the WER rate is still within reasonable range for a much better RTF. A detailed overview of the results is given in Table 5.3.

In this section we have shown that SPSA is an efficient means to optimise free

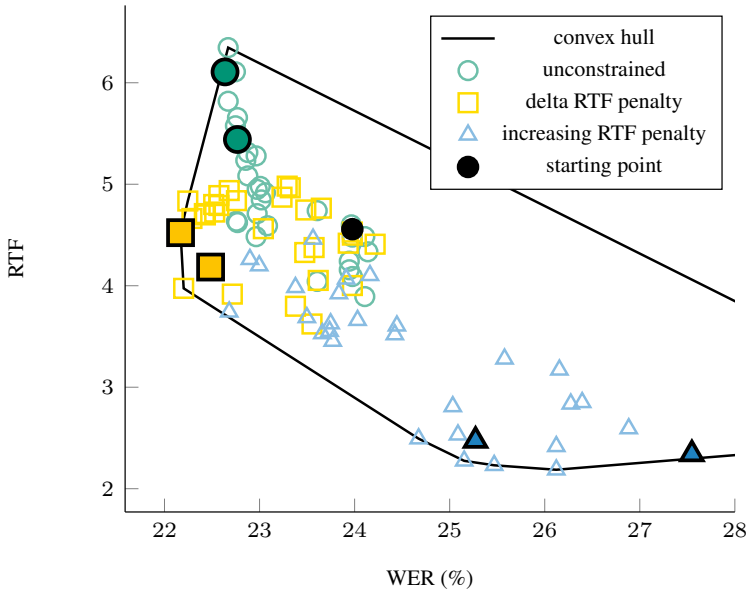


Figure 5.7: Results for DiSCo "planned clean". Scatter plot with all configurations, on the DiSCo test corpora. The final optimisation iteration is marked by filled-out symbols.

parameters of an ASR decoder in terms of both WER and RTF. In an unconstrained setting, the WER improves rapidly, but the RTF also increases in an undesirable way, as shown in Section 5.1. By adding the RTF to the loss function, one is able to stabilise the increase in time requirements. Overall, we have achieved an improvement of 1.8 % absolute WER on the DiSCo planned clean task and an improvement of 3.4 % absolute WER on the DiSCo spontaneous task, over the baseline speech recognition configuration without SPSA optimisation, while still having a reasonable RTF. When a specific RTF is required for an application scenario, using the "increasing" penalty function for SPSA allows reaching reasonable performance, given this constraint. The risk for local optima seems higher in this setting, though, which is why we would recommend multiple optimisation runs.

5.3 Comparison with State-of-the-art Methods

While there has been recent scientific contributions in the field of automatic speech recognition decoder parameter optimisation, no thorough comparison of the methods, in terms of convergence speed, decoding speed and performance has been undertaken. Hence, in this section we conduct a series of experiments with three different state-of-the-art decoding paradigms, which were available at the timepoint of the experiments, namely GMM-HMM, DNN-HMM and SGMM-HMM, and evaluate the performance of four different optimisation methods found in the literature including SPSA, both for

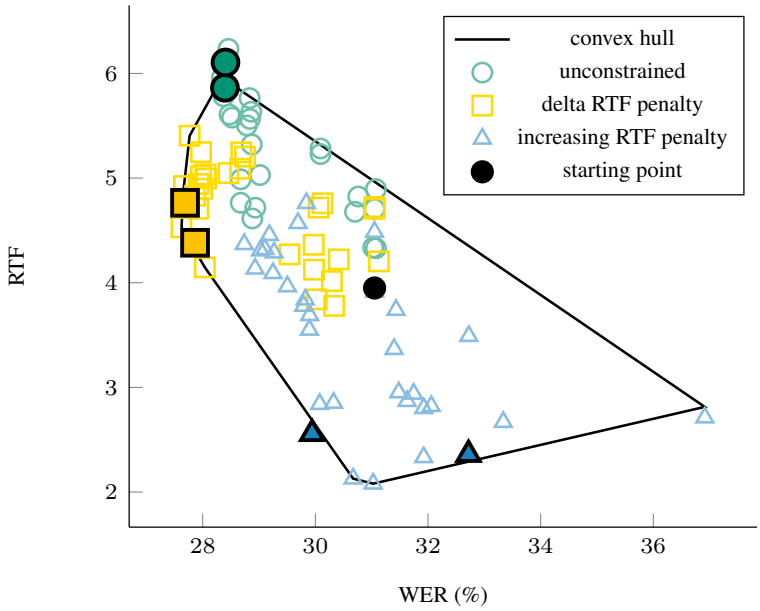


Figure 5.8: Results for DiSCo "spontaneous clean". Scatter plot with all configurations, on the DiSCo test corpora. The final optimisation iteration is marked by filled-out symbols.

unconstrained and time-constrained decoder optimisation and compare the results.

First, we introduced the state-of-the-art optimisation methods found in literature. In Section 5.3.1 the downhill-simplex method is described. The evolutional strategies algorithm is described in Section 5.3.2. In Section 5.3.3 the gradient descent method is described. We have already described the SPSA algorithm in both unconstrained and constrained settings in the previous sections (Section 5.1 and Section 5.2 respectively). Since the decoder parameters from the GMM-HMM decoder (Section 5.1.2), which was used in the previous experiments, differ significantly from the parameters of the DNN-HMM and SGMM-HMM decoding algorithms, we describe parameters the DNN-HMM and SGMM-HMM approach in Section 5.3.4.

5.3.1 Downhill Simplex

Downhill Simplex was introduced in [70], also known as Nelder-Mead method, performs searching in an n-dimensional space by repeatedly transforming a simplex, which is spanned by a collection of $n + 1$ vertices in the parameter search space. This results in $n + 1$ loss function calls for initialisation. In each iteration, the highest ranked (i.e. worst w.r.t. the optimisation criterion) vertex is optimised towards the center of gravity of the remaining vertices. The three standard transforming operations to do so are named reflection, expansion, and contraction, each requiring one loss function call, respectively. If neither operation improves over existing vertices, shrinkage towards the

lowest ("best") ranked point in the simplex is performed, an operation which requires n new loss function calls. One iteration can thus vary considerably in terms of time requirements and either consist of 1, 2, 3, or $n+3$ loss function calls. For implementation details, we use the following algorithmic values: 1.0 for reflection, 2.0 for expansion and 0.5 for contraction and shrinkage.

5.3.2 Evolutional Strategies

The optimisation technique was created in the early 1960s and further developed by [118]. Evolutional strategies have been employed for ASR optimisation in [64], where the experiments do not take the RTF into account. Evolutional strategies is a stochastic nonlinear optimisation algorithm which adopts the idea of natural selection from Darwin's theory. The process starts with a number of random parameter tuples, and in each iteration, "offspring" tuples are generated by recombining and mutating the parents. Only the "fittest" solutions can "survive" into next generation. The number of loss function calls in Evolutional Strategies heavily depends on the choice of the population size while the latter is computed from dimension size of the search space [119] (i.e. the number of parameters required by the speech decoder in our case). For convenience, we use the CMAES tool [120].

5.3.3 Gradient Descent

Gradient descent as employed by Hannani and Hain [66] aims at finding the optimal configuration by tracking the optimal curve describes the optimum WER at any RTF. To track the curve, a starting point is computed first by applying any unconstrained search method. A set of parameter candidates are generated from perturbation of the previous point's parameter set in order to estimate the gradient of the curve. Consequently, gradient of the curve and next parameter are chosen under the constraint of minimal cost. The process is repeated until RTF value is sufficiently small.

In contrast, gradient descent has been used for time-constrained decoding parameter optimisation but not in an unconstrained setting [66]. Neither of these papers compare the method to other optimisation methods. Downhill Simplex is a well-established method that has good convergence but is considered slow.

5.3.4 DNN-HMM and SGMM-HMM Decoder Parameters

In Table 5.4 the decoder parameters for the DNN-HMM and the SGMM-HMM approach are listed.

5.3.5 Time-Unconstrained Experiments

In this section we perform two sets of experiments. First, we compare three gradient-free optimisation techniques in terms of error rate improvement and the required number of loss function evaluations (#eval).

Parameter name	Start	Minimum	Maximum
Decoding beam	11.0	1.0	20.0
Lattice beam	8.0/6.0	1.0	20.0
Acoustic scaling	0.1	0.05	0.2
Maximum active states	7,000	1,000	15,000
Speaker vector beam	4.0	1.0	10.0

Table 5.4: Free parameters of the decoding process in the Kaldi toolkit. Continuous parameters are marked by a trailing .0. Speaker vector beam is exclusive for SGMM decoding. Lattice beam defaults to 8.0 for the DNN decoder and 6.0 for the SGMM decoder.

Here, we use the same GMM-HMM system as employed in the previous sections (Section 4.3.2) for a fair comparison. The GMM-HMM experiments conducted with Julius start with the same, manually set start configuration as in the previous sections, and which is specified in Table 5.1. The results on the development set and the four test sets are given in Table 5.5.

Method	#eval	WER Dev.	WER DiSCo planned	WER DiSCo spont.	WER LinkedTV planned	WER LinkedTV spont.
GMM-HMM baseline	1	29.6	24.0	31.1	26.4	50.0
GMM-HMM + Downhill Simplex	21	29.5	24.7	31.2	26.5	49.8
GMM-HMM + Downhill Simplex	95	27.4	22.2	27.5	24.2	45.0
GMM-HMM + SPSA	21	27.8	22.7	28.6	25.1	46.5
GMM-HMM + SPSA	40	27.7	22.6	28.4	24.5	45.6
GMM-HMM + Evolutional Strategies	21	28.2	23.1	28.8	25.2	46.9
GMM-HMM + Evolutional Strategies	91	27.7	21.6	26.8	24.5	44.9

Table 5.5: WER [%] results of ASR system configuration on various corpora.

In general, it can be seen that the error rate decreases considerably in all cases. After 21 loss function calls, SPSA has the best results in terms of word error rate (WER) when compared to downhill simplex and evolutional strategies. For SPSA, a first asymptotic behaviour can already be seen after 10 decoder runs – note that, for Julius, these numbers of decoder runs need to be employed just for spanning the starting simplex alone for the downhill simplex method. See Figure 5.9 for the word error rate development over time. At full convergence, however, SPSA does not recover from a local minimum and performs slightly worse than the others, probably due to a hasty decay in the perturbation and learning step size (c.f Table 5.5).

The reason why the number of evaluations should be reduced is because an evaluation of the parameters on a development set is time consuming, considering the size of the employed GerTV development set (Section 4.3.1, which is 3.5 hours of speech data in our case and the RTF which for example was in the range of approximately 1-6 in the previous experiments (Section 5.2). Hence, it can take several hours per evaluation, especially if the task is not parallelised, which is often the case, e.g. due to memory

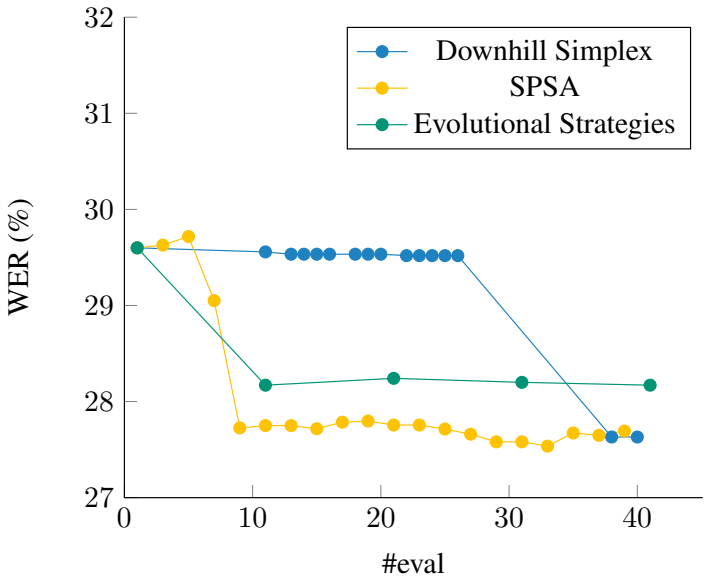


Figure 5.9: Comparison between Downhill Simplex, SPSA, Evolution Strategy after 41 Julius calls (#eval). Each dot represents one iteration.

limitations.

