BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

IMPROVING ROBUSTNESS OF SPEAKER RECOGNITION USING DISCRIMINATIVE TECHNIQUES

ZVYŠOVÁNÍ ROBUSTNOSTI SYSTÉMŮ PRO ROZPOZNÁVÁNÍ MLUVČÍCH POMOCÍ DISKRIMINATIVNÍCH TECHNIK

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — **Ing. ONDŘEJ NOVOTNÝ**
AUTOR PRÁCE

SUPERVISOR — **doc. Dr. Ing. JAN ČERNOCKÝ**
ŠKOLITEL

BRNO 2021

Abstract

This work deals with discriminative techniques in speaker verification systems to improve robustness of the systems against factors that negatively affect their performance. These factors include noise, reverberation, or the transmission channel.

The thesis consists of two main parts. In the first part, it deals with a theoretical introduction to current state-of-the-art speaker verification systems. The recognition system's steps are described, starting from the extraction of acoustic features, the extraction of vector representations of recordings, and the final recognition score computation. Particular emphasis is paid to the techniques of extraction of a vector representation of a recording, where we describe two different paradigms: the i-vectors and the x-vectors. The second part of the work focuses more on discriminative techniques to increase robustness. Their description is organized to match the gradual passage of the recording through the verification system. First, attention is paid to signal pre-processing using a neural network for noise reduction and speech enhancement. This pre-processing is a universal technique independent of the verification system. The work follows by focusing on the use of a discriminative approach in the extraction of features and the extraction of vector representations of recordings.

Furthermore, this work sheds light on the transition from generative systems to discriminative systems. In order to give a fuller context, the work also describes techniques that had historically preceded this transition. All presented techniques are always experimentally verified and their advantages evaluated. We are proposing several techniques that have proved successful in both the generative approach in the form of i-vectors and discriminative x-vectors, and thanks to them, considerable improvement has been achieved. For completeness, in the field of robustness, other techniques are included in the work, such as normalization of scores or multi-condition training. Finally, the work deals with the robustness of discriminative systems in terms of data used in their training.

Keywords

Speaker verification, generative training, discriminative training, speech enhancement, i-vector, x-vector, robustness, noise, reverberation, neural networks.

Bibliographic citation

NOVOTNÝ, Ondřej. *Improving Robustness of Speaker Recognition using Discriminative Techniques*. Brno, 2021. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Dr. Ing. Jan Černocký

Abstrakt

Tato práce pojednává o využití diskriminativních technik v oblasti rozpoznávání mluvčích za účelem získání větší robustnosti těchto systémů vůči vlivům negativně ovlivňující jejich výkonnost. Mezi tyto vlivy řadíme šum, reverberaci nebo přenosový kanál.

Práce je rozdělena do dvou hlavních částí. V první části se věnujeme teoretickému úvodu do problematiky rozpoznávání mluvčích. Popsány jsou jednotlivé kroky rozpoznávacího systému od extrakce akustických příznaků, extrakce vektorových reprezentací nahrávek, až po tvorbu finálního rozpoznávacího skóre. Zvláštní důraz je věnován technikám extrakce vektorové reprezentace nahrávky, kdy popisujeme dvě rozdílná paradigmata možného přístupu, i-vektory a x-vektory.

Druhá část práce se již více věnuje diskriminativním technikám pro zvýšení robustnosti. Techniky jsou organizovány tak, aby odpovídaly postupnému průchodu nahrávky rozpoznávacím systémem. Nejdříve je věnována pozornost předzpracování signálu pomocí neuronové sítě pro odšumění a obohacení signálu řeči jako univerzální technice, která je nezávislá na následně použitém rozpoznávacím systému. Dále se zaměřujeme na využití diskriminativního přístupu při extrakci příznaků a extrakci vektorových reprezentací nahrávek.

Práce rovněž pokrývá přechod od generativního paradigmatu k plně diskriminativnímu přístupu v systémech pro rozpoznávání mluvčích. Veškeré techniky jsou následně vždy experimentálně ověřeny a zhodnocen jejich přínos. V práci je navrženo několik přístupů, které se osvědčily jak u generativního přístupu v podobě i-vektorů, tak i u diskriminativních x-vektorů, a díky nim bylo dosaženo významného zlepšení.

Pro úplnost jsou, v oblasti problematiky robustnosti, do práce zařazeny i další techniky, jako je normalizace skóre, či více-scénářové trénování systémů. Závěrem se práce zabývá problematikou robustnosti diskriminativních systému z pohledu dat využitých při jejich trénování.

Klíčová slova

Rozpoznávání mluvčího, generativní trénování, diskriminativní trénování, obohacování řečového signálu, i-vektor, x-vektor, robustnost, šum, reverberace, neuronové sítě.

Bibliografická citace

NOVOTNÝ, Ondřej. *Zvyšování robustnosti systémů pro rozpoznávání mluvčích pomocí diskriminativních technik*. Brno, 2021. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Školitel doc. Dr. Ing. Jan Černocký

Improving Robustness of Speaker Recognition using Discriminative Techniques

Declaration

I hereby declare that this thesis and the work reported herein was composed by and originated entirely from me. The work has been supervised by Jan Černocký, Lukáš Burget, and Ondřich Plchot. Information derived from the published and unpublished work of others has been acknowledged in the text, and references are given in the list of sources. Some reported systems were created by the members of the BUT Speech@FIT research group.

.....
Ondřej Novotný
May 2, 2021

Acknowledgements

I would like to thank Honza Černocký and Lukáš Burget for their excellent leadership during my studies. I was honored to study and work under the supervision of such great personalities. Honza's leadership allowed me to make significant progress in education and participate in prestigious conferences and projects.

I would also like to thank all the BUT Speech@FIT research group members for their help and support. Especially the team dealing with speaker recognition: Pavel Matějka, Anna Silnova, Johan Rohdin, Ladislav Mošner, Alicia Lozano Díez, and Hosein Zeinali. Special thanks belong to Oldřich Plchot and Onřej Glembek, who were both very willing to help with any problem, always taking the time to consult on any topic or article. I wish to thank all co-authors of underlying publications for their support and consent to the use some texts.

Very special thanks go to my parents for their support and guidance that brought me and allowed me to study at such a prestigious school. To conclude, one special thank goes to my wife, Anežka. She supported me and always patiently helped me in life during all the difficulties of this work.

Contents

1 Introduction **1**
- 1.1 Speaker Verification System . . . 2
- 1.2 Features . . . 3
  - 1.2.1 Mel-Frequency Cepstral Coefficients . . . 3
  - 1.2.2 Feature Derivatives . . . 4
  - 1.2.3 Feature Normalization . . . 4
- 1.3 Voice Activity Detection . . . 5
- 1.4 Score Normalization . . . 6
  - 1.4.1 Z-norm . . . 7
  - 1.4.2 T-norm . . . 7
  - 1.4.3 ZT-norm . . . 7
  - 1.4.4 S-norm . . . 7
  - 1.4.5 Adaptive Normalization . . . 8
- 1.5 Calibration . . . 9
- 1.6 Motivation and Contribution . . . 9
  - 1.6.1 Claims . . . 10
  - 1.6.2 Structure of the Thesis . . . 10

2 Factors Influencing SV Performance **12**
- 2.1 Transmission Channel . . . 13
- 2.2 Language . . . 13
- 2.3 Acoustic Environment . . . 13
  - 2.3.1 Additive Noise . . . 14
  - 2.3.2 Reverberation . . . 14
- 2.4 Recording System . . . 15

3 SRE Datasets and Evaluation Metrics **17**
- 3.1 Speaker Verification Evaluation . . . 17
  - 3.1.1 Detection Error Tradeoff Plot . . . 18
  - 3.1.2 Equal Error Rate . . . 19
  - 3.1.3 Detection Cost Function . . . 20
- 3.2 Datasets . . . 21
  - 3.2.1 NIST . . . 21
  - 3.2.2 Fisher English . . . 23
  - 3.2.3 Switchboard . . . 23
  - 3.2.4 PRISM . . . 24
  - 3.2.5 SITW . . . 24
  - 3.2.6 BUT Retransmitted Data . . . 25

3.3 Data Augmentation Design . . . 25
3.3.1 Noise . . . 25
3.3.2 Reverberation . . . 26
3.3.3 Composition of the Training Set . . . 27
3.3.4 Kaldi Data Augmentation Recipe . . . 27
3.3.5 Selected Benchmark Scenarios . . . 28

4 Embedding-Based Speaker Verification 31
4.1 Generatively Trained Embedding — i-vector . . . 31
4.1.1 Gaussian Mixture Modeling of Acoustic Features . . . 31
4.1.2 i-vectors . . . 37
4.2 Discriminatively Trained Embedding — x-vector . . . 41
4.2.1 Time Delayed Neural Network . . . 42
4.2.2 Original x-vector Network . . . 43
4.2.3 Variants of x-vector Network . . . 44
4.3 x-vectors vs. i-vectors . . . 46

5 Scoring 47
5.1 Cosine Similarity Scoring . . . 47
5.2 Linear Discriminant Analysis . . . 47
5.3 Probabilistic Linear Discriminant Analysis . . . 48
5.3.1 General PLDA . . . 49
5.3.2 Two Covariance PLDA . . . 50
5.3.3 Trial Scoring . . . 50

6 Multi-Conditional Training 52
6.1 Experimental Setup . . . 53
6.1.1 Evaluation Set . . . 53
6.1.2 System Description . . . 54
6.1.3 Results . . . 54

7 Speech Enhancement 56
7.1 Signal Enhancement Autoencoder . . . 57
7.2 Multi-Conditional Training vs. Speech Enhancement . . . 58
7.2.1 Experimental Setup . . . 59
7.3 Speech Enhancement and Other Discriminative Approach in SV . . . 64
7.3.1 MFCC i-vector System . . . 64
7.3.2 SBN-MFCC i-vector System . . . 64
7.3.3 x-vector Systems . . . 64
7.3.4 Results . . . 65
7.3.5 Analysis over the Range of Operating Points . . . 72

8 Discriminative Techniques in Generative SV 74
8.1 Stack Bottle-neck Features . . . 74
8.2 DNN Aligment . . . 75
8.2.1 Experimental Setup . . . 76
8.2.2 System Definition . . . 77
8.2.3 Results . . . 77
8.3 Discriminatively Re-trained i-vector Extractor . . . 81

8.3.1 T-matrix Re-estimation 81
8.3.2 T-matrix Factorization 86

9 Impact of Normalization on Language Robustness 90
9.1 Experimental Setup 90
9.1.1 Evaluation Sets 90
9.1.2 System Description 91
9.1.3 Normalization Cohorts 91
9.2 Results 91
9.3 Summary 94

10 Impact of Data on the Robustness of Discriminative Systems 96
10.1 Experimental Setup 96
10.1.1 x-vectors System 96
10.1.2 PLDA Augmentation Sets 97
10.1.3 Embedding Extractor Augmentation Sets 97
10.2 Results 97

11 Conclusions 100
11.1 Summary 100
11.2 Future Work 101
11.2.1 Possible Improvement of x-vectors 101
11.2.2 Possible Improvement of i-vectors 102

Bibliography 104

Nomenclature

ANN Artificial Neural Network

ASR Automatic Speech Recognition

BN Bottleneck

DCF Detection Cost Function

DCT Discrete Cosine Transformation

DET Detection Error Tradeoff

DFT Discrete Fourier Transformation

DNN Deep Neural Network

EER Equal Error Rate

EM Expectation-Maximization

GMM Gaussian mixture model

HMM Hidden Markov Model

HTPLDA Heavy-Tailed Probabilistic Linear Discriminant Analysis

LDA Linear Discriminant Analysis

LDC Linguistic Data Consortium

LLR Log-Likelihood Ratio

LVCSR Large Vocabulary Continuous Speech Recognition

MAP Maximum a Posteriori

MFCC Mel-Frequency Cepstral Coefficients

ML Maximum-Likelihood

NIST National Institute of Standards and Technology

NN Neural Network

PCA Principal Component Analysis

PLDA Probabilistic Linear Discriminant Analysis

PRISM Promoting Robustness in Speaker Modeling

SAD Speech Activity Detection

SBN Stacked Bottleneck features

SNR Signal to Noise Ratio

SRE Speaker Recognition

SRR Speech to Reverberation Ratio

SSNR Segmental Signal to Noise Ratio

TDNN Time Delay Neural Network

UBM Universal Background Model

VAD Voice Activity Detection

Chapter 1

Introduction

Speech is the most natural and common way of human communication. Humans are able to process the information contained in speech very quickly and efficiently. However, to this day, automatic processing of information contained in speech is still very challenging. Automatic speech processing consist of various techniques that usually focus on one specific application, such as automatic speech recognition (ASR), keyword spotting (KWS), Language recognition (LRE), gender identification (GID), or emotions detection. This work deals with *speaker recognition* (SRE), whose principle lies in extracting speaker-characterizing information from the input speech signal.. Nevertheless, speech conveys a wide range of information. Primarily, it carries linguistic information (a message content), it is also determines the language of the utterance, but it also carries information about the speaker's vocal tract (this is also related to his or her gender or age), it carries information about the speaker's emotional and health condition, or its geographical specifications, which indicate, in addition to the language, a dialect. All of the above mentioned types of information contribute to the identity of the speaker.

From the point of view of recorded speech, information about the acoustic environment, the recording hardware, or speech codding is also carried. All the mentioned aspects of speech and its recordings then affect the performance of the application. In most cases, this variability can be expected to have a negative effect. It is desirable that automatic speech processing systems be robust to the above mentioned phenomena, i.e. system performance stays unaffected by them.

The automatic speaker recognition represents the main topic of this work. Speaker recognition is the process of classifying a recording based on relevant information about the identity of the speaker in the recording. This task can be divided into two categories: *speaker identification* and *speaker verification*.

Speaker identification is a multi-class classification problem in which the speech is assigned to one of N classes (in the set of N speakers). It is necessary to distinguish between an *open* and a *closed* set of speakers identification. In the case of a closed set identification, we are sure that the given speech always belongs to one speaker from the given set. In the case of an open set identification, this condition no longer applies, and in this case, the system needs to assign the speech to one of N classes or say that it is none of the N knows speakers. *Speaker verification*, on the other hand, is a two-class (or detection) problem. The goal of the application is to decide whether the pair of (or more general two sets of) recordings come from the same speaker or not.

In both *speaker identification* and *speaker verification*, we can divide the task into two categories based on the awareness of the content of the utterance. The first category is a

text-dependent system, in which the system knows the content of the utterance (content is part of verification; phrases have to match). An example of such an application can be found, for example, in the banking sector, where a client can be asked in a phone conversation to enter a specific phrase or to enter a selected password. The second category is *text-independent* system, in which the system does not know or ignores the content of the speech itself. This approach finds its application e.g. in law enforcement, in which the content of speech is not known a-priori.

In this work, we focus on text-independent speaker verification and its robustness to the above-mentioned aspects of speech and recording (language, acoustic environment, or recording system) using discriminative techniques. Furthermore, this work sheds light on the transition from generative systems to discriminative systems. In order to give a fuller context, the work also describes techniques that had historically preceded this transition.

1.1 Speaker Verification System

Discrete waveform of speech is the raw input of an automatic speaker recognition system. The continuous form of the speech signal is processed by sampling (the sampling theorem must be observed) and quantization to obtain a digital form of speech. The sampling is usually performed at frequency of 8 kHz (*narrow-band*) or 16 kHz (*wide-band*).

In [Reynolds, 2002], the author presents that we can consider several types of information, which can be extracted from the speech signal (ordered from the closest to the raw audio signal):

- **Acoustic**: Spectral representation of speech that conveys vocal tract information.
- **Prosodic**: Features derived from prosody (pitch, energy, syllable length, tracks, etc.) (for a thorough overview, see [Kockmann, 2012]).
- **Phonetic**: Sequence of speaker-specific phonemes.
- **Idiolect**: Sequence of words.
- **Linguistic**: Linguistic pattern, charateristic for speaker's conversations.

It can be deduced that with a higher layer, the demands on the amount of training data for collection of relevant information increase. In this work, we focus mainly on the acoustic level of information.

As already mentioned, speaker verification (SV) is a two-class problem. The SV system evaluates a *trial*. The trial consists of two sets of utterances, called *enrollment* and *test*, where we decide on the accordance of the speaker and produce a score for the trial at hand. The problematics of computing a score for a verification trial is later described in Section 3.1. It represents a comparison of two hypotheses: whether the utterances come from the same speaker or not. In a modern SV, system for each utterance, an embedding (fixed-length vector) is extracted and subsequently used for comparison of speakers. A general scheme of such a system is outlined in Figure 1.1. For each speech signal, acoustic features are first extracted and silence is removed using voice activity detection (VAD)[1]. After the extraction of features, the embeddings are extracted and subsequently compared to produce scores.

[1]We describe this topic in detail together with score normalization and calibration in the following sections of this chapter.

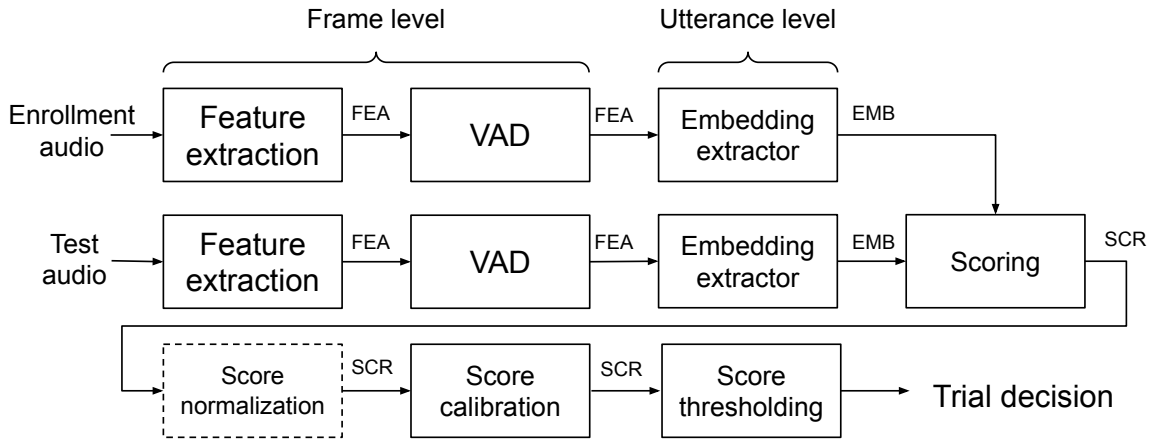


Figure 1.1: The generic speaker verification system pipeline divided into basic steps. The input is a pair of utterances (trial). The output of the system is a trial decision, whether it is a target trial (utterances come from the same speaker) or an impostor trial (utterances come from different speakers).

Separate chapters (4 and 5) are dedicated to the theory behind embeddings and their comparison as these constitute key parts of the system. The raw score obtained from the comparison of the embeddings is then calibrated (and normalized, but the normalization is not mandatory) and the final result is a decision whether the utterances on both sides of the verification trial come from the same speaker or not.

1.2 Features

1.2.1 Mel-Frequency Cepstral Coefficients

Mel-Frequency Cepstral Coefficients (MFCCs) have been introduced in [Davis and Mermelstein, 1980, Rabiner and Juang, 1993]. For SV, the MFCCs have been consistently among the best features for a very long time, and that is why they represent a baseline for every newly proposed feature extraction method. Figures 1.3 and 1.2 show the extraction pipeline and intermediate results before obtaining the MFCC feature vector for a single signal frame of speech. In the first step, the absolute value of short-term Discrete Fourier Transform (DFT) is used to acquire the amplitude of the spectrum. The spectrum is then divided into frequency bands using the Mel-filters [Rabiner and Juang, 1993].

Filter banks are an array of sub-band filters (see Figure 1.4), usually in some non-linear scale, such as Mel- or Bark-scale [Stevens et al., 1937, Zwicker, 1961], which separate signal into multiple components. The purpose of the Mel-scales is to mimic non-linearity of human hearing by being more discriminative at lower frequencies and less discriminative at higher frequencies.

The vector of band energies is computed as the weighted sum of squared values of the magnitude spectrum. The overall energy is computed as a sum of squared samples (usually used as 0th coefficient). Then a logarithm of the energies is taken to compensate for the dynamic range of the values and to emulate human perception of the sound loudness. In the last step, Discrete Cosine Transformation (DCT) is used to de-correlate and reduce the dimensionality of the vector to acquire the final vector of the MFCC coefficients.