For the second set of the experiments we want to evaluate the unconstrained SPSA algorithm for different speech recognition configurations, which were considered state-of-the-art at the timepoint of the experiments, namely the GMM-HMM approach (Section 4.3.2) using Julius decoder [62], and the hybrid DNN-HMM (Section 4.3.4) and SGMM-HMM approach (Section 4.3.3) using the decoder from the Kaldi toolkit [55].

For the GMM-HMM experiments conducted with Julius we use the decoder parameters as described in Section 5.1.2 with the default values as specified in Table 5.1. The free parameters in the DNN and SGMM decoder of Kaldi are more limited in number (Section 5.3.4). For the baseline, we used the predefined values as given in Table 5.4.

The results are listed in Table 5.6. It can be seen that the performance of the decoder in terms of word error rate always improves for all the different speech recognition configurations when applying the SPSA algorithm. However, the improvements are smaller for the DNN-HMM and the SGMM-HMM approach implemented using the Kaldi Toolkit, where we also employed a smaller number of parameters for the decoding process. Nevertheless, both the decoder parameter optimisation and the employment of SPSA for decoder parameter optimisation has been proven to be beneficial for the more recently published approaches, namely the DNN-HMM and the SGMM-HMM approach.

In this section, we investigated several gradient-free optimisation techniques and their extensions for the optimisation of free speech decoding parameters in terms of word error rate. By comparing Downhill Simplex, SPSA and Evolutional Strategies

Method	**#eval**	**WER** Dev.	**WER** DiSCo planned	**WER** DiSCo spont.	**WER** LinkedTV planned	**WER** LinkedTV spont.
GMM-HMM	1	29.6	24.0	31.1	26.4	50.0
GMM-HMM + SPSA	40	27.7	22.6	28.4	24.5	45.6
DNN-HMM	1	23.9	18.4	22.6	21.0	37.8
DNN-HMM + SPSA	44	23.8	18.2	22.4	20.7	37.7
SGMM-HMM	1	23.5	18.1	22.5	20.8	36.6
SGMM-HMM + SPSA	34	23.0	17.6	22.0	20.5	36.4

Table 5.6: WER [%] results of different ASR paradigms with standard setting and SPSA adapted parameters.

on the same GMM-HMM decoding task, we came to the conclusion that SPSA is a fast method for convergence but more prone to local minima. SPSA is a universal optimisation technique, and we also showed improvements for other ASR decoding paradigms, namely DNN-HMMs and SGMM-HMMs.

5.3.6 Time-Constrained Experiments

While the experiments in the last section showed some nice improvements over the baseline performance in terms of word error rate, the quality came at a price of a slowed down decoding speed by roughly a factor of 2. If time is not critical, this behaviour can be ignored, although we showed in Section 5.1 that by simply adding the RTF onto the word error rate for a simple time-constrained optimisation, the WER improves comparably at no additional processing time.

However, this finding is unlikely to generalise when a substantial drop of the RTF is required. For the experiments conducted in this section, we penalise slow decoding runs in various ways in order to reach a reasonable trade-off between real-time factor and word error rate. Further, we compare our approaches with a gradient-descent based approach as proposed in [66]. Again, we conduct the most promising setting on multiple decoding paradigms.

We employ the following strategies to lower the RTF:

SPSA-increase, as introduced in Section 5.2.4, where we add the RTF penalty that is multiplied by the current iteration i, until a certain threshold t is reached, and fixed afterwards, and is expressed by (c.f. Equation 5.10):

$$\mu\left(\hat{\theta}_k\right) = \left(\mathrm{RTF}\left(\hat{\theta}_k\right) - t\right) \cdot \tilde{k}\,, \text{for RTF}\left(\hat{\theta}_k\right) > t \tag{5.11}$$

In **SPSA-adaptive**, we keep the RTF penalty more versatile (rather than fix it once), by incrementing it as long as two consecutive loss function calls exceed the RTF threshold, and decrementing it whenever two consecutive loss function calls are below the RTF threshold. It employes Equation 5.10, but with a more versatile $\tilde{k}_i$

computation: we increment $\tilde{k}$ if RTF exceeds the threshold and decrement otherwise.

$$\tilde{k}_i = \begin{cases} \tilde{k}_{i-1} + 1, & \text{if RTF } \left(\hat{\theta}_k\right) > t \\ 1, & \text{if } \tilde{k}_{i-1} = 1 \text{ and RTF } \left(\hat{\theta}_k\right) \leq t \\ \tilde{k}_{i-1} - 1, & \text{otherwise} \end{cases} \tag{5.12}$$

In our experiment with **gradient descent**, the starting point was obtained from the solution found by the unconstrained SPSA optimisation run as described above. The algorithm stops when it is unable to find any better tuple (i.e. a decoding run with lower RTF).

The results can be found in Table 5.7. In the experiments, the RTF threshold, which reflects the desired RTF, was set to a smaller RTF (60-80 %) compared to the unoptimised configuration. The desired RTF for the GMM-HMM was set to 3.0. The results indicate that the increasing RTF penalty (Figure 5.10) leads to considerably stabler results than the adaptive loss function (Figure 5.11). For the DNN-HMM approach we set the desired RTF threshold to 0.8 and for the SGMM-HMM approach to 1.5, and were able to obtain these factors with little WER degradation.

Decoding paradigm	**#eval**	**RTF**	**WER** Dev.	**WER** DiSCo planned	**WER** DiSCo spont.	**WER** LinkedTV planned	**WER** LinkedTV spont.
GMM-HMM	1	4.2	29.6	24.0	31.1	26.4	50.0
GMM-HMM + SPSA unconstrained	40	6.5	27.7	22.6	28.4	24.5	45.6
GMM-HMM + SPSA increasing	86	2.8	28.9	23.1	28.5	25.2	46.5
GMM-HMM + SPSA adaptive	86	3.0	29.1	23.0	28.1	25.6	46.3
GMM-HMM + SPSA + Gradient Descent	80	3.9	28.7	23.6	29.5	25.5	47.6
DNN-HMM	1	1.1	23.9	18.4	22.6	21.0	37.8
DNN-HMM + SPSA increasing	16	0.8	24.2	18.7	22.8	21.2	38.1
SGMM-HMM	1	2.5	23.5	18.1	22.5	20.8	36.6
SGMM-HMM + SPSA increasing	28	1.6	24.3	18.9	23.3	21.2	38.0

Table 5.7: WER [%] results of different ASR paradigms with standard setting and SPSA adapted parameters.

As for time-constrained optimisation where a certain RTF is desired, we compared various loss function strategies for SPSA and a gradient descent approach found in the literature. In our set of experiments, gradient descent was unable to reach the set RTF threshold. The SPSA extension using an increasing RTF penalty showed comparable behaviour amongst the decoder paradigms.

5.4 Summary and Contributions

In this chapter we approached the fast, robust and efficient speech recognition decoder parameters optimisation. ASR decoder parameter optimisation is necessary to unlock the full potential of a speech recognition system by tuning the parameters to suffice certain requirements. These can be either to provide the optimal parameter for best

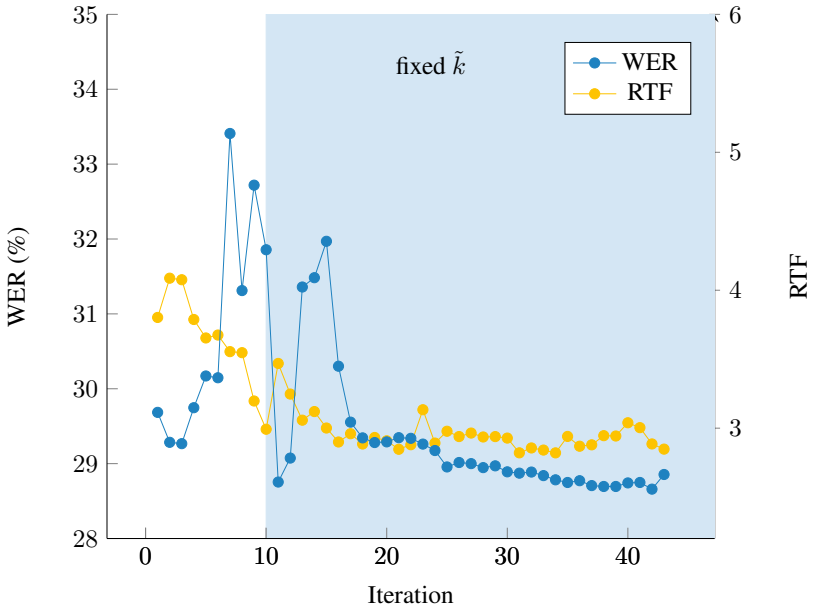


Figure 5.10: Increasing RTF penalty. Optimisation runs on the development set, with different RTF-penalised loss functions.

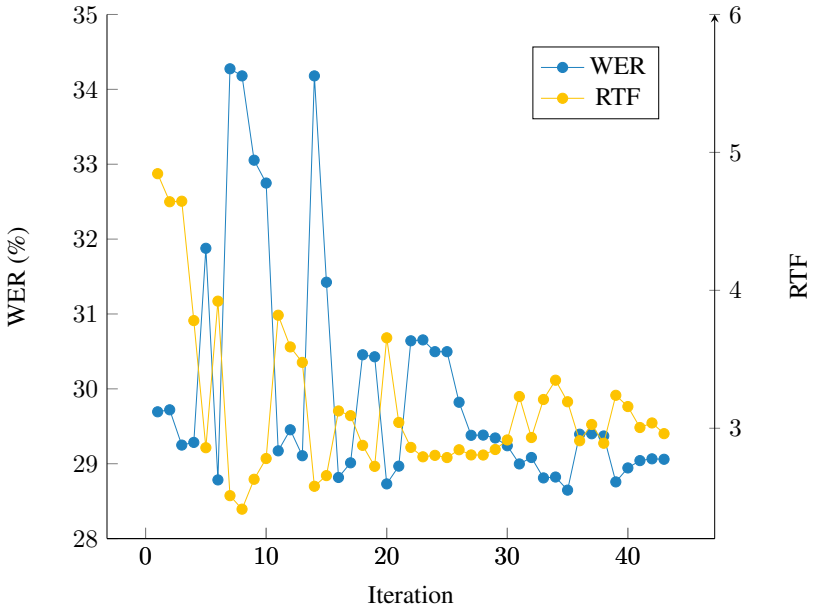


Figure 5.11: Adaptive RTF penalty. Optimisation runs on the development set, with different RTF-penalised loss functions.

performance in terms of word-error-rate (WER), or to provide the optimal parameters for the best performance in a different domain (e.g. telephone data, background car noise), or to provide a speech recognition system that is fast in terms of real-time-factor (RTF) and accurate as well. Ideally, the optimisation procedure should be fast. This is why we introduced the SPSA algorithm to the task of speech recognition decoder parameter optimisation in this chapter. This algorithm, which was invented in [116] and further developed in [88] in a general mathematical context, has not been employed to the task of ASR decoder parameter optimisation before. In Section 5.1 we employed the algorithm to optimise the performance expressed in WER in an unconstrained setting. We experienced, that the decoding time in terms of real-time-factor (RTF) of the optimised configuration is getting slower due to the increased complexity of the resulting configuration. Hence, in Section 5.2 we extend the SPSA algorithm by introducing a RTF penalty in the loss function, with the intention of getting a fast and precise speech recognition system. By introducing different RTF penalty terms we were able to get speech recognition system that are both fast and precise. In Section 5.3 we compare the SPSA algorithm for both constrained and unconstrained setting to state-of-the-art methods, and show that the SPSA algorithm is faster in convergence (in terms of the number of evaluations of the development set) and provides comparable results, due to the simultaneous perturbation of all parameters at a time. We end up with a method that is fast in convergence, easy to deploy (by using the implementation guide from [117]) and which allows to receive fast speech recognition systems with high accuracy.