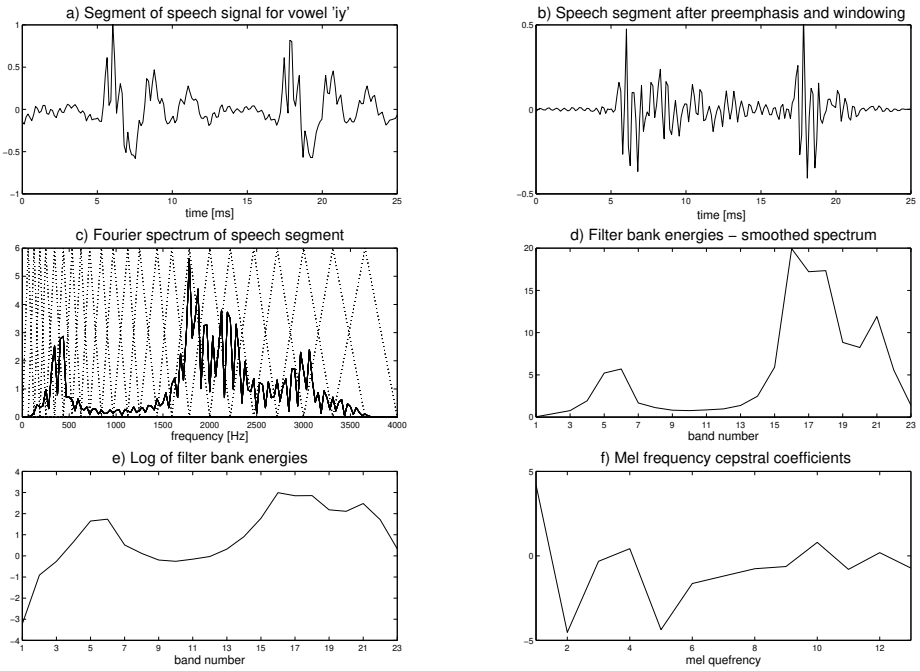


Figure 1.2: Outputs of individual steps of MFCC computation. (Source: [Burget, 2004])

1.2.2 Feature Derivatives

MFCC features are usually referred to as static features since they contain information only from a single frame. We can provide extra additional information to the feature vectors by extending them with feature derivative approximation [Furui, 1986]. Usually, first-, second-, and/or third-order derivative approximations are used (higher orders are rarely used), commonly referred to as delta (providing information about speech rate, also referred to as velocity), double-delta (or acceleration), and triple-delta coefficients [Mason and Zhang, 1991]. The commonly used formula for the first order derivation for a feature vector $\mathbf{c}$ in the frame k is a linear combination of the $\pm N$ surrounding feature vectors:

$$\triangle \mathbf{c}(k) = \sum_{j=-N}^{N} j\mathbf{c}(k-j) \tag{1.1}$$

where N is set to 2. Higher order-derivatives can be obtained by a recursive application of the formula 1.1 on the lower-order derivatives. The i-vector-based system (generative embedding, described in Chapter 4) usually achieved better results with first- and second-order derivatives.

1.2.3 Feature Normalization

Speech signals and their features contain distinct characteristics, which vary with recording conditions (noise, reverberation, type of microphone). It was observed that additive noise decreases the variance; and convolutive noise shifts the mean values of MFCC coefficients. To deal with this unwanted inter-session variability of features, a simple mean and variance normalization has been proposed [Boll, 1979, Openshaw and Masan, 1994]. The normalization is performed on the entire utterance with the assumption that the unwanted variability

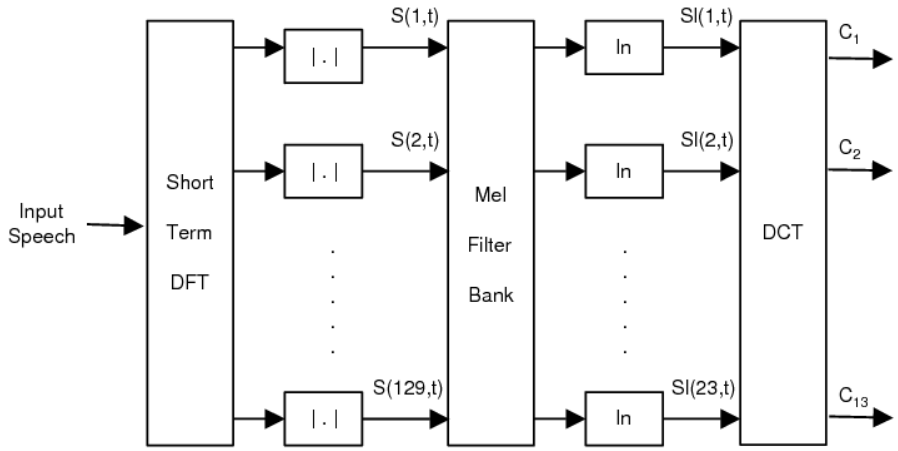


Figure 1.3: Block diagram showing steps of MFCC computation. (Source: [Burget, 2004])

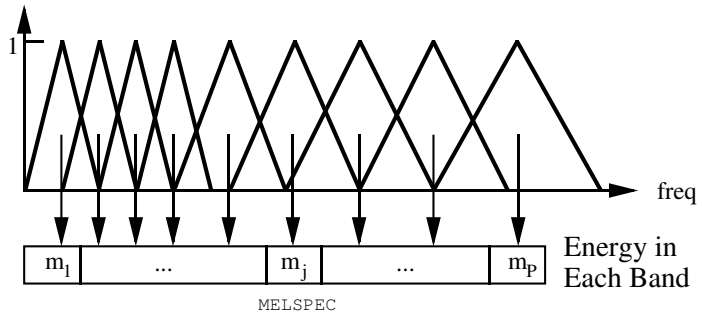


Figure 1.4: Mel-Scale Filter Bank. (Source: [Young et al., 2006])

is constant over the entire utterance. For the $k-th$ frame in utterance d, the normalized $i-th$ coefficient is computed in the following way:

$$\hat{c}_{d,i}(k) = \frac{c_{d,i}(k) - \mu_{d,i}}{\sigma_{d,i}}, \tag{1.2}$$

where the normalization parameters mean μ_d and standard deviation σ_d are estimated from the given utterance d.

Short-Time Normalization

In SV, it is also common for the normalization to be performed on short segments and applied locally. The feature vector being normalized is always in the center of the sliding window—this approach proved to be effective in some SV systems and it can cope with intersession variability. Usually, the size of the sliding window is set to 3-5s.

1.3 Voice Activity Detection

Voice Activity Detection (VAD), also known as Speech Activity Detection (SAD), is an important pre-processing step in most of the speech-processing applications. The task is

to find the presence of speech in a mixed signal of speech, noise, and silence. The output of this step is segmenting containing voice activity. Let us mention typical approaches to voice activity detection:

- **Energy Thresholding** as in [Kirill et al., 2009], in which the voice activity is classified on the basis of frame energy and adaptive threshold.
- **Gaussian Mixer Model classification** presented in [Ying et al., 2011], in which the model consists of two Gaussian components, which respectively describe the speech and non-speech log-power distributions.
- **Hidden Markov Models** as described in [Tatarinov and Pollák, 2004] in which the detector consists of 3-state HMM and thresholding of log-likelihood comparing two hypotheses for a frame with or without voice activity.
- **Artificial Neural Network** discriminator from [Arslan and Engin, 2019].

The output of the detector itself is usually binary (0 for silence, 1 for speech), but the output can differ based on the specific application of the system, i.e. distinguishing several classes based on the combination of speech, noise, and silence [Silovský, 2011].

In this work, we used VAD based on a hybrid of Artificial Neural Network and Hidden Markov Models. It is used as a phoneme recognizer trained on the SPEECHDAT Hungarian database [Matějka et al., 2006]. Strings of phonemes, re-labeled by two classes – silence (all models of silence) and speech (all valid phonemes) represent the output of the recognizer.

1.4 Score Normalization

The goal of the score normalization is to reduce unwanted log-likelihood ratio score variability. It leads to improved performance and calibration and a more reliable threshold setting (the setting is described in Section 3.1). Typically, the normalization step shifts and scales the score distributions for the individual models and/or benchmarks to allow for a single detection threshold setting. The shifts and scales are usually estimated using a set of utterances referred to as a normalization cohort, which usually consists only of impostor speakers. The scale and shift are applied as below:

$$s_{\text{norm}} = \frac{s - \mu}{\sigma}, \tag{1.3}$$

where score s is normalized by scale σ and shift μ estimated from the cohort.

We provide the reader with a quick overview of several types of normalization techniques, especially those used in this work (based on [Matějka et al., 2017]). However, the range of normalization techniques is wide. We could achieve significantly better accuracy with score normalization, such as Z-norm [Reynolds et al., 2000], T-norm [Auckenthaler et al., 2000], combinations of both (TZ-norm and ZT-norm) [Aronowitz et al., 2005] or other variants such as H-norm [Reynolds, 1997], D-norm [Ben et al., 2002], KL-T-norm [Ramos-Castro et al., 2007], S-norm [Kenny, 2010], normalized cosine similarity [Shum et al., 2010], speaker clusters [Apsingekar and Leon, 2011] and many others [Fortuna et al., 2004, Zigel and Cohen, 2003, Aronowitz and Aronowitz, 2010].

1.4.1 Z-norm

Zero score normalization employs impostor score distribution for enrollment file. It uses a cohort $\mathcal{E} = \{\varepsilon_i\}_{i=1}^{N}$ with N speakers which we assume to be different from the speakers in enrolment utterances e and test utterances t. The cohort scores are

$$S_e = \{s(e, \varepsilon_i)\}_{i=1}^{N} \tag{1.4}$$

and are formed by scoring enrollment utterance e with all files from cohort $\mathcal{E}$. The normalized score is then:

$$s(e,t)_{z\text{-}norm} = \frac{s(e,t) - \mu(S_e)}{\sigma(S_e)}, \tag{1.5}$$

where $\mu(S_e)$ and $\sigma(S_e)$ are mean and standard deviation of S_e. It compensates for the bias and scale between the enrollment model scores evaluated against the test data. This procedure is presented in Figure 1.5 as `STEP 1`. The advantage of Z-norm is the possibility to pre-compute the normalization statistics offline.

1.4.2 T-norm

Test score normalization is similar to Z-norm with the difference that it normalizes the impostor score distribution for the test utterance. T-norm can be expressed by:

$$S_t = \{s(t, \varepsilon_i)\}_{i=1}^{N} \tag{1.6}$$

$$s(e,t)_{t\text{-}norm} = \frac{s(e,t) - \mu(S_t)}{\sigma(S_t)}, \tag{1.7}$$

where $\mu(S_t)$ and $\sigma(S_t)$ are mean and standard deviation of S_t. The method is marked as `STEP 3` in Figure 1.5. The normalization parameters must be estimated online when scoring a test utterances t. It compensates for intersession variability between the tested utterance t and a set of speaker models.

1.4.3 ZT-norm

ZT-norm or TZ-norm use Z- and T-norm in series, and might use different cohorts for each step. By doing this, the scores are normalized with respect to both enrollment and test utterances. The procedure consists of performing `STEPS 1-3` in Figure 1.5.

1.4.4 S-norm

The symmetric normalization (S-norm) computes an average of normalized scores from Z-norm and T-norm. S-norm is symmetrical as $s(e,t) = s(t,e)$, while the previously mentioned normalizations depend on the order of e and t.

$$\begin{aligned} s(e,t)_{s\text{-}norm} &= \frac{1}{2} \cdot (s(e,t)_{z\text{-}norm} + s(e,t)_{t\text{-}norm}) \\ &= \frac{1}{2} \cdot \left(\frac{s(e,t) - \mu(S_e)}{\sigma(S_e)} + \frac{s(e,t) - \mu(S_t)}{\sigma(S_t)} \right) \end{aligned} \tag{1.8}$$

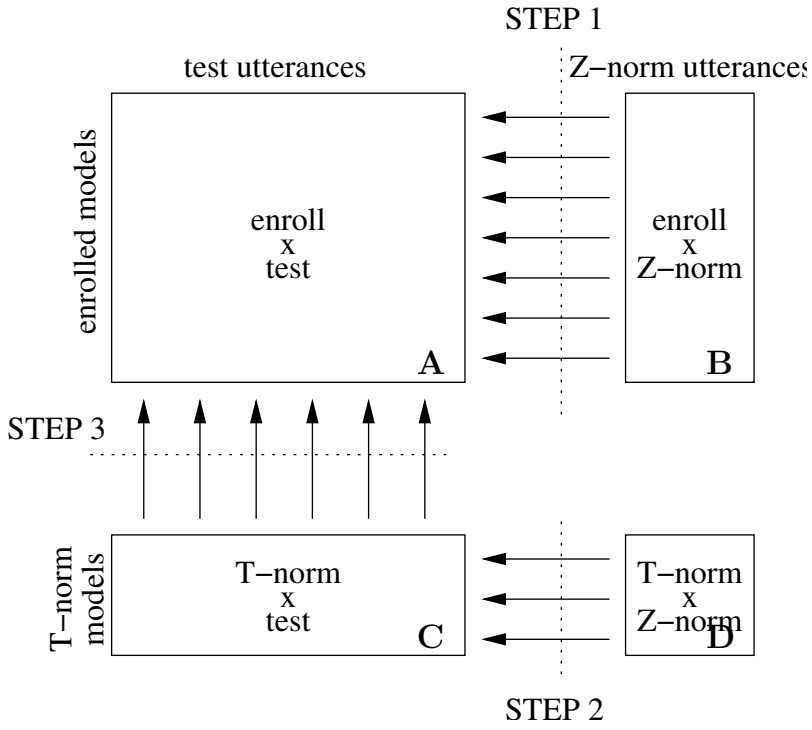


Figure 1.5: Application of ZT-norm. The boxes denote matrices of complete scores, i.e. all models against all scored utterances. (Source: [Glembek, 2012])

1.4.5 Adaptive Normalization

In adaptive T-norm [Sturim and Reynolds, 2005] or Top-norm [Zigel and Wasserblat, 2006], only part of the cohort is selected[2] to compute mean and variance for normalization. We call this selection adaptive, as the selected cohort might change for every speaker or utterance.

Two variants of adaptive cohort selection can be found in the literature: the adaptive cohort can be either selected to be X closest (most positive scores) files to the enrollment file $\mathcal{E}_e^{top}$, or, as in [Cumani et al., 2011], to the test file $\mathcal{E}_t^{top}$. We have to take into consideration that such cohorts differ for each enrollment utterance e or test utterance t respectively. The cohort scores based on such selections for the enrollment utterance are then:

$$S_e(\mathcal{E}_e^{top}) = \{s(e,\varepsilon)\}_{\forall \varepsilon \in \mathcal{E}_e^{top}} \tag{1.9}$$

and correspondingly for the test utterance t:

$$S_t(\mathcal{E}_t^{top}) = \{s(t,\varepsilon)\}_{\forall \varepsilon \in \mathcal{E}_t^{top}} \tag{1.10}$$

Later in this work, we investigate two variants of S-norm: the normalized score for the first one called **adaptive S-norm1** is

$$s(e,t)_{as\text{-}norm1} = \frac{1}{2} \cdot \left(\frac{s(e,t) - \mu(S_e(\mathcal{E}_e^{top}))}{\sigma(S_e(\mathcal{E}_e^{top}))} + \frac{s(e,t) - \mu(S_t(\mathcal{E}_t^{top}))}{\sigma(S_t(\mathcal{E}_t^{top}))} \right) \tag{1.11}$$

[2]Usually X top scoring or most similar files.

and the second variant, **adaptive S-norm2**, is defined as

$$s(e,t)_{as\text{-}norm2} = \frac{1}{2} \cdot \left(\frac{s(e,t) - \mu(S_e(\mathcal{E}_t^{top}))}{\sigma(S_e(\mathcal{E}_t^{top}))} + \frac{s(e,t) - \mu(S_t(\mathcal{E}_e^{top}))}{\sigma(S_t(\mathcal{E}_e^{top}))} \right) \tag{1.12}$$

1.5 Calibration

The term calibration stands for properties of scores provided by the system that can be interpreted as log-likelihood ratios, but also for the process of learning the score transformation function to obtain proper log-likelihood ratio. We expect the system to provide proper log-likelihood ratios; in practice, it is often not the case for many systems and their outputs have to be transformed into log-likelihood ratios.

The process of calibration is described in [Brümmer and du Preez, 2006]. It is typically realized with a monotonically increasing transformation function, usually a linear function:

$$f(s) = as + b, \tag{1.13}$$

where s is the recognition score, a and b are the parameters of the transformation function, and $f(s)$ is the calibrated score in the proper form of a log-likelihood ratio. As $f()$ is monotonous, it will not change order of scores and discriminability of the detector. The function parameters are usually found by optimizing a cross-entropy objective function of a logistic regression on a supervised development set. The objective function is defined as [Brümmer et al., 2007]:

$$\begin{aligned} Q(a,b,P_{tar}) = {} & \frac{P_{tar}}{N_{tar}} \sum_{t \in tar} \log_2 \left(1 + \exp^{(-f(s_t) - \text{logit}P_{tar})}\right) \\ & + \frac{(1-P_{tar})}{N_{non}} \sum_{t \in non} \log_2 \left(1 + \exp^{(f(s_t) + \text{logit}P_{tar})}\right), \end{aligned} \tag{1.14}$$

where P_{tar} represents the parameter of the objective function and allows us to tune optimization to the target operating point independently on P_{tar} in the calibration development set. This action is equivalent to the threshold selection described later in Section 3.1. If the operating point of the application is unknown, usually, the value $P_{tar} = 0.5$ is used.

The calibration process does not consider the currently processed data (in contrast to some normalization techniques such as t-norm or s-norm). The statistic distribution of development data can differ from the distribution of evaluation data. In this case, incorrect calibration function parameters may be found; such parameters can lead to a wrong calibration (setting a threshold far from an ideal value).

1.6 Motivation and Contribution

The work on this thesis began during the summer SV workshop at Royal Turin Polytechnic in 2015, which focused on short-duration speaker recognition and the use of discriminative techniques in SV, such as DNN alignment or speech enhancement. Inspired by [Plchot et al., 2016a], my focus was set on data augmentation and its impact on speech enhancement performance and robustness [Novotný et al., 2018a].

With the transition of SV systems from generative to discriminative ([Snyder et al., 2017]), it was expected that speech enhancement would become unnecessary (because of the significantly higher robustness of the discriminative systems). This hypothesis was explored

in [Novotný et al., 2019c], where we showed various scenarios in which the speech enhancement did actually significantly improve the SV results of the new systems.

Subsequently, I focused on the general robustness of the SV systems, especially against noise and reverberation, which was studied in [Novotný et al., 2018b], and on robustness to language, which was studied in [Novotný et al., 2016]. In [Matějka et al., 2017], we explored the robustness to language from the perspective of normalization. However, the discriminative systems proved to be very demanding in terms of the training data. This property motivated us for an attempt to combine the generative and discriminative approaches, and subsequently exploit the advantages of both methods [Novotný et al., 2019b, Novotný et al., 2019a].

My work analyzes the individual steps of a SV system and suggests the techniques on how to increase the robustness in the given step.

1.6.1 Claims

The goal of this work is to investigate a state-of-the-art text-independent speaker verification system and to improve its individual parts using a discriminative approach. The main contributions of this work are summarized in the following points:

- **Analysis of Speech Enhancement with Neural Networks (NN)**: We have systematically investigated an NN speech enhancement and its effect on performance as a pre-processing step in case of generative and discriminative speaker verification systems (Chapter 7).
- **Including discriminative techniques in generative SV**: We have designed and plugged discriminative techniques into generatives SV systems (Chapter 8).
- **Discriminative training of an i-vector extractor**: We propose to use a discriminative approach to the i-vector extractor generative model re-training as a way of improvement of the original generative model (Section 8.3).
- **Extension of the discriminative training of the i-vector extractor with factorization**: The proposed extension of the discriminative i-vector model training takes memory requirements into account, which significantly decreases the number of model parameters (Section 8.3.2).
- **Analysis of data and its impact on the robustness**: We have analyzed the performance of discriminative speaker verification systems as a function of amount of training data (Chapter 10).

1.6.2 Structure of the Thesis

- **Chapter 2** describes the factors influencing the performance of SV, which we consider in setups improving its robustness.
- **Chapter 3** introduces the used evaluation data and the evaluation metrics of SV performance. Secondly, this chapter contains a description of the data augmentation process.
- **Chapter 4** outlines the basic concept of acoustic modeling and embeddings as features for speaker modeling.

- **Chapter 5** presents (a possible approach to) embedding scoring.
- **Chapter 6** introduces the concept of multi-conditional training as the primary approach used in improving the robustness of a system.
- **Chapter 7** presents the speech enhancement based on NN as a pre-processing step on a signal level.
- **Chapter 8** studies the discriminative technique used to improve the formerly generative SV system.
- **Chapter 9** discusses the score normalization techniques and their impact on the SV robustness.
- **Chapter 10** provides an analysis of the SV system performance and robustness based on the training data composition.
- **Chapter 11** concludes this thesis.