Chapter 6

Dialects in Speech Recognition

Speech recognition systems which are trained on the standard language, like the standard German broadcast speech recognition system which we developed and optimised in the previous chapters and which is employed in the Fraunhofer IAIS audio mining system, nowadays show excellent performance on speech data which has characteristics similar to the training data. However, if a mismatch between the training data and the testing data is present, these systems usually show degraded performance. Mismatches include background noise, reverberation and the presence of dialectal speech. Mismatches regarding dialectal speech compared to standard language speech include mismatches in vocabulary, syntax, semantics, phonetics and prosody. Germany and other German speaking countries such as Austria and Switzerland have a broad variety of regional dialects. Each dialect has its own differences compared to the standard language or compared to other dialects. In speech recognition, these differences most often cause decoding errors and hence degraded performance. One way to cope with the manifold of dialects in speech recognition is to identify the dialect in advance and then use dialectal speech recognition models to decode the data [121]. In order to provide excellent speech recognition performance in the audio mining system which we aim to deploy for regional broadcasters, we want to adopt this approach. Therefore in this chapter we develop a German dialect identification system, which is able to discriminate between the standard language and various regional German dialects, and aim to train optimised dialectal speech recognition models to decode the speech data. The related work concerning dialect identification and related measures has already been discussed in Section 3.5. In Section 6.1 we discuss the manifold of German dialects. In Section 6.2 we develop and optimise a German dialect identification system. In Section 6.3 we develop dialectal speech recognition models.

6.1 German Dialects

In Germany, Switzerland and Austria, German is among the official languages and the first language of the majority of the population. These regions also show a large variety of German dialects. The manifold of German dialects are depicted in

Figure 6.1. According to this map the German dialects can be grouped into Low German ("Niederdeutsch"), Middle German ("Mitteldeutsch") and Upper German ("Oberdeutsch") dialects. The Low German dialects may further be divided into West Low German ("Westniederdeutsch") and East Low German ("Ostniederdeutsch") dialects. Similarly, the Middle German dialects can be divided into West Middle German ("Westmitteldeutsch") and East Middle German ("Ostmitteldeutsch") dialects.

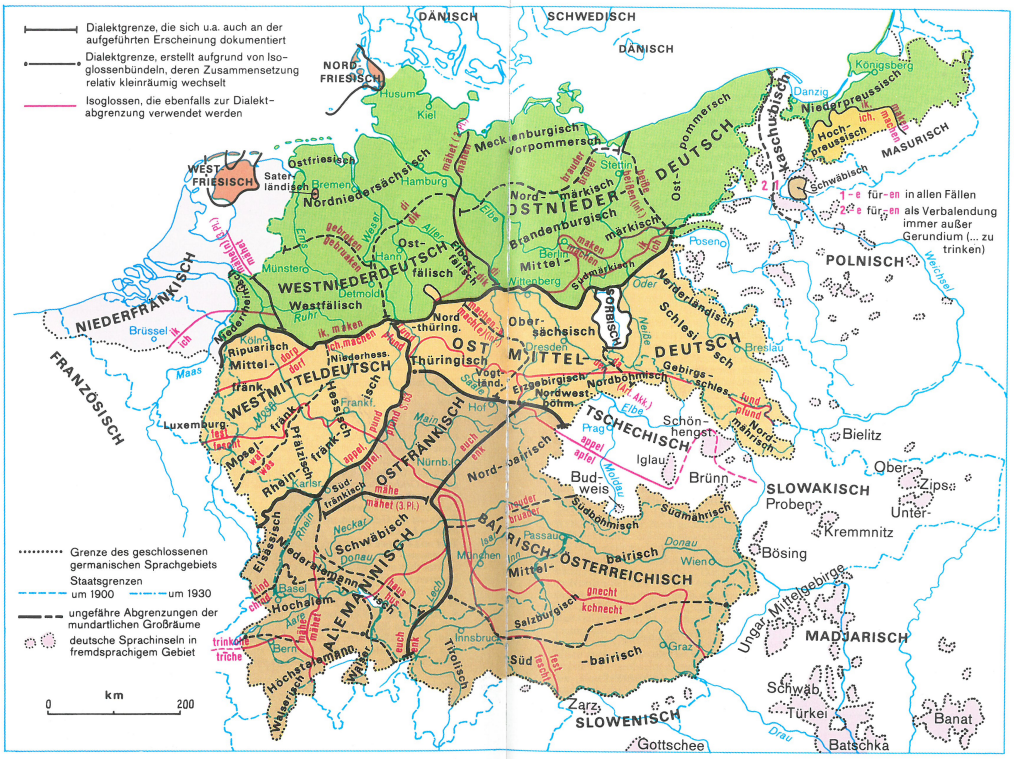


Figure 6.1: The structure of the Central European dialects of Germanic descent [122] (Status 1900)

Upper German dialects can be divided into East Franconian ("Ostfränkisch"), Bavarian ("Bairisch") and Alemannic ("Alemannisch") dialects. Alemannic dialects include the Swabian dialect ("Schwäbisch"), and Bavarian dialects include Northern Bavarian ("Nordbairisch"), Middle Bavarian ("Mittelbairisch") and Southern Bavarian ("Südbairisch"). While linguists distinguish between standard language, colloquial speech and dialects, the categorisation to these classes is very difficult due to the large variability. Especially in the south of Germany linguists assume a continuum from dialect to standard language for the phonetic area [122]. The colloquial languages are certainly the most widely used language form, the old dialects are losing more and more speakers and domains, they are superseded in favor of more spacious oriented language forms, the colloquial languages [122].

6.2 German Dialect Identification

In this section the development and optimisation of a German dialect identification system for the employment in the German broadcast domain are reported. The dialect identification system is intended to derive the dialect of a speaker in advance, to choose the optimal speech recognition models for decoding. It is also intended to enrich the retrieved information, so that more specialised search queries are possible, which facilitates the search, e.g. for journalists.

6.2.1 German Dialect Identification Based on the RVG1 Database

In this section we describe the development of a German dialect identification system for the broadcast domain. We first identified existing German dialect corpora, that can be used for this purpose. We found out that annotated dialectal speech resources are very rare. The only German speech corpus that covers the utterances from speakers of several German speaking regions including Austria and Switzerland is the BAS-RVG1 (Bayerisches Archiv für Sprachsignale - Regional variants of German) corpus [123]. It covers 85 utterances containing digits, connected digits, telephone numbers, phonetically balanced sentences, computer command and 1 minute of spontaneous monolog per speaker (498 total speakers). The speakers were recorded in an office environment with 4 microphones. The utterances were read from a screen. The speakers are grouped according to the regional origin into the regions Low Franconian, Western Low German, Eastern Low German, Western Central German, Eastern Central German, Alemannic, East Franconian, South Franconian and Bavarian/Austrian.

We used the monologue sentences of the speakers and the region labels to train a dialect identification system based on phone recognition followed by language model (PRLM, Section 3.5.1, [78]). Therefore we employed an open-loop phoneme recogniser on the corpus data and trained a m-gram language model on the recognised phoneme strings for each region. The open-loop phoneme recogniser is based on the standard German broadcast speech recogniser based on DNN-HMM trained on 636 hours of speech data as described in Section 4.3.4 and which was the best configuration at the time of the experiments. However, instead of a word-level language model, a phoneme language model was employed, which covers the full set of the phonemes with equal probabilities, so that the phoneme decoder outputs the likeliest phoneme sequence according to the audio data without any knowledge of the underlying language. The decoded phoneme sequences are used to train a m-gram language model for each region. 10 % of the data was withheld as test data in advance. For each sentence in the test set, we calculate the perplexity (c.f. Section 3.4.9) of each regional language model and hypothesise the region for which the perplexity is minimal. Table 6.1 shows the accuracy of the system for different m-gram language model orders. The best accuracy of 19.2 % can be reached with a language model order of 4. Since we have 9 dialectal regions and an equal distribution of regions in the test set, a random choice would have a precision of 1/9=11.1 %. Therefore this approach is well above chance level, though the accuracy is still far from satisfactory.

m-gram order	Accuracy [%]
1	8.3
2	14.3
3	16.5
4	19.2
5	18.2
6	18.0
7	17.2

Table 6.1: Dialect identification accuracy based on PRLM [78] on BAS-RVG1 corpus [123] using language models of different orders

After listening to the monologues we found out that many speakers, although originating from several places across the German dialect regions, do not have a noticeable dialect. We also derived from the corpus statistics that the majority of the speakers (259) were students with an average age of 24.6 years. The average age of all speakers is 29.5 years. 81 speakers self diagnosed their language as standard German. We assume that the self diagnosed language/dialect labels would provide better labels than the assignment of a speaker to a dialect region by the region of origin. We also assume that a dialect identification system trained on unprocessed microphone data would show degraded performance in the broadcast domain, where the signals are typically post-processed using limiters, compressors, expanders and spacialisation effects. Also the imbalanced age distribution might pose a problem in the broadcast domain. This is why we follow a different approach in the following section.

6.2.2 Upper German Broadcast Dialectal Database

We intensified the cooperation with a regional broadcaster, namely the *Bayerischer Rundfunk* (BR). BR generously provided us with a set of 302 broadcast media files (146 hours) with an average file length of 29.0 minutes for research purposes. The data contains mostly regional programmes from Bavaria, and hence it covers a large number of dialectal and standard German speakers. The data has been clustered in advance into the dialects Bavarian, Swabian and (East-)Franconian by the BR. However, a refined annotation of the speakers and their dialect as well as the time boundaries of the corresponding speech segments was necessary, which we performed manually using the annotation tool ELAN [124]. An example annotation is using ELAN tool is depicted in Figure 6.2.

The gender of the speakers (male or female) was annotated. The annotation of the speaker name, if available, was performed as well in the form "<firstname> <name>". In case no speaker name was available we annotated the speaker name as "<gender> <consecutive number>", so that within a media file the speaker name is unique. We annotated the dialects Bavarian, Franconian, and Swabian, as well as German standard language on a segment level. We only annotated segments where a single speaker is

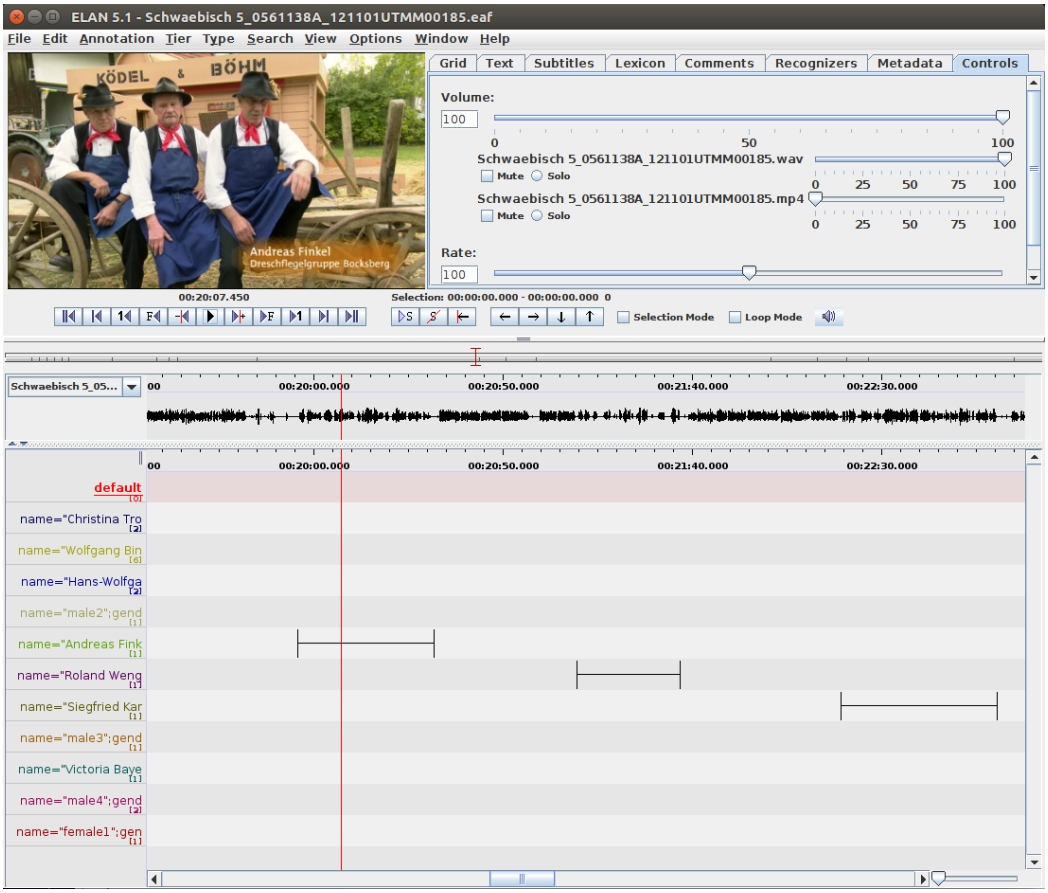


Figure 6.2: Example of an ELAN annotation for the task of dialect identification

present. The underlying text of the spoken utterances has not been annotated. We did not do this, because on the one hand it is a very time consuming process, and on the other hand because there is no standardised way to annotate dialects other than standard German. Also, it is not necessary for the task of dialect identification and dialect detection.

So far we have fine-annotated 52 media files with a total of 2,710 speech segments from 398 speakers (282 male, 116 female) in an ongoing process with a total segment size of 11.8 hours. An average of 106.5 seconds and an average of 6.8 segments have been annotated per speaker in this dataset. The average length of a segment is 15.6 seconds with a rather high standard deviation of 12.6 seconds, which is not unusual for real-world data from the broadcast domain. 75 standard German speakers, 149 Bavarian dialect speakers, 89 Swabian dialect speakers and 85 Franconian speakers have been annotated. We converted the audio data into RIFF/WAVE format (16 kHz sampling rate, 16 bits per sample, 1 channel).

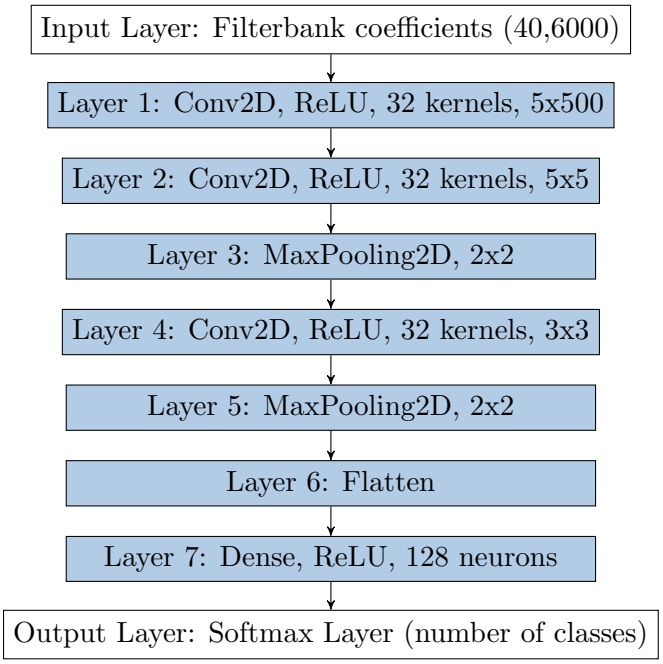


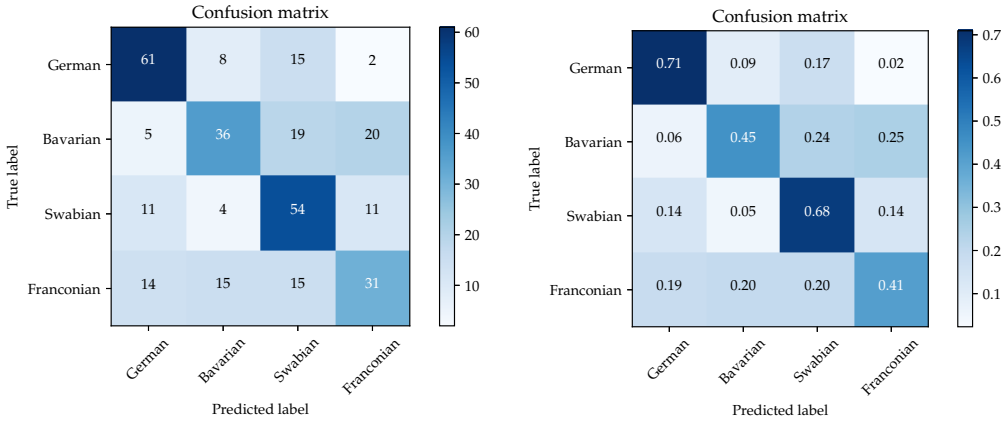
Figure 6.3: CNN architecture

6.2.3 German Broadcast Dialect Identification

In this section we use of the dialect database (Section 6.2.2) to train a dialect identification system that is able to discriminate between standard German and Franconian, Swabian and Bavarian dialect. We split the data into a training, validation and a test set. First of all, we exclude segments that are smaller than one second from the data. In a second step, we only keep a maximum of two minutes of speech data per speaker. We balance the number of speakers per dialect for each subset to have a balanced dataset. Also, speakers are disjunct across the subsets, so that a single speaker can only occur in one of the datasets. The selection which speaker goes to which dataset was performed randomly. The training set covers the segments of 35 speakers per dialect with a total of 635 segments (2.6 hours). The validation set covers 20 speakers per dialect with a total of 341 segments (1.3 hours) and the test set also covers 20 speakers per dialect with a total of 321 segments (1.3 hours).

For the task of dialect identification we train a convolutional neural network using the Keras Toolkit [125] with Tensorflow [126] backend. First, the audio signal is filtered by a first order IIR preemphasis filter ($a = 0.97$). A set of 40 mel-spaced filterbank coefficients is extracted for windows of length 25 ms with a hopsize of 10 ms. The filters cover the whole range from 0-8 kHz. The filterbank coefficients of the whole segment (zero-padded, 1 minute maximum) are then fed into a convolutional neural network (CNN), whose output is a probability function over the investigated dialects, implemented by the softmax output layer. This network has 4 outputs, one for each

dialect (including standard German). By using the softmax function the network is able to provide either class decisions or class probabilities if a soft decision is desirable. The employed CNN architecture is depicted in Figure 6.3. We use AdaDelta optimiser using the mean squared error as the loss function. For training we employ an adaptive learning rate algorithm and an early stopping mechanism based on the validation loss to avoid overfitting.



(a) Segment level (b) Normalised

Figure 6.4: Confusion matrix results of the dialect identification system on the test set

The results of this approach on the test set in terms of segment level and normalised confusion matrices are depicted in Figure 6.4. The accuracy of the system on the test set is 56.7 %. This is a promising result, considering the number of classes (4), the small amount of training data (2.6 hours) and also the fact that this data is collected from a heterogeneous set of real-world broadcast media files with speech segments of limited and highly variable length.

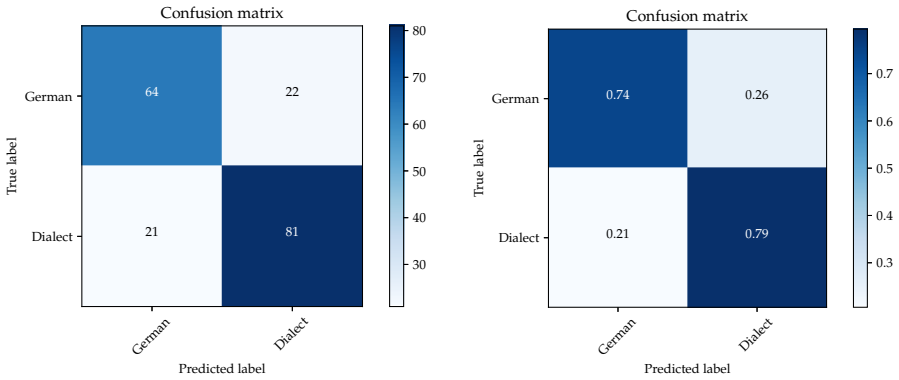
6.2.4 German Broadcast Dialect Detection

We use the same dialectal database (Section 6.2.2) to train a dialect detection system that is able to distinguish between standard German and dialectal speech. Using the soft decision classifier output the system is able to infer the "dialectness" of a speaker. Also, the algorithm can determine whether the standard German speech recognition system is likely to show degraded performance, which is the case if dialectal speakers are present.

Again, we split the data into balanced training, validation and testing datasets in a similar fashion as in Section 6.2.3. The training set covers the segments of each 35 speakers for standard German and dialectal speech with a total size of 1.2 hours. Both the validation and the testing set cover the segments of each 20 speakers for standard German and dialectal speech. The total size of the validation set is 0.8 hours and also the total size of the testing set is 0.8 hours. The dialectal speakers are also

balanced across the dialects Bavarian, Franconian and Swabian. The selection of the speakers was performed randomly. The employed CNN architecture is again depicted in Figure 6.3 and is similar to the classifier for the dialect identification task. However, in this case the neural network only has 2 outputs, one for standard German and one for dialectal speech. Also, the training procedure of the neural network is similar to the approach in Section 6.2.3.

The results of this approach on the test set are depicted in Figure 6.5. The accuracy of the system is 77.1 %, which is a promising result considering the even smaller amount of training data (1.2 hours). Note that the segment level confusion matrix is unbalanced to some extent. Even though we use an equal number of distinct speakers in the two categories, the number of annotated segments per speaker is subject to natural deviations.