Chapter 2

Factors Influencing SV Performance

SV systems have to deal with various factors that decrease the final system performance. Three main factors determine the performance of each SV system:

- **system topology**: used acoustic feature, type of embedding (including embedding model itself), or the verification backend.
- **training data**: total amount of speakers in the training set, number of utterances per speaker, or the quality of audio records.
- **operating environment (test data)**: known/unknown target environment, also referred to as in-domain and out-of-domain, respectively.

In an ideal case, we have a well-selected system topology trained on data from the target source without any other unwanted variability. In reality, we have to settle for a compromise: the topology is subjected to computational capacity, the training dataset is based on the available data, and test data are usually unknown.

As the training data usually differ from application (test) data, a phenomenon called *domain-mismatch* can occur. The term **domain** stands for a wide range of attributes of the data, such as:

- Transmission channel (related to a codec and a sampling frequency)
- Language
- Utterance length
- Acoustic environment (i.e. noise, reverberation)
- Recording system (i.e. recording hardware, recording setup)

The long-term goal of the research in the field of speaker recognition is to design a system invariant to a domain, and to achieve optimal performance even in an unknown environment.

Usually, the application domain is at least partially known (i.e. we know the language, the sampling frequency, or a codec). When designing a training set, we aim at approximating the application domain. This can be achieved by an appropriate selection of the available data, and/or their simulation (more details about simulation in Section 3.3).

Next, we focus on each attribute of the domain and its effect on the speaker recognition pipeline.

2.1 Transmission Channel

A transmission channel has a subset of inseparable properties:

- The transmission channel itself (landline, cellular, radio, type of microphone)
- A codec, which usually tries to balance the quality of the audio and the bit-rate.
- A sampling frequency, which indicates the frequency range of the recording (given by the sampling theorem).

The sampling frequency is a necessary key attribute that we have to know for the system design. For a long time, the SV systems were dominated by a sampling frequency of 8000 Hz. This frequency is given by the transmission channel, and for many years the SV was dominated by the telephone channel. The landline telephone band typically operates in the range of 300–3400 Hz, a standard based on Bell Labs studies of the requirements for intelligible speech dating back to the 1920s.

At the time of writing of this work, the interest in 8000 Hz audios is declining and recordings with the sampling frequency of 16000 Hz are used more and more often.

This transformation was caused by technological progress. Because of innovative transmission technologies, the capacity of the transmission band has expanded as well as the increased computing capability, which can process larger amounts of data.

The codec determines the coding of the signal itself for the purpose of storage or transmission. The codec affects the signal with its assumptions about the input. It can be either lossy or lossless. It can expect speech, music, or general signals.

We deal with the issue of robustness to the transmission channel in almost all experiments presented in Chapters 6–10. The evaluation benchmarks are divided into the following groups: telephone channel, interview, and simulated data.

2.2 Language

The next variability we have to deal with is language variability. In the best case, the language variability is contained in within-speaker variability. For this case to apply, each speaker must speak multiple languages in the training set. Unfortunately, this is very rare and most speakers usually know just a single language. This results in language variability leaking to between-speaker variability. However, there are methods to compensate for this (i.e. [Glembek et al., 2014]). Robustness to the language is discussed mainly in Chapters 8 and 9.

2.3 Acoustic Environment

The acoustic environment provides an infinite variety of signal degradations. It is the main type of signal degradation which we focus on in this work. We can approximate an acoustic environment by two types of signal degradation: noise and reverberation. Usually, in literature, the term „noise“ represents all types of signal degradation (including reverberation). In this work, we distinguish between noise and reverberation. The term noise stands for the

unwanted additive signal. The term reverberation represents the convolutional distortion (i.e. the acoustic effect of a room). The issue of noise and reverberation is mainly discussed in Chapter 7.

2.3.1 Additive Noise

The additive noise is a basic and elementary model of distortion of a wanted signal $x(t)$ with an unwanted signal $n(t)$ using an additive operation:

$$s(t) = x(t) + n(t) \tag{2.1}$$

In the case of additive noise, we can measure the degree of distortion of the original signal as a signal-to-noise ratio (SNR):

$$SNR = 10 \log_{10} \left(\frac{P_s}{P_n} \right), \tag{2.2}$$

where P_s is the power of signal, P_n is the power of noise. The calculation above is overly general and not very suitable for our use. As mentioned earlier, we can divide the signal into speech and silence segments (non-speech in general). We are primarily interested in the speech part of the recording. If we measure SNR on the entire recording, including segments of silence, these silence segments distort the resulting measurement.

Therefore, when we mention SNR in this work, we refer to the segmental-signal-to-noise ratio (SSNR) variant:

$$SNR = 10 \log_{10} \frac{\sum_{i=1}^{N} e_s[i] \cdot vad[i]}{\sum_{i=1}^{N} e_n[i] \cdot vad[i]}, \tag{2.3}$$

in which e_s and e_n are the energy of the signal and the noise in i-th frame), in which the segments are weighted according to VAD. As a result, we only get the SNR between the noise and the speech, and we have an accurate indication of the distortion that enters the system (frames classified as silent are discarded from further processing).

2.3.2 Reverberation

The reverberation effect occurs under conditions where the speaker or other signal source is far from the recording microphone. Sound waves propagate through the environment, bounce off of obstacles and walls, and reach the microphone slightly later than a wave flying straight to the microphone. The reflections are cumulatively repeated until the waves lose all energy and are absorbed.

Reverberation can be seen as a convolutional model, in which the convolutional kernel is determined by the room impulse response (RIR):

$$s(t) = h(t) \star x(t), \tag{2.4}$$

where $h(t)$ is the RIR and $x(t)$ is the wanted signal. An example of a room impulse response and its parts are shown in Figure 2.2.

If the microphone is placed close to the source, mostly direct signal with a tiny portion of reverberation is recorded. In the case of a close-talk microphone, we can consider the

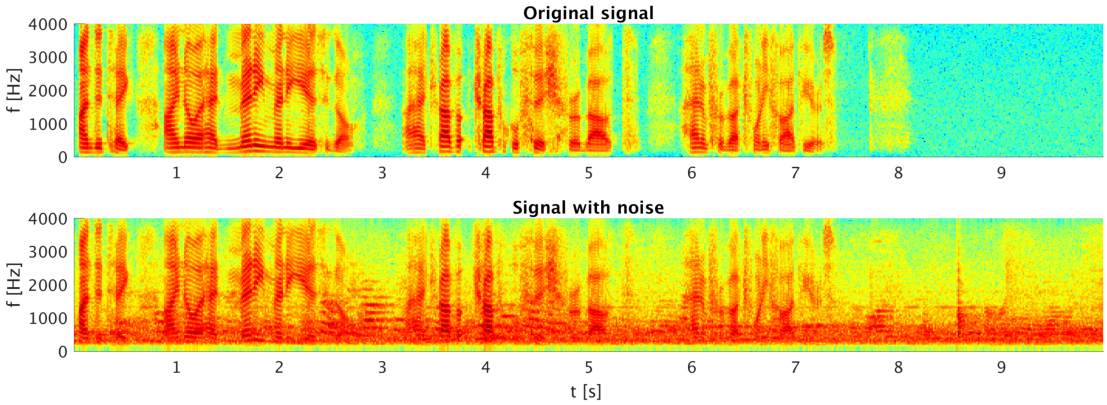
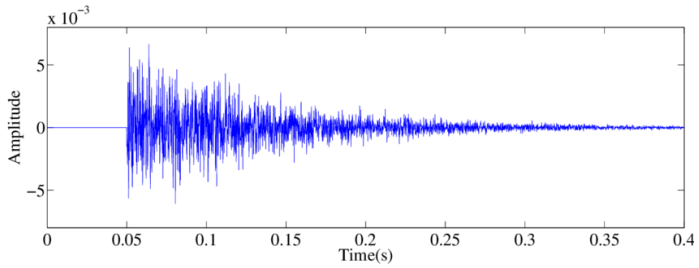


Figure 2.1: Example of additive noise. From the top, the spectrogram of the original clean signal and the spectrogram of the signal with noise with signal-to-noise ratio 10 dB.

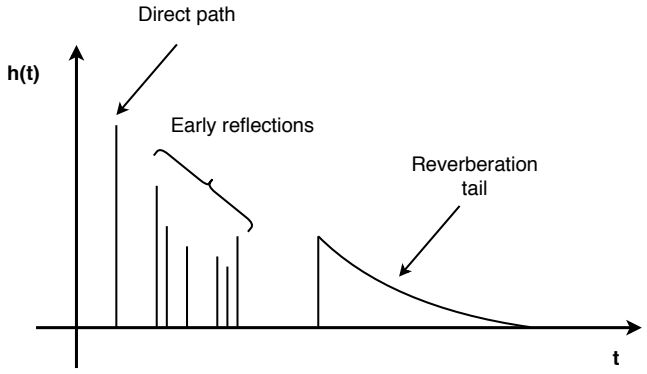
reverberation effect to be negligible. In the case of increasing distance between the microphone and the sound source, the energy of the direct signal decreases and the energy of the reflected waves returning to the microphone remains the same. Reverberation and signal distortion are also determined by the used microphone and its orientation in relation to the source. For example, a hypercardioid microphones is less affected by reverberation when pointed at a source because its direct polar pattern does not capture signals reflected from other directions (see Figure 2.3).

2.4 Recording System

The term recording system stands primarily for the hardware that was used to make the recording. As mentioned above, in the past, we worked mainly with telephone data. However, advances in technology and the proliferation of mobile devices have greatly expanded this variability. Different recording devices are equipped with various types of microphones (carbon microphones, electrodynamic microphones). Microphones vary in their polar pattern and in their frequency characteristics. Differences concerning the devices themselves can include different location, count, or orientation. It is also possible to estimate the position of the speaker relative to the recording system. As mentioned above, close-talk microphones can significantly reduce some of the negative effects of acoustic environments. Problems generally arise as the distance between the speaker and the microphone increases. In this work, we focus on single-channel recordings. It is beyond the scope of this work to analyze recordings from multiple microphones or their impact on SV robustness.



(a)



(b)

Figure 2.2: (a) Example of real room impulse response in time domain. (b) Description of the main parts of the room impulse responses.