(a) Segment level (b) Normalised

Figure 6.5: Confusion matrix results of the dialect detection system on the test set

6.3 Dialectal Speech Recognition

In this section we describe the development of a Swiss German dialectal speech recognition system.

6.3.1 Swiss German

Switzerland has four national languages: German/Swiss German (63%), French (22.7%), Italian (8.1%), Romansh (0.5%); the numbers in brackets are the percentages of the population speaking them[1]. As can be derived from Figure 6.6, French ("Französisch") is spoken in the west, Italian ("Italienisch") is spoken primarily in Ticino ("Tessin"), Val Bregaglia and Val Pschiavo and Romansh speakers are distributed over Graubünden.

[1] `http://www.swissinfo.ch`; Accessed on October 27th, 2017

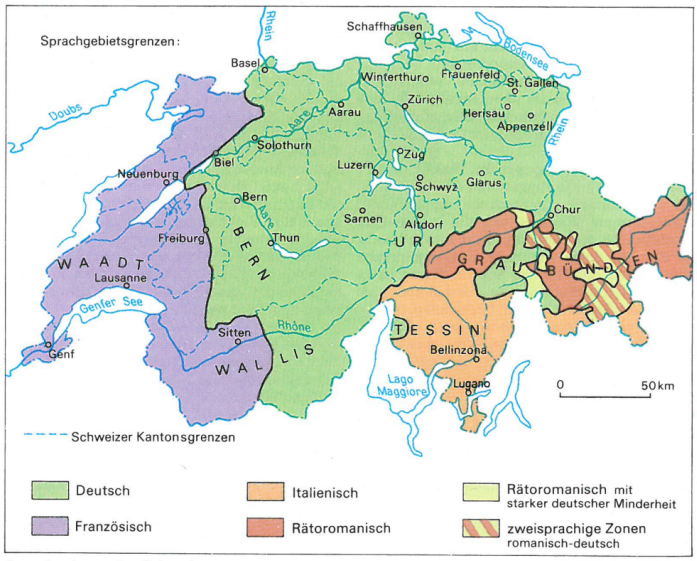


Figure 6.6: Language map of Switzerland [122]

Swiss German is not a uniform language, but is composed of many regional dialects of Alemannic origin. Although school and science maintain the standard German high-level language, after the lessons the teacher speaks in dialect to the pupils, as well as the professor to the students. On television and radio, dialect is spoken, intertwined with segments in which the reporter speaks in standard German, with the exception of a few broadcasts. Swiss German is spoken, standard German is written [122].

6.3.2 SRF Meteo Weather Report Dataset

In this section we describe the Swiss German SRF Meteo dataset, which *Schweizer Radio und Fernsehen* generously provided us for research purposes. This dataset consists of Swiss German weather reports of SRF Meteo. The speakers speak Swiss German dialect, and the textual annotation is standard German. The dataset consists of 290 Meteo weather report broadcasts with a total of 10,201 speech segments and a total of 6.5 hours of annotated speech and a total of 83,449 annotated words. The contained speech is to a large extent about weather forecasts and contains a large number of place names.

For the experiments we separated the Meteo dataset into a training, development and a test set as listed in Table 6.2.

Dataset	#Shows	#Segments	#Words	Avg. Words	#Unique	Size(h)	Avg. Length (s)
Training	260	9,181	75,215	8.2	2,981	5.9	2.3
Development	15	493	3,995	8.1	742	0.3	2.2
Test	15	527	4,242	8.1	778	0.3	2.2

Table 6.2: Statistics of the Meteo data subsets used in this work

We choose to have 260 weather reports in the training set and each 15 weather reports in the development and the testing set. The distribution of the weather reports into the datasets was performed randomly. When considering only the text of the training set for the training of a language model, the development set and the test set have an out-of-vocabulary (OOV) rate (Section 3.4.9) of $\text{OOV}_{\text{dev}} = 7.6\%$ and $\text{OOV}_{\text{test}} = 9.1\%$. This seems quite high, however the running OOV rate is acceptable considering the small amount of training data, namely $\text{OOV}_{\text{r,dev}} = 1.4\%$ and $\text{OOV}_{\text{r,test}} = 1.7\%$.

6.3.3 Swiss German Speech Recognition

In this section we aim to train a Swiss German Speech recognition system. One way to approach this is to adapt the standard German speech recognition system to the Swiss German data. For the experiments we employ the TDNN ASR model (Section 4.3.8), which was the best configuration at the time of the experiments.

The results of the standard German TDNN ASR system, which performed well for the standard German evaluation data (Section 4.3.8), are naturally worse on the Meteo data ($\text{WER}_{\text{dev}} = 81.0\%$, $\text{WER}_{\text{test}} = 79.5\%$), since there is a large mismatch in speech, phonetics and language between standard German and Swiss German. By replacing the language model trained from broadcast text by a language model trained on the text of the Meteo training dataset, we can reduce this mismatch for the Meteo evaluation data to $\text{WER}_{\text{dev}} = 64.98\%$ and $\text{WER}_{\text{test}} = 64.73\%$. In the following we try to further reduce the mismatch, especially the mismatch caused by the pronunciation, in a data-driven manner.

Standard German Speech Phoneme Decoder

We first create a phoneme decoder and then use the phoneme decodings to create a Swiss German grapheme-to-phoneme (G2P) model. For the training of the standard German Speech phoneme decoder, we use the TDNN acoustical models. For the training of the standard German phoneme language model, which is required for the phoneme decoder, we replace the words from the text of the Meteo training dataset by its likeliest pronunciations derived from our standard German G2P model which is based on Sequitur G2P [53] and the German pronunciation lexicon Phonolex [52]. Then we train a 5-gram phoneme language model and use it for decoding of the speech signals.

Data-Driven Pronunciation Modelling

By decoding the Swiss German Meteo training set using the phoneme language model, we get some suggestions of how the speech in the audio data was pronounced. However, the data is organised in utterances, rather than words. Nonetheless, we train a Swiss German G2P model by using the phrases (whitespaces are replaced by an underscore) followed by the pronunciations from the phoneme decodings. The trained Swiss German pronunciation model is able to provide some good suggestions in the n-best list for the

pronunciation of several words, as can be seen in Table 6.3. In this table we also show a non-standardised Swiss German dialectal text annotation of an online Swiss German dictionary[2] for comparison. The pronunciations from the Swiss German G2P were found in a data-driven manner, without any knowledge of the online Swiss German dictionary. As can be learned from Table 6.3 the pronunciations learned from the Swiss German G2P are often quite near to the textual correspondents from the online Swiss German dictionary.

Standard German	German G2P	Data-Driven Swiss German G2P	Swiss German Online Dictionary
Montag	m o: n t a: k	m a: n t i: k	Mäntig
Dienstag	d i: n s t a: k	ts i: S t i: k	Ziischtig
Mittwoch	m I t v O x	m I t b u: x	Mittwuch
Donnerstag	d O n 6 s t a: k	d a n S t i: k	Danschtig
Freitag	f r aI t a: k	f r i: t I k	Fritig
Samstag	z a m s t a: k	Q a m S t i: k	Samschtig
Sonntag	z O n t a: k	z o d I k	Sunntig

Table 6.3: Phoneme translations of standard German words using the standard German and the speech data-driven Swiss German G2P

We then created several lexicons, which were composed by the 1-best standard German pronunciation and a n-best list of the data-driven Swiss German G2P. The intention was to keep the 1-best standard German pronunciation as a backup, when no meaningful Swiss German pronunciation can be found by the method. We then used the enriched lexicons with the standard German TDNN models and a language model trained on the text from the Meteo training dataset. The results are depicted in Figure 6.7. We optimised the length of the n-best list on the Meteo development set. Derived from the results of the experiments, the optimal variant is to add a 2-best list of the data-driven Swiss German G2P to the 1-best standard German pronunciations. Using this adapted configuration, which includes both reasonable Swiss German and standard German pronunciations, the WER could be reduced for the Meteo development and test set to $\text{WER}_{\text{dev}} = 60.3\%$ and $\text{WER}_{\text{test}} = 56.4\%$.

Directly Trained Swiss German Speech Recognition

We also wanted to evaluate how far we can get, when we train the Swiss German models in a straight forward manner by either using grapheme pronunciations, standard German phoneme pronunciations or the combined pronunciation as described in Section 6.3.3. When using a grapheme pronunciation, each word is modelled by a sequence of its graphemes (i.e. Montag ⇒ m o n t a g). When using standard German phoneme pronunciations, we use the standard German pronunciation model, which is trained on the standard German Phonolex pronunciation lexicon [52] using Sequitur

[2] `https://www.pauker.at/pauker/DE_DE/SC/wb`

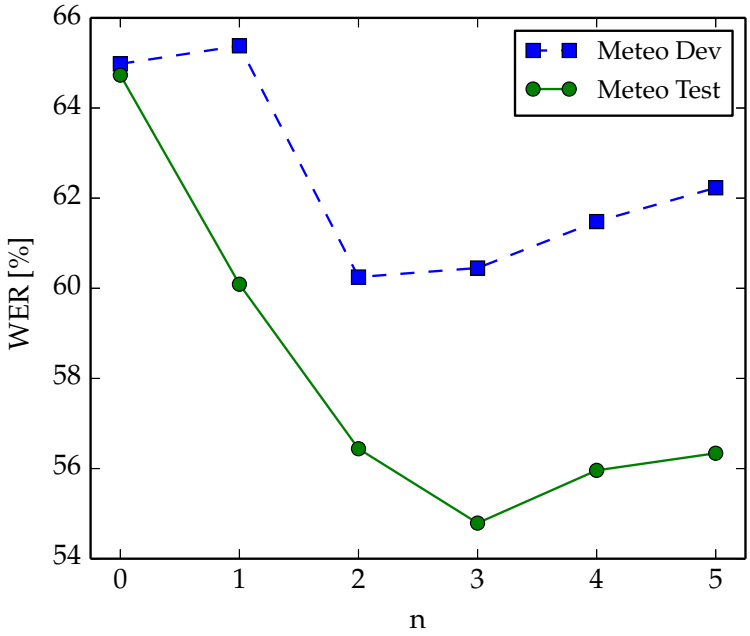


Figure 6.7: WER for different n for configurations with 1-best standard German pronunciation and n-best Swiss German pronunciations from the speech data driven G2P model

G2P [53]. For the training of the acoustical model we use the training dataset of the SRF Meteo dataset. For training the language model we use IRSTLM toolkit [127] and we use a 5-gram model with modified shift beta algorithm with back-off weights. For training of the Swiss German ASR system, we either use Eesen [48] toolkit, when using long short term memory (LSTM) recurrent neural networks (RNN) with connectionist temporal classification (CTC) training, or the Kaldi toolkit [55], when using Hidden Markov Models with Gaussian Mixture Models (HMM-GMM), or hybrid HMM with feed forward Deep Neural Networks (HMM-DNN) or the state-of-the-art time delay neural networks with projected long short-term memory (TDNN-LSTMP) layers. The results are shown in Table 6.4. The HMM-GMM, DNN and TDNN-LSTMP models from the Kaldi toolkit are trained with bootstrapping and provide more stable results in this setup (i.e. a setup with a small amount of training data) compared to the RNN models, which use Connectionist Temporal Classification (CTC) instead, and which are trained directly on the audio data. It is also remarkable that there is no big difference when comparing standard German grapheme pronunciations to standard German phoneme pronunciations. Both setups perform almost equally well. The use of the combined standard German and Swiss German pronunciation performed slightly worse compared to standard German and grapheme pronunciations for the HMM-GMM case. We believe this is the case because during training the algorithm needs a consistent single pronunciation, so the algorithm can model the pronunciation including the possible mismatches consistently. The TDNN-LSTM models trained with the standard

German G2P pronunciations performed best on the Meteo test set ($WER_{test} = 23.8\%$) given the experiments performed.