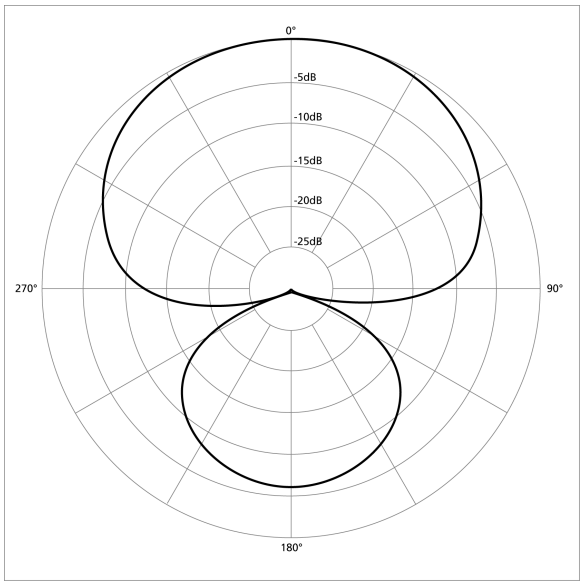


Figure 2.3: An example of hypercardioid microphone's polar pattern. Polar pattern refers to the sensitivity of a given microphone to sounds arriving from different angles to its central axis. (Source: www.teachmeaudio.com)

Chapter 3

SRE Datasets and Evaluation Metrics

Let us now get acquainted with the data and evaluation metrics that are used in this work. We also introduce the reader to the types of errors in SV systems and data augmentation, which represents a key attribute to building of robust systems.

3.1 Speaker Verification Evaluation

SV is a two-class problem. We present SV trials t to the SV systems. In general, a trial t consists of two sets of utterances, but for simplicity, we use two utterances $t = (d_1, d_2)$, enrollment, and test. The classes represent two possible situations:

- If the enrollment and the test utterances come from the same speaker, we refer to such a situation as a *target trial*.
- If the enrollment and the test utterances do **not** come from the same speaker, we refer to such a situation as a *non-target trial* or an *impostor trial*.

To evaluate the system, a labelled set of trials $\mathcal{T}$ is required. For each trial, we need to have target/non-target label. The goal of the system is to assign the correct label to the trial. During this process, two types of errors can occur:

- **False-alarm**: A non-target trial is incorrectly classified as a target trial.
- **Miss-detection**: A target trial is incorrectly classified as a non-target trial (usually simply referred to as Miss).

We can estimate the miss and false-alarm rates as probabilities $p(miss|\mathcal{T})$ and $p(fa|\mathcal{T})$:

$$p(miss|\mathcal{T}) = \frac{N_{miss}}{|\mathcal{T}_{tar}|}, \tag{3.1}$$

$$p(fa|\mathcal{T}) = \frac{N_{fa}}{|\mathcal{T}_{non}|}, \tag{3.2}$$

where $|\mathcal{T}_{tar}|$ and $|\mathcal{T}_{non}|$ are the numbers of target and non-target trials, and N_{miss} and N_{fa} are the numbers of missed and false alarm detections made by the SV system.

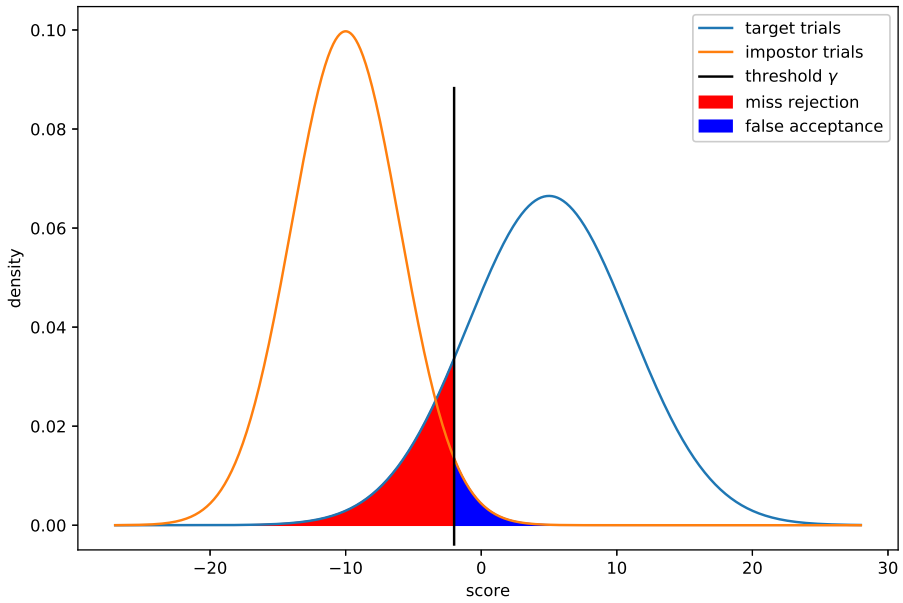


Figure 3.1: Distribution of scores for target and non-target trials. The red area on the left of the decision threshold represents measure P_{miss}, and the blue area represents measure P_{fa}.

The raw output of the verification system is usually a score. The score should reflect the confidence of the system in the decision, having a higher value for a *target trial* and vice-versa for a *non-target trial*, preferably as log-likelihood ratio of these two hypotheses: $\mathcal{H}_s$ for same speakers, $\mathcal{H}_d$ for different speakers. The score s for trial t is given as:

$$s = \log \frac{p(t|\mathcal{H}_s)}{p(t|\mathcal{H}_d)}. \tag{3.3}$$

The score can be converted into a hard decision by *thresholding*. By moving the threshold τ, the balance between two types of errors (see Figure 3.1) changes and allows the user to choose the operating point of the system. The motivation to choose different operating points depends on the system application. For example, a bank has an incentive to keep the number of False-alarms low, as this error has a much higher cost than a Miss.

3.1.1 Detection Error Tradeoff Plot

It is always desirable to know the individual systems' performance across a wide range of operation points (thresholds). In the SRE community, the Detection Error Tradeoff (DET) plot is commonly used [Martin et al., 1997]. It is an alternative graph to the commonly used Receiver Operation Characteristic (ROC) curve. The DET curve represents the miss detection probability as a function of a false alarm probability. DET plot (example shown in Figure 3.2) has both axis transformed by probit function:

$$probit(p) = \sqrt{2}\ \mathrm{erf}^{-1}(2p - 1), \tag{3.4}$$

where p is a value of the P_{fa} or P_{miss} and erf^{-1} is the inverse function to the error function erf. Therefore, the DET plot represents the dependency of $probit(P_{miss})$ on $probit(P_{fa})$.

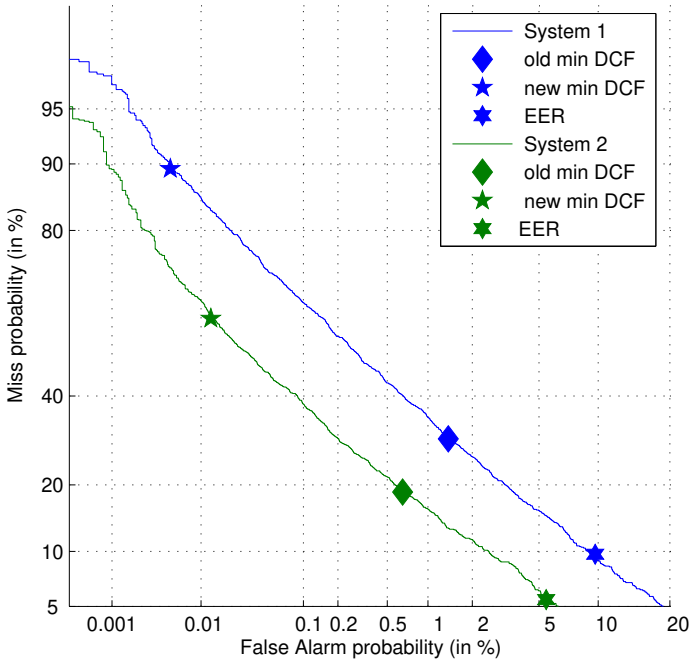


Figure 3.2: Example of the DET curve comparing two systems. The markers (in each DET) correspond to $\mathrm{DCF}^{\mathrm{min}}_{\mathrm{new}}$(★), $\mathrm{DCF}^{\mathrm{min}}_{\mathrm{old}}$(♦), and EER (✸). (Source: [Glembek, 2012])

Based on the values P_{fa} and P_{miss} for a selected operating point, we can plot the individual operating point of interest and the region of statistical significance. The statistical significance is often determined by Doddington's rule of 30 (for detailed interpretation, see [Doddington et al., 2000]), stating that for a meaningful evaluation, it is necessary to have at least 30 misses and at least 30 false alarms.

The DET plot is not dependent on calibration because it depends only on score order and not on actual values. Assuming the distribution of score for target and non-target trials are Gaussian, the DET curve will be approximately linear. The curve located closer to the origin reflects better performance of the system.

3.1.2 Equal Error Rate

Comparing systems by DET curves can be inconvenient. Sometimes, a single scalar summarizing system performance can be desirable. Equal Error Rate (EER) is a single number evaluation of the performance of a biometric system. EER is defined as a location on the DET curve in which P_{miss} and P_{fa} are equal. The EER value is independent on calibration and it can be shown [Brümmer, 2010] that this point acts as a scalar summary of the whole DET curve. Its value indicates how close is the DET curve to the origin.

EER can give an approximate comparison between systems. Although EER might seem an attractive option, it is not very practical in real-world applications. A real-world application usually operates in the region of low false alarm or low miss rate.

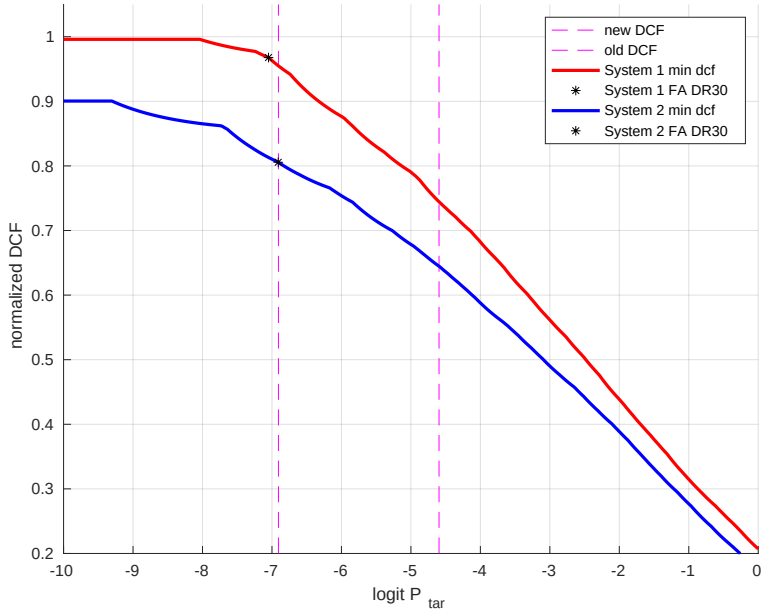


Figure 3.3: Plot of $\mathrm{DCF}^{\min}$as a function of effective prior. Intersection of $\mathrm{DCF}^{\min}$curves with vertical dashed violet lines correspond (from left to right) to the $\mathrm{DCF}^{\min}_{\text{new}}$and $\mathrm{DCF}^{\min}_{\text{old}}$.