Model	Pronunciation	Development	Test
HMM-GMM	German G2P	39.7	28.9
HMM-GMM	Grapheme	40.3	29.6
HMM-GMM	Combined G2P	41.3	30.8
RNN	German G2P	44.5	32.7
RNN	Grapheme	45.0	32.3
HMM-DNN	German G2P	37.1	27.1
HMM-DNN	Grapheme	37.7	27.0
TDNN-LSTMP	German G2P	34.9	**23.8**
TDNN-LSTMP	Grapheme	**34.8**	24.3

Table 6.4: WER [%] results on the Meteo development and test set of directly trained Swiss German speech recognitions systems using different types of pronunciation lexicons; standard German G2P, combined data-driven Swiss German and standard German G2P or grapheme sequences

6.4 Summary and Contributions

In this chapter we approached the dialectal robustness of speech recognition systems in the German broadcast domain with the aim to provide optimal performance of the Fraunhofer IAIS audio mining system also for regional broadcasters. We followed the strategy to identify the underlying dialect in advance and then use dialectal speech recognition models to decode the data [121]. Therefore we first briefly discussed the manifold of German dialects in Section 6.1.

In Section 6.2 we developed and optimised a German dialect identification system. However, existing annotated dialectal resources are limited, and often of limited use as we found out in Section 6.2.1, where we trained a dialect identification system on the data with limited success. Hence, we built up a close cooperation a regional broadcaster from Bavaria, namely the *Bayerischer Rundfunk*, who provided us with 302 media files of regional programmes from Bavaria for research purposes. The dialects that are covered in the data are Bavarian, Franconian, Swabian and also standard German. 2,710 utterances from 52 media files from a total of 398 speakers (282 male, 116 female) have been annotated so far in an ongoing process. The current total size of the annotated data is 11.8 hours. This data has been used to train and evaluate both a dialectal identification and a dialect detection system based on convolutional neural networks (CNNs). The dialect identification system that has been trained on 2.6 hours of speech data taken from the proposed corpus is able to distinguish between four dialects with an accuracy of 56.7 %. The dialect detection system which has been trained on 1.2 hours of speech data taken from this corpus is able to distinguish between standard German and dialectal speech with an accuracy of 77.1 %. These are

promising results considering the low amount of training material available and the challenges that are accompanied with real-world broadcast domain data. To further increase the accuracy of the systems we will continue our efforts to extend the corpus size by annotating a larger amount of regional programmes. Is is also possible to extend the system to other dialects by exploiting regional programmes from other German speaking regions.

We intend to use specialised dialectal speech recognition models after the dialect identification step. We approached the adaptation and a training of a dialectal speech recognition system in Section 6.3. As already mentioned, annotated dialectal resources are limited, and the annotation of dialectal resources is costly. However, we established a close cooperation with the *Schweizer Radio und Fernsehen*, who generously provided us an annotated Swiss German dataset (Section 6.3.2). Since there is no standardised way to write Swiss German other than standard German, the annotations of the Swiss German audio corpus are standard German, in contrast to the audio material which is highly dialectical Swiss German. The desired output of the Swiss German speech recognition system is again standard German. Unfortunately we lacked a Swiss German pronunciation lexicon that maps standard German words into Swiss German pronunciations. We approached this problem by successfully adapting our standard German speech recognition system to the Swiss German pronunciations by the employment of a Swiss German G2P model which was learned in a data-driven manner by phoneme decodings derived from the standard German speech recognition system with the use of a phoneme language model. It turned out that by adding a 2-best list of Swiss German pronunciations derived from the data-driven Swiss German G2P model to the 1-best standard pronunciations, the adapted model provided the best results, when adapting the standard German model. However, the training of an ASR system directly on the Swiss German data by replacing the missing Swiss German pronunciation by either a standard German phoneme or grapheme sequences, provided even better results. The use of the combined lexicon did not prove to be beneficial in this case when training a system directly on the Swiss German audio data in contrast to the adaptation of the standard German model. The standard German TDNN models which perform very well on standard German data, showed degraded performance on the Swiss German data with word error rates as high as 79.5 % on the test corpus. For the Swiss German speech recognition models, the use of TDNN-LSTMP provided the best results with word error rates as low as 23.8 % on the corresponding Swiss German test set. This are encouraging results given the small amount of available training data in the Swiss German case.

The results reported in this chapter are encouraging. We intended to integrate several dialects in the dialect identification and speech recognition pipeline. However, a lot of dialectal speech resources need to be acquired, which is a time consuming and costly task, since many of the required resources simply do not exist in an annotated form at this time.

Chapter 7

Scientific Achievements and Conclusions

7.1 Scientific Achievements

In this section, we revisit the scientific goals defined at the beginning of this work in Chapter 2 and examine how far we have accomplished them.

Related to the long-term development of the German broadcast speech recognition system (Chapter 4):

- We extended the amount of training data dramatically and exploited the data for the training of the speech recognition system.
- We evaluated several state-of-the-art speech recognition algorithms that appeared during the term of this thesis from the scientific community for their employment in the German broadcast domain.
- We evaluated the speech recognition systems for the employment in a productive audio mining system. Two speech recognition models that are the outcome of this thesis have been employed in the Fraunhofer IAIS audio mining system.

Related to the automatic speech recognition decoder parameter optimisation (Chapter 5):

- We adopted a fast and efficient parameter optimisation algorithm, which has not been used in the context of speech recognition before, for the speech recognition decoder parameter optimisation.
- We employed the algorithm to optimise the accuracy of the speech recognition system.
- We extended the algorithm to jointly optimise the accuracy and the decoding speed.

- We compared the algorithm to other state-of-the-art optimisation algorithms.

Related to dialectal robustness of the speech recognition system (Chapter 6):

- We adopted a strategy how to deal with the manifold of dialects for the German speech.
- We established a close and continuing cooperation with regional broadcasters such as the *Bayerischer Rundfunk* and *Schweizer Rundfunk und Fernsehen*.
- We created a novel German dialect corpus for the task of dialect identification in the broadcast domain, where we annotated a considerable quantity of dialectal and standard German speech.
- We trained dialectal speech recognition systems with the help of data-driven pronunciation modelling and model adaptation.

7.2 Publications

The following enumeration covers the publications of the author conducted within the term of this thesis and which are related to the work in this thesis.

- M. Stadtschnitzer, D. Stein, and R. Bardeli, "Employing Stochastic Constrained LMS Algorithm for ASR Frontend Processing," in *Proc. of The 2nd CHiME Speech Separation and Recognition Challenge*, Vancouver, Canada, 2013
- J. Schwenninger, D. Stein, and M. Stadtschnitzer, "Automatic Parameter Tuning and Extended Training Material: Recent Advances in the Fraunhofer Speech Recognition System," *Proc. Workshop Audiosignal- und Sprachverarbeitung*, 2013
- D. Stein, J. Schwenninger, and M. Stadtschnitzer, "Simultaneous Perturbation Stochastic Approximation for Automatic Speech Recognition," *Proc. Interspeech*, pp. 622–626, 2013
- D. Stein, B. Krausz, J. Löffler, R. Marterer, R. Bardeli, M. Stadtschnitzer, and J. Schwenninger, "Automatic Audio and Video Event Recognition in an Intelligent Resource Management System," *IJISCRAM*, vol. 5, no. 4, pp. 1–12, 2013
- M. Stadtschnitzer, J. Schwenninger, D. Stein, and J. Koehler, "Exploiting the Large-Scale German Broadcast Corpus to Boost the Fraunhofer IAIS Speech Recognition System," in *Proceedings of the 9th Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland, May 2014
- M. Stadtschnitzer, J. Koehler, and D. Stein, "Improving Automatic Speech Recognition for Effective Topic Segmentation," in *Proc. DAGA - 40. Jahrestagung für Akustik*, Oldenburg, Germany, 2014

- M. Stadtschnitzer, C. Schmidt, and D. Stein, "Towards a Localised German Automatic Speech Recognition," in *Proc. 11. ITG Fachtagung Sprachkommunication*, Erlangen, Germany, 2014
- T. L. Nguyen, D. Stein, and M. Stadtschnitzer, "Gradient-Free Decoding Parameter Optimization on Automatic Speech Recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 3261 – 3265
- H. Le, Q. Bui, B. Huet, B. Cervenková, J. Bouchner, E. Apostolidis, F. Markatopoulou, A. Pournaras, V. Mezaris, D. Stein, S. Eickeler, and M. Stadtschnitzer, "LinkedTV at MediaEval 2014 Search and Hyperlinking Task," in *Proceedings of the MediaEval 2014 Workshop*, Catalunya, Spain, October 2014
- M. Stadtschnitzer and C. Schmidt, "Implementation of a Live Dialectal Media Subtitling System," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Dresden, Germany, Sep. 2015, pp. 728–729
- C. Schmidt, M. Stadtschnitzer, and J. Köhler, "The Fraunhofer IAIS Audio Mining System: Current State and Future Directions," in *Proceedings of the 12th ITG Conference on Speech Communication*, Paderborn, Germany, 2016
- M. Stadtschnitzer and C. Schmidt, "Joint Standard German and Bavarian Subdialect Identification of Broadcast Speech," in *Proceedings of DAGA - 44. Jahrestagung für Akustik*, München, Germany, Mar. 2018
- M. Stadtschnitzer and C. Schmidt, "Adaptation and Training of a Swiss German Speech Recognition System using Data-driven Pronunciation Modelling," in *Proceedings of DAGA - 44. Jahrestagung für Akustik*, München, Germany, Mar. 2018
- M. Stadtschnitzer and C. Schmidt, "Data-Driven Pronunciation Modelling of Swiss German Dialectal Speech for Automatic Speech Recognition," in *Proc. of 11th Edition of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan, May 2018

7.3 Conclusions

In this thesis, we have addressed three issues with regard to robust speech recognition in the German broadcast domain.

First, we developed and optimised the speech recognition system, which is part of the Fraunhofer IAIS audio mining system, over a long period of time. Therefore, we continuously evaluated a large number of speech recognition algorithms that became available in the scientific community in the course of this thesis for the employment in the German broadcast domain and found the optimal configuration of the systems for

our use case and extended the training data size by annotating and exploiting a large quantity of speech data.

Second, we introduced a fast and efficient parameter optimisation algorithm, which has not been employed in the context of speech recognition before, to speech recognition decoder parameter optimisation and compared the algorithm to state-of-the-art decoder parameter optimisation algorithms. We employed the algorithm to optimise the accuracy of the speech recognition system and extended the algorithm to jointly optimise the accuracy and the decoding speed.

Third, to handle the manifold of German dialects for speech recognition in the broadcast domain, we adopted an approach to determine the dialect of a speaker in advance to choose from dialectal speech recognition models. Therefore we acquired, employed and created German dialectal resources to train dialect identification and dialectal speech recognition models.

By approaching the three issues we arrived at a German broadcast speech recognition system with high performance and optimal decoder parameters that fulfils the requirements of a productive audio mining system and that is able to cope with the manifold of German dialects which occur in the programmes of the public broadcaster ARD and its regional broadcast stations.

Appendix A

Toolkits

In this section the software and toolkits that were used throughout this thesis, are briefly described.

A.1 HTK Toolkit

The Hidden Markov Model Toolkit (HTK) [61] is a toolkit for the training and manipulation of Hidden Markov Models (HMM). HTK is primarily built for speech recognition research, however it has also been used for other topics including speech synthesis and character recognition. HTK is developed and maintained by the University of Cambridge Department of Engineering (CUED). The current stable version is version 3.4.1.

`http://htk.eng.cam.ac.uk/`

A.2 Kaldi

The Kaldi Toolkit [55] is a open-source toolkit for the automatic speech recognition research. The speech recognition systems trained with Kaldi are based on finite-state transducers (implemented by the open-source framework OpenFst). Kaldi includes example scripts for the training of speech recognition systems for the most prominent speech recognition research databases (e.g. TIMIT, Switchboard, etc.). Kaldi is written in C++ and is written in a very modular fashion (i.e., it consists of a large set of scripts and executables).

`http://kaldi-asr.org/`

A.3 Eesen

The Eesen Toolkit [48] is open-source toolkit for the creation of automatic speech recognition systems based on bi-directional Recurrent Neural Networks (RNN) with Long Short-time Memory (LSTM) units and on Connectionist temporal classification (CTC) alignment. The toolkit was created by Yajie Miao based on the Kaldi Toolkit. The toolkit discards the exhaustive bootstrapping process by replacing it with a straightforward learning problem.

`https://github.com/yajiemiao/eesen`

A.4 RNNLM

The RNNLM Toolkit [98] is a open-source toolkit for the training and evaluation of statistical language models (LM) based on recurrent neural networks (RNN) for automatic speech recognition and machine translation. The toolkit is written by Tomas Mikolov. The author reports significant improvements over classic backoff m-gram models.

`http://www.fit.vutbr.cz/~imikolov/rnnlm/`

A.5 IRSTLM

The IRSTLM toolkit [127] is a open-source toolkit that allows the efficient training of large scale n-gram language models. It allows to train language models with billions of m-grams on conventional computers. The software has been integrated in the open-source statistical machine translation decoder Moses and is also part of the Kaldi toolkit. The software is developed by FBK-HLT in Trento.

`http://hlt-mt.fbk.eu/technologies/irstlm`

A.6 Sequitur-G2P

Sequitur-G2P [53] is a statistical grapheme-to-phoneme converter developed at RWTH Aachen University - Department of Computer Science. The software was developed by Maximilian Bisani. The software is released under the GNU Public License.

`https://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html`

A.7 TheanoLM

TheanoLM [140] is a toolkit for the training of neural network language models and which uses the Python library Theano. The authors report significant improvement over back-off n-gram models.

`https://github.com/senarvi/theanolm`

A.8 Keras

Keras [125] is a high-level neural network library written in Python. It employs either TensorFlow or Theano as backend. Keras allows for rapid prototyping and supports both recurrent neural networks, convolutional neural networks, dense networks and combinations of architectures.

`https://keras.io/`

List of Figures

List of Tables

Bibliography

[1] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Trans. Information Theory*, vol. IT-13, pp. 260–269, 1967.

[2] L. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 4–16, 1986.

[3] J. Godfrey and E. Holliman, "Switchboard-1 release 2 ldc97s62," Web Download, 1993, linguistic Data Consortium.

[4] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comput.*, vol. 15, pp. 297–301, 1965.

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

[6] D. O'Shaughnessy, *Speech communication: human and machine*. Addison-Wesley, 1987.

[7] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 1, pp. 52–59, 1986.

[8] L. E. Baum and T. Petire, "Statistical inference for probabilistic function of finite state markov chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

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

[10] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. of INTERSPEECH*, Signapore, 2014, pp. 338–342.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 1–46.

[12] N. Chen, S. Urban, C. Osendorfer, J. Bayer, and P. van der Smagt, "Estimating finger grip force from an image of the hand using Convolutional Neural Networks and Gaussian processes," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3137–3142.

[13] G. Saon and M. Picheny, "Recent advances in conversational speech recognition using convolutional and recurrent neural networks," *IBM Journal of Research and Development*, vol. 61, no. 4, pp. 1:1–1:10, July 2017.

[14] S. Ganapathy, K. Han, S. Thomas, M. Omar, M. V. Segbroeck, and S. S. Narayanan, "Robust Language Identification Using Convolutional Neural Network Features," in *Proc. of INTERSPEECH*, Signapore, China, 2014.

[15] C. Bartz, T. Herold, H. Yang, and C. Meinel, "Language Identification Using Deep Convolutional Recurrent Neural Networks," in *Proc. of International Conference on Neural Information Processing (ICONIP)*, Guangzhou, China, 2017.

[16] W. Commons, "Typical CNN architecture," Available at https://commons.wikimedia.org/wiki/File:Typical_cnn.png; Accessed on 2018-02-22., 2015.

[17] B. H. Juang and L. R. Rabiner, "Automatic speech recognition – a brief history of the technology development," 2005.

[18] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic recognition of spoken digits," *J. Acoust. Soc. Am.*, vol. 24, pp. 627–642, 1952.

[19] H. F. Olson and H. Belar, "Phonetic typewriter," *J. Acoust. Soc. Am.*, vol. 28, no. 6, pp. 1072–1081, 1956.

[20] D. B. Fry and P. Denes, "The design and operation of the mechanical speech recognizer at university college london," *J. British Inst. Radio Engr.*, vol. 19, pp. 211–229, 1959.

[21] T. B. Martin, A. L. Nelson, and H. J. Zadell, "Speech recognition by feature abstraction techniques," Air Force Avionics Lab, Tech. Rep., 1764, tech. Report AL-TDR-64-176.

[22] T. K. Vintsyuk, "Speech discrimination by dynamic programming," *Kibernetika*, vol. 4, no. 2, pp. 81–88, 1968.

[23] A. Newell, "Harpy, production systems and human cognition," Computer Science Department, Carnegie Mellon University, Tech. Rep., 1978, paper 2319.

[24] J. G. David Forney, "The viterbi algorithm," *Proc. of the IEEE*, vol. 61, pp. 268–278, 1973.

[25] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electronics and Communications in Japan*, vol. 53A, pp. 36–43, 1970.

[26] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *Journal of the Acoustical Society of America*, vol. 50, no. 2, pp. 637–655, 1971.

[27] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1035–1074, 1983.

[28] J. D. Ferguson, "Hidden markov analysis: An introduction in hidden markov models for speech," Institue for Defense Analyses, Princeton, NJ, Tech. Rep., 1980.

[29] L. R. B. F. Jelinek and R. L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE. Trans. on Information Theory*, vol. IT-21, pp. 250–256, 1975.

[30] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 179–190, 1983.

[31] P. Mermelstein, "Distance measure for speech recognition, psychological and instrumental," *Joint Workshop on Pattern Recognition and Artificial Intelligence*, 1976.

[32] J. S. Bridle and M. D. Brown, "An experimental automatic word-recognition system," Joint Speech Research Unit, Tech. Rep., 1974, technical Report 1003.

[33] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer Speech and Language*, vol. 9, no. 1, pp. 171–185, 1995.

[34] Q. Huo and D.-H. Lee, "On-line adaptive learning of the cdhmm based on the approximate recursive bayes estimate," *IEEE Trans. on Speech and Audio Proc.*, vol. 5, no. 2, 1997.

[35] H. Hermansky, "Perceptual linear pedictive (plp) analysis of speech," *J. Acoust. Soc. Am.*, vol. 87, no. 4, pp. 1738–1752, 1990.

[36] L. Den, A. Acero, M. Plumpe, and X. Huang, "Large-vocabulary speech recognition under adverse acoustic environments," *Proc. Int. Conf. Spoken Language Processing (ICSLP)*, pp. 806–809, 2000.

[37] D. Povey, B. Kingsbury, L.Mangu, G. Saon, H. Soltau, and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," *Proc. Int. Conf. Acoustic, Speech, and Signal Processing*, pp. 961–964, 2000.

[38] M. P. Cooke, P. G. Green, and M. D. Crawford, "Handling missing data in speech recognition," *Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP)*, pp. 1555–1558, 1994.

[39] M. P. Cooke, A. Morris, and P. D. Green, "Missing data techniques for robust speech recognition," *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 863–866, 1997.

[40] R. P. Lippmann and B. A. Carlson, "Using missing feature theory to actively select features for robust speech recogntion with interruptions, filtering and noise," *Proc. Eurospeech*, pp. 37–40, 1997.

[41] S. Kapadia, V. Valtchev, and S. Young, "MMI Training for continuous phoneme recognition on the TIMIT database," in *Proceedings of ICASSP*, vol. 2, 1993, pp. 491–494.

[42] B. H. Juang, W. Chou, and C. H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Transactions on Speech Audio Processing*, vol. 5, no. 3, pp. 257–265, May 1997.

[43] E. McDermott, T. Hazen, J. L. Roux, A. Nakamura, and S. Katagiri, "Discriminative training for large vocabulary speech recognition using minimum classification error," *IEEE Transactions of Speech Audio Processing*, vol. 5, no. 3, pp. 203–223, 2007.

[44] D. Povey and P. Woodland, "Minimum phone error and i-smoothing for improved discriminative training," in *Proceedings of ICASSP*, 2002, pp. 105–108.

[45] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio Speech Lang. Processing*, vol. 20, no. 1, pp. 30–42, 2012.

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

[47] F. Seide, G. Li, X. Chen, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," *Proc. Interspeech*, pp. 437–440, 2011.

[48] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, Scottsdale, Arizona, USA, Dec. 2015, pp. 167–174.

[49] T. Bayes, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions*, vol. 53, pp. 370–418, 1763.

[50] H. Ney, "Introduction to automatic speech recognition. technical report," RWTH Aachen University, Aachen, Germany, Lecture script, 2007.

[51] C. M. University, "The CMU Pronouncing Dictionary," http://www.speech.cs.cmu.edu/cgi-bin/cmudict, version 0.7b, accessed October 19th, 2017.

[52] BAS - Bavarian Archive for Speech Signals, "Pronunciation lexicon PHONOLEX," https://www.phonetik.uni-muenchen.de/forschung/Bas/BasPHONOLEXeng.html, 2013.

[53] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, pp. 434–451, Jul. 2008.

[54] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[55] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011, iEEE Catalog No.: CFP11SRW-USB.

[56] R. Bakis, "Continuous speech word recognition via centisecond acoustic states," *Journal of the Acoustical Society of America*, vol. 59, no. 1, Apr. 1976.

[57] H. Ney, "Acoustic modeling of phoneme units for continuous speech recognition," in *Proc. European Signal Processing Conference*, Barcelona, Spain, Sep. 1990, pp. 66–72.

[58] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[59] R. Kneser and H. Ney, "mproved backing-off for m-gram language modeling," in *Proc. of ICASSP*, vol. 1, 1995, pp. 181–184.

[60] L. E. Baum, "An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[61] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[62] A. Lee, T. Kawahara, and K. Shikano, "Julius – an Open Source Real-Time Large Vocabulary Recognition Engine," in *Proceedings of Eurospeech*, Aalborg, Denmark, 2001, pp. 1691–1694.

[63] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel, "Sphinx-4: A Flexible Open Source Framework for Speech Recognition," Sun Microsystems Inc., Tech. Rep., 2004.