3.1.3 Detection Cost Function

NIST introduced Detection Cost Function (DCF) as a metric for evaluation of verification systems, that focuses on a particular operating point of interest. It is the primary metric in NIST's Speaker Recognition Challenge series (Section 3.2.1).

It is designed to consider the overall cost based on the two types of error. It is defined as a weighted sum of the false-alarm probability and the miss-detection probability:

$$C_{\text{Det}} = C_{\text{miss}} P(\text{miss}|\mathcal{T}, \tau) P_{tar} + C_{\text{fa}} P(\text{fa}|\mathcal{T}, \tau) P_{non} \tag{3.5}$$

with

$$P_{non} = 1 - P_{tar} \tag{3.6}$$

where C_{miss} and C_{fa} are the relative costs of the detection errors, and P_{tar} and P_{non} are the prior probabilities for the trial coming from the same speaker and coming from different speakers, respectively. The triplet $\langle C_{\text{miss}}, C_{\text{fa}}, P_{tar} \rangle$ defines the target operating point considering the desire an application for which the system is evaluated.

To make the DCF measurement more intuitive, C_{Det} is further normalized by C_{Default} to allow the comparison of difficulty of various evaluation sets. C_{Default} is the best *a-priori* cost that could be obtained without processing the input data and setting all trials to either same speaker or different speakers, whichever is smaller:

$$C_{\text{Default}} = \min \begin{cases} C_{\text{miss}} P_{tar} \\ C_{\text{fa}} P_{non} \end{cases} \tag{3.7}$$

and

$$C_{\text{Norm}} = C_{\text{Det}} / C_{\text{Default}}. \tag{3.8}$$

Table 3.1: NIST DCF parameters for SRE08 and SRE10.

	C_{fa}	C_{miss}	P_{tar}
DCF_{old}	10	1	0.01
DCF_{new}	1	1	0.001

Table 3.2: NIST DCF parameters for SRE16.

ID	C_{fa}	C_{miss}	P_{tar}
1	1	1	0.01
2	1	1	0.005

The cost is computed from hard decisions independently of the threshold selection. The threshold should be selected in such way as to minimize the cost computed on the development set. Minimum possible DCF, referred to as min-DCF (see Figure 3.3), can be computed by setting the optimal threshold for the given test set:

$$\min C_{\text{Det}} = \min_{\tau} \left[C_{\text{miss}} P(\text{miss}|\mathcal{T}, \tau) P_{tar} + C_{\text{fa}} P(\text{fa}|\mathcal{T}, \tau) P_{non} \right]. \tag{3.9}$$

The cost computed from the actual hard decision is referred to as act-DCF. The difference between the act-DCF and min-DCF it referred to as a calibration loss, and it reflects how well the system is calibrated. The calibration was discussed in Section 1.5.

Until 2008, NIST used a metric referred to as "old-DCF". In 2010, NIST introduced a new metric referred to as "new-DCF". Parameters for these metrics are shown in Table 3.1. In 2016, NIST redefined the primary metric again. Final $C_{primary}$ is computed as an average of two operating points (defined in Table 3.2):

$$\text{C}_{primary} = \frac{C_{\text{Det}_1} + C_{\text{Det}_2}}{2} \tag{3.10}$$

3.2 Datasets

When designing an SV system, it is always necessary to consider the target application and adapt the training and test data accordingly. Usually, we can divide the data used in the development of the SV system into three parts: **training** (on which the system is trained), **development** (to monitor performance and allow the potential adjustment of parameters), and **evaluation/test** set (final evaluation). The evaluation set can usually be divided into several benchmarks, focusing in detail on selected aspects of the evaluation (such as cross-language trials and noisy environment trials). Evaluation sets also carry evaluation metrics, and metrics can differ between sets. We have selected several metrics to evaluate the performance of later presented systems.

3.2.1 NIST

The NIST (US National Institute of Standards and Technology) has been organizing speaker recognition evaluations since 1996. The NIST has the following objective goals for evaluations series:

- Effective measuring of system-calibrated performance of the current state of technology.
- Provide a standard test benchmark that helps the research community to explore the new ideas in speaker recognition.
- Support community in their development of advanced technologies.

The evaluations are intended to be of interest to all researchers working on the general problem of text-independent speaker recognition.

NIST databases released for each evaluation have gradually become a standard benchmark in the field of text-independent speaker recognition. Each evaluation consists of a different set of common evaluation conditions designed to explore the performance of the system under specific constraints. The constraints are usually designed to group the data according to some characteristics, such as language, dialect, channel, nominal length, number of utterances per trial side, speaking style, or the number of speakers in the test side of the trial.

In this part of the thesis, we describe in more detail the evaluations from years 2010 and 2016, which are used in this work. For a detailed summary of other years, we recommend reading e.g. [Plchot, 2014].

NIST SRE 2010

NIST SRE 2010 (NIST Speaker Recognition Evaluation [NIST, 2010]) has long been considered one of the most popular sets of benchmarks. The evaluation in NIST SRE 2010 focuses on a more severe penalization of false alarms, which was achieved by decreasing the cost of miss and target trial ratio. The ratio of the cost of false alarm to miss is 1000:1, i.e. a false alarm is 1000 times more serious than a miss. In order to obtain statistically significant results in case of a low false-alarm rate, the number of trials was significantly increased. The detection cost function (discussed in Section 3.1.3) serves as a primary metric.

In evaluation plan [NIST, 2010], NIST defined nine benchmark scenarios with the following characteristics:

1. All trials involving interview speech from the same microphone in training and test
2. All trials involving interview speech from different microphones in training and test
3. All trials involving interview training speech and normal vocal effort conversational telephone test speech
4. All trials involving interview training speech and normal vocal effort conversational telephone test speech recorded over a room microphone channel
5. All different number trials involving normal vocal effort conversational telephone speech in training and test
6. All telephone channel trials involving normal vocal effort conversational telephone speech in training and high vocal effort conversational telephone speech in test
7. All room microphone channel trials involving normal vocal effort conversational telephone speech in training and high vocal effort conversational telephone speech in test

8. All telephone channel trials involving normal vocal effort conversational telephone speech in training and low vocal effort conversational telephone speech in test

9. All room microphone channel trials involving normal vocal effort conversational telephone speech in training and low vocal effort conversational telephone speech in test

NIST SRE 2016

NIST SRE 2016 (SRE16, [NIST, 2016]) is focused on telephone conversations. Unlike in previous years, data were collected outside of the North America. The organizers have once again introduced two training scenarios: A fixed training scenario with a strictly defined training set, which allows better cross-system comparison, and an open training scenario, in which all limits from the fixed scenarios are removed. Languages used in the evaluation are Tagalog and Cantonese (referred to as the major languages) and Cebuano and Mandarin (referred to as the minor languages). The available development set was composed of both major and minor languages, while the test set only contained data from the two major languages. In SRE16, NIST defines only same-sex and same-language trials. However, on the contrary to the previous SREs, gender labels were not provided. SRE16 benchmark contains both single-enrollment and multiple-enrollment (three segments) trials, in which approximately 60 secs of speech segments are provided to build the model of the target speaker.

3.2.2 Fisher English

Fisher English [Cieri et al., 2004] is a collection of telephone speech collected by LDC in 2003. The dataset was created to address a need to build robust Automatic Speech Recognition (ASR) systems. Many participants made a few short calls speaking to other participants, where called participants typically did not know about assigned topics. This approach maximizes inter-speaker variability, vocabulary breadth, and it also increases formality. The dataset contains 11,699 telephone conversations, each lasting up to ten minutes.

3.2.3 Switchboard

Switchboard 2 Phase II [Graff et al., 1999] was released in 1999. It consists of 4,472 telephone conversations involving 679 participants. Participants were recruited from US midwestern college campuses. Each recruit was asked to participate in at least ten five-minute phone calls.

Switchboard 2 Phase III [Graff et al., 2002] was collected between 1997 and 1998 (released in 2002). The collection was focused primarily in the American South, and it consists of 2,728 calls from 640 participants (292 Male, 348 Female), all native English speakers. Both of these corpora consist of landline calls only.

Switchboard Cellular Part 1 [Graff et al., 2001] was collected between 1999 and 2000 (released in 2001), mainly focusing on GSM cell phone technology. The dataset consists of 1,309 five- to six-minute calls, or 2,618 sides (1,957 GSM), from 254 participants (129 Male, 125 Female), under varied environmental conditions.

Switchboard Cellular Part 2 [Graff et al., 2004] was collected by LDC in 2000 and released in 2004. The dataset consists of 2,020 five-six minute conversations on cellular phones, or 4,040 sides (2,950 cellular, 2,405 female, 1,635 male), from 419 participants.

3.2.4 PRISM

The PRISM (Promoting Robustness in Speaker Modeling) is a large speaker recognition evaluation set based on NIST SRE data from 2005 to 2010. The set was presented in [Ferrer et al., 2011b] and [Ferrer et al., 2011a]. Compared to NIST SRE data, the PRISM set is extended with additional types of variations, namely noise, reverberation, language, channel type, speech style, and vocal efforts.

The evaluation set was created using data from NIST SRE from 2005 to 2010 (specifically SRE 2005, 2006, 2008, and 2010). NIST SRE 2004, together with Fisher English and Switchboard, is also included in the dataset, but for training purposes only. The set is divided into several subsets designed to evaluate the performance of a SV system under different conditions: language, noise, reverberation, speech style, channel, and vocal efforts. Even though PRISM is designed as a separate set, NIST SRE 2010 for 1-side and 8-side training is included.