[64] J. Kacur and J. Korosi, "An accuracy optimization of a dialog asr system utilizing evolutional strategies," in *5th International Symposium on Image and Signal Processing and Analysis*, Istanbul, 2007, pp. 180–184.

[65] B. Mak and T. Ko, "Automatic estimation of decoding parameters using large-margin iterative linear programming," in *Proc. of Interspeech*, 2009, pp. 1219–1222.

[66] A. El Hannani and T. Hain, "Automatic optimization of speech decoder parameters," *Signal Processing Letters, IEEE*, vol. 17, no. 1, pp. 95–98, 2010.

[67] I. Bulyko, "Speech recognizer optimization under speed constraints," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[68] J. Kiefer and J. Wolfowitz, "Stochastic estimation of a regesssion function," *Ann. Math. Stat.*, vol. 23, pp. 462–466, 1952.

[69] F. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 160–167.

[70] J. Nelder and R. Mead, "The Downhill Simplex Method," *Computer Journal*, vol. 7, p. 308, 1965.

[71] P. Lambert and R. Banchs, "Tuning machine translation parameters with SPSA," in *Proc. IWSLT*, 2006, pp. 190–196.

[72] M. Mohri, "Finite-State Transducers in Language and Speech Processing," *Computational Linguistics*, vol. 23, no. 2, 1997.

[73] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.

[74] D. Klakow and J. Peters, "Testing the correlation of word error rate and perplexity," *Speech Communication*, vol. 38, no. 1–2, pp. 19–28, 2002.

[75] A. Lazaridis, E. Khoury, J.-P. Goldman, M. Avanzi, S. Marcel, and P. N. Garner, "Swiss french regional accent identification," in *Proc. Odessey: The Speaker and Language Recognition Workshop*, 2014.

[76] F. Biadsy, J. Hirschberg, and N. Habash, "Spoken arabic dialect identification using phonotactic modeling," in *Proc. EACL 2009 Workshop on Computational Approaches to Semantic Languages*, Athens, Greece, Mar. 2009, pp. 53–61.

[77] M. H. Bahari, R. Saeidi, H. V. hamme, and D. V. Leeuwen, "Accent recognition using i-vector, gaussian mean supervector and gaussian posterior probability supervector for spontaneous telephone speech," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver B.C., 2013, pp. 7344–7348.

[78] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, 1996.

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

[80] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2003.

[81] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[82] W. Campbell, D. Sturim, and D. Reynolds, "Support vector machines using gmm supervectors for speaker verification," *Signal Processing Letters*, vol. 13, no. 5, pp. 308–311, 2006.

[83] P. Kenny, P. Ouellet, N. Dehal, V. Gupta, and P. Dumouchel, "A study of inter-speaker variability in speaker verification," *IEEE Transactions of Audio, Speech and Language Processing*, vol. 16, no. 5, pp. 980–988, 2008.

[84] C. Schmidt, M. Stadtschnitzer, and J. Köhler, "The Fraunhofer IAIS Audio Mining System: Current State and Future Directions," in *Proceedings of the 12th ITG Conference on Speech Communication*, Paderborn, Germany, 2016.

[85] D. Baum, D. Schneider, R. Bardeli, J. Schwenninger, B. Samlowski, T. Winkler, and J. Köhler, "DiSCo - a German evaluation corpus for challenging problems in the broadcast domain," in *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC)*, Valletta, Malta, May 2010.

[86] D. Schneider, J. Schon, and S. Eickeler, "Towards Large Scale Vocabulary Independent Spoken Term Detection: Advances in the Fraunhofer IAIS Audiomining System," in *Proc. Association for Computing Machinery's Special Interest Group Information Retrieval (ACM SIGIR)*, Singapore, 2008.

[87] M. Stadtschnitzer, J. Schwenninger, D. Stein, and J. Koehler, "Exploiting the Large-Scale German Broadcast Corpus to Boost the Fraunhofer IAIS Speech Recognition System," in *Proceedings of the 9th Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland, May 2014.

[88] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37:3, Mar. 1992.

[89] D. Povey, L. Burget, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. K. Goel, M. Karafiát, A. Rastrow, R. C. Rose, P. Schwarz, and S. Thomas,

"Subspace gaussian mixture models for speech recognition," in *Proc. ICASSP*, 2010, pp. 4330–4333.

[90] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *Fourth International Conference on Spoken Language (ICSLP)*, Philadelphia, USA, Oct. 1996.

[91] V. Nair and G. Hinton, "3-d object recognition with deep belief nets," *Adv. Neural Inf. Process. Syst.*, vol. 22, pp. 1339–1347, 2007.

[92] V. Mnih and G. Hinton, "Learning to detect roads in high-resolution aerial images," in *Proc. 11th Eur. Conf. Comput. Vision (ECCV)*, Sep. 2010.

[93] R. Salakhutdinov and G. Hinton, "Semantic hashing," in *Proc. SIGIR Workshop Inf. Retrieval Applicat. Graph. Models*, 2007.

[94] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn. ser. ICML'08*, New York, 2008.

[95] M. Hwang and X. Huang, "Shared-distribution hidden markov models for speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 1, no. 4, pp. 414–420, Jan. 1993.

[96] T. Mikolov, M. Karafiat, L. Burget, J. H. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of Interspeech*, Makuhari, Chiba, Japan, 2010, pp. 1045–1048.

[97] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[98] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. H. Cernocky, "Rnnlm - recurrent neural network language modeling toolkit," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Dec. 2011.

[99] M. Zeiler, M. Ranzato, R. Monga, and e. a. M. Mao, "On rectified linear units for speech processing," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[100] J. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[101] Y. Miao, S. Rawat, and F. Metze, "Deep maxout neural networks for speech recognition," in *Proc. ASRU*, 2013.

[102] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Proceedings of 2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 215–219.

[103] A. Graves, N. Jaitly, and A. rahman Mohamed, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

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

[105] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning (ACM)*, 2006, pp. 369–376.

[106] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct 1990.

[107] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of INTERSPEECH*, Dresden, Germany, Sep. 2015.

[108] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, Mar. 1989.

[109] G. Cheng, V. Peddinti, D. Povey, V. Manohar, S. Khudanpur, and Y. Yan1, "An exploration of dropout with LSTMs," in *Proceedings of INTERSPEECH*, Stockholm, Sweden, Aug 2017.

[110] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[111] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, May 2014.

[112] J. Carletta and S. A. et. al., "The ami meeting corpus: a pre-announcement," in *Proceedings of the Second international conference on Machine Learning for Multimodal Interaction (MLMI)*, Edinburgh, UK, 2005, pp. 28–39.

[113] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," in *ArXiv e-prints*, Feb. 2014.

[114] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proceedings of INTERSPEECH*, Dresden, Germany, Sep. 2015.

[115] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017.

[116] J. C. Spall, "A stochastic approximation technique for generating maximum likelihood parameter estimates," 1987, pp. 1161–1167.

[117] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34:3, Jul. 1998.

[118] I. Rechenberg, "Evolutionsstrategie – optimierung technischer systeme nach prinzipien der biologischen evolution," Ph.D. dissertation, Technical University of Berlin, 1971.

[119] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[120] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, Eds. Springer, 2006, pp. 75–102.

[121] F. Biadsy, "Automatic dialect and accent recognition and its application to speech recognition," Ph.D. dissertation, Columbia University, 2011.

[122] W. König, *dtv-Atlas - Deutsche Sprache*. Deutscher Taschenbuch Verlag, 2011, iSBN: 978-3-423-03025-0.

[123] S. Burger and F. Schiel, "RVG 1-A Database for Regional Variants of Contemporary German," in *Proceedings of the First International Conference on Language Resources and Evaluation*, 1998, pp. 1083–1087.

[124] H. Brugman and A. Russel, "Annotating Multimedia/Multi-modal resources with ELAN," in *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, 2004.

[125] F. Chollet *et al.*, "Keras: The Python Deep Learning library," https://github.com/fchollet/keras, 2015.

[126] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," https://www.tensorflow.org, 2015.

[127] M. Federico, N. Bertoldi, and M. Cettolo, "Irstlm: an open source toolkit for handling large scale language models," in *Proceedings of Interspeech*, Brisbane,Australia, 2008.

[128] M. Stadtschnitzer, D. Stein, and R. Bardeli, "Employing Stochastic Constrained LMS Algorithm for ASR Frontend Processing," in *Proc. of The 2nd CHiME Speech Separation and Recognition Challenge*, Vancouver, Canada, 2013.

[129] J. Schwenninger, D. Stein, and M. Stadtschnitzer, "Automatic Parameter Tuning and Extended Training Material: Recent Advances in the Fraunhofer Speech Recognition System," *Proc. Workshop Audiosignal- und Sprachverarbeitung*, 2013.

[130] D. Stein, J. Schwenninger, and M. Stadtschnitzer, "Simultaneous Perturbation Stochastic Approximation for Automatic Speech Recognition," *Proc. Interspeech*, pp. 622–626, 2013.

[131] D. Stein, B. Krausz, J. Löffler, R. Marterer, R. Bardeli, M. Stadtschnitzer, and J. Schwenninger, "Automatic Audio and Video Event Recognition in an Intelligent Resource Management System," *IJISCRAM*, vol. 5, no. 4, pp. 1–12, 2013.

[132] M. Stadtschnitzer, J. Koehler, and D. Stein, "Improving Automatic Speech Recognition for Effective Topic Segmentation," in *Proc. DAGA - 40. Jahrestagung für Akustik*, Oldenburg, Germany, 2014.

[133] M. Stadtschnitzer, C. Schmidt, and D. Stein, "Towards a Localised German Automatic Speech Recognition," in *Proc. 11. ITG Fachtagung Sprachkommunication*, Erlangen, Germany, 2014.

[134] T. L. Nguyen, D. Stein, and M. Stadtschnitzer, "Gradient-Free Decoding Parameter Optimization on Automatic Speech Recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 3261 – 3265.

[135] H. Le, Q. Bui, B. Huet, B. Cervenková, J. Bouchner, E. Apostolidis, F. Markatopoulou, A. Pournaras, V. Mezaris, D. Stein, S. Eickeler, and M. Stadtschnitzer, "LinkedTV at MediaEval 2014 Search and Hyperlinking Task," in *Proceedings of the MediaEval 2014 Workshop*, Catalunya, Spain, October 2014.

[136] M. Stadtschnitzer and C. Schmidt, "Implementation of a Live Dialectal Media Subtitling System," in *Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Dresden, Germany, Sep. 2015, pp. 728–729.

[137] M. Stadtschnitzer and C. Schmidt, "Joint Standard German and Bavarian Subdialect Identification of Broadcast Speech," in *Proceedings of DAGA - 44. Jahrestagung für Akustik*, München, Germany, Mar. 2018.

[138] M. Stadtschnitzer and C. Schmidt, "Adaptation and Training of a Swiss German Speech Recognition System using Data-driven Pronunciation Modelling," in

Proceedings of DAGA - 44. Jahrestagung für Akustik, München, Germany, Mar. 2018.

[139] M. Stadtschnitzer and C. Schmidt, "Data-Driven Pronunciation Modelling of Swiss German Dialectal Speech for Automatic Speech Recognition," in *Proc. of 11th Edition of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan, May 2018.

[140] S. Enarvi and M. Kurimo, "TheanoLM - An Extensible Toolkit for Neural Network Language Modeling," in *Proc. of INTERSPEECH*, San Francisco, USA, 2016.