The noise and reverberation subsets are created by adding real noise and reverberation to the data from NIST 2010 and 2008. To minimize negative effects of tracks other than noise and reverberation, only clean microphone data were selected from telephone and interview benchmarks. Also, a mixture of 15 noise samples (from bars, cafeterias, offices, and airports) from `Freesound.org` was collected to be used as noise. The samples were selected as free of from single-speaker foreground speech and sound artifacts. Afterwards, the noise samples were mixed with the clean segments at 8, 15, and 20 dB SNR using the FaNT Tool [Hirsch, 2005]. To prevent overly positive scenarios, different noises were added to training, enrollment, and test samples. The reverberation using different reverberation times of 0.3, 0.5, and 0.7 seconds is added to the clean signals. In the case of reverberation, a set of candidate rooms was generated using the RIR tool [McGovern, 2009]. The tool allows modeling of room impulse response based on different aspects such as room size, speaker and microphone location, wall, floor and ceiling reflection coefficients, speed of sound, etc. The rooms were modeled to cover the usual configuration of size, speaker-microphone location and reflectivity. Only the configurations resulting in reverberation time close to 0.3, 0.5 and 0.7 seconds were used. In total, twelve rooms (four for each reverberation time) were modeled for training, three for enrolment, and three for test (one for each reverberation time for each case). The `fconv` tool[1] was used to convolve room impulse response with the original audio signals.

The language subset is based on data from multiple corpora designed for NIST SRE evaluations and focusing on evaluating speaker recognition performance in multiple languages, including same-language and cross-language trials. It was crafted from the NIST SRE 2005–2008 datasets by selecting 500 speakers for whom there exists at least one session in a language other than English. Additional 300 speakers (that appear only in English conversations) were added from the NIST SRE 2010. The trials were created as a Cartesian product of all sessions, resulting in 3590/130880 male, and 6304/297683 female target/non-target trials, respectively. Note that a half of the trials is still in English.

3.2.5 SITW

The SITW [McLaren et al., 2016] dataset is a large collection of realistic speech from individuals across a wide array of challenging acoustic and environmental conditions. These audio recordings do not contain any artificially added noise, reverberation or other arti-

[1] https://www.mathworks.com/matlabcentral/fileexchange/5110-fast-convolution

facts. This database was collected from open-source media. The *sitw-core-core* benchmark comprises audio files, each of them containing a continuous speech segment from a single speaker. The enrollment and test segments contain between 6 and 180 seconds of speech each. SITW also introduced other evaluation benchmarks called *sitw-assist-core/multi*, *sitw-assistclean-core/multi* and *sitw-assist/core-multi*, in which the participants are motivated to focus on speaker diarization. In *sitw-assist-core/multi*, the Person of Interest (POI) segment is partially in the enrollment part of the trial, participants can use it to find the other POI segments in the enrolment audio. In *sitw-assist/core-multi*, the information about multiple speakers in test audio is known. For our purposes, we are using *sitw-core-core* only.

3.2.6 BUT Retransmitted Data

To evaluate the impact of room acoustics on the accuracy of speaker verification, a proper dataset of reverberant audio is needed. An alternative that fills a qualitative gap between unsatisfying simulation (despite the improvement of realism reported in [Ravanelli et al., 2016]) and costly and demanding real speaker recording, is retransmission. To our advantage, we can also use the fact that a *known* dataset can be retransmitted so that the performances are readily comparable with known benchmarks. Hence, this was our method to obtain a new dataset.

The retransmission took place in a room with floor plan displayed in Figure 3.4. The configuration fits several purposes: the loudspeaker–microphone distance rises steadily for microphones 1–6 to study deterioration as a function of distance, microphones 7–12 form a large microphone array mainly focused to explore beamforming (beyond the scope of this paper but studied in [Mošner et al., 2018]).

For this work, a subset of NIST SRE 2010 data was retransmitted. The dataset consists of 459 female recordings[2] with nominal durations of three and eight minutes. The total number of female speakers is 150. The files were played in sequence and recorded simultaneously by a multi-channel acquisition card that ensured sample precision synchronization.

We denote the retransmitted data as benchmark *BUT-RET*, where *BUT-RET-orig* represents the original (not retransmitted) data and *BUT-RET-merge* stands for data from 14 benchmarks: trials for all 14 microphones were pooled to a single benchmark.

3.3 Data Augmentation Design

For training of the discriminative techniques, we needed a fairly large amount of clean utterances from which we formed a parallel dataset of clean and augmented (noisy, reverberated or both) utterances. We chose Fisher English database Parts 1 and 2 as they span a large number of speakers (11971) and the audio is relatively clean and without reverberation. These databases combined contain over 20,000 telephone conversational sides or approximately 1800 hours of audio.

3.3.1 Noise

We prepared a noise dataset that consists of three sources of different types of noise:

[2]NIST SRE 2010 female recordings are known to be more challenging for SV systems; therefore, they were our choice for retransmission.

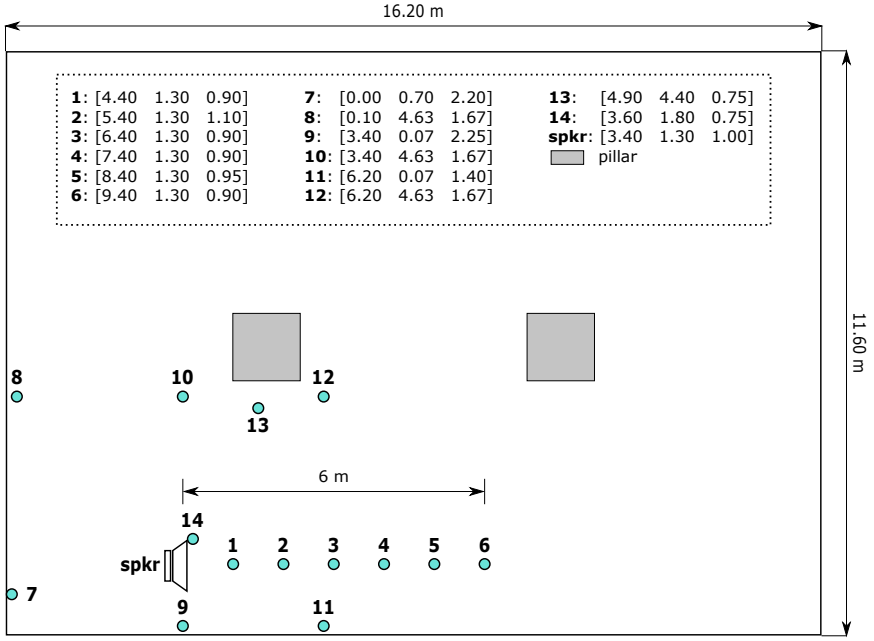


Figure 3.4: Floor plan of the room in which the retransmission took place. Coordinates are in meters and lower left corner is the origin. (Source: [Mošner et al., 2018])

- 272 samples (4 minutes long) taken from the Freesound library[3] (real fan, heating/ventilation/air conditioning systems, street, city, shop, crowd, library, office and workshop).
- 7 samples (4 minutes long) of artificially generated noises: various spectral modifications of white noise + 50 and 100 Hz hum.
- 25 samples (4 minutes long) of babbling noises by merging speech from 100 random speakers from Fisher database selected using speech activity detector.

Noises were divided into three disjoint groups for training (223 files), development (40 files) and test (41 files).

3.3.2 Reverberation

We prepared two sets with room impulse responses (RIRs). The first set consists of real room impulse responses from several databases: AIR[4], C4DM[5] [Stewart and Sandler, 2010], MARDY[6], OPENAIR[7], RVB 2014[8], and RWCP[9]. Together, they form a set with all types of rooms (small rooms, big rooms, lecture room, restrooms, halls, stairs, etc.). All room models have more than one impulse response per room (different RIR was used for source

[3] http://www.freesound.org
[4] http://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/
[5] http://isophonics.net/content/room-impulse-response-data-set
[6] http://www.commsp.ee.ic.ac.uk/~sap/resources/mardy-multichannel-acoustic-reverberation-database-at-york-database/
[7] http://www.openairlib.net/auralizationdb
[8] http://reverb2014.dereverberation.com/index.html
[9] http://www.openslr.org/13/

of the signal and source of the noise to simulate different locations of their sources). Rooms were split into two disjoint sets, with 396 rooms for training, 40 rooms for test.

The second set consists of artificially generated RIRs using "Room Impulse Response Generator" tool from E. Habets [Habets, 2006]. The tool can model the size of room (3 dimensions), reflectivity of each wall, type of microphone, position of source and microphone, orientation of microphone towards the audio source, and number of bounces (reflections) of the signal. We generated a pair of RIRs for each room model (one used for source of the sound, one for source of the noise). Again, we generated two disjoint sets, with 1594 RIRs for training and 250 RIRs for test.

For some techniques presented in this work, it is necessary to have paired data, in which each pair is composed of a clean and an augmented recording. If reverberation is applied, the recording may shift in time depending on the distance of the microphone and the signal source used to calculate the RIR. In order to preserve the time alignment between the original and the augmented recording, it is necessary to pre-process all RIRs: shift them to the left by the delay of the direct path, as shown in Figure 2.2b.

3.3.3 Composition of the Training Set

To mix the reverberation, noise and signal at a given SNR, we followed the procedure showed in Figure 3.6. The pipeline begins with two branches, where speech and noise are reverberated separately. Different RIRs from the same room are used for signal and noise, to simulate different positions of sources.

In the following step, we set a ratio of noise and signal energies to obtain the required SNR. Energies of the signal and noise are computed from A-weighted frames given by original signal's voice activity detection (VAD). It means the computed SNR is really present in speech frames which are important for SV (frames without voice activity are removed during processing). A-weighting (Figure 3.5) helps adjusting the ratio so that the human ear can also correctly estimate the SNR. That is, the SNR corresponds primarily to the energy of noise at human-audible frequencies.

The useful signal and noise are then summed at desired SNR, followed by optional filtering by telephone channel (see page 9 in [ITU, 1994]) or codec application. In case we want to add only noise or reverberation, only the appropriate part of the algorithm is used.

3.3.4 Kaldi Data Augmentation Recipe

In specific experiments, we also use the Kaldi repository and its recipes[10]. These recipes serve as a certain standard for the presented system and data preparation. In these cases, we use the augmentation defined by Kaldi. The method of adding noise and reverberation is similar to the one described above. The difference lies in mixing clean and augmented data. A different database of noise and RIRs is also used.

In the original Kaldi recipe, the training data were augmented with reverberation, noise, music, and babble noise and combined with the original clean data. The package of all noises and room impulse responses can be downloaded from OpenSLR[11] [Ko et al., 2017], and includes MUSAN noise corpus with 843 noises [Snyder et al., 2015].

For data *augmentation with reverberation*, the total amount of RIRs is divided into two equally distributed lists for medium and small rooms.

[10] https://github.com/kaldi-asr/kaldi
[11] http://www.openslr.org/resources/28/rirs_noises.zip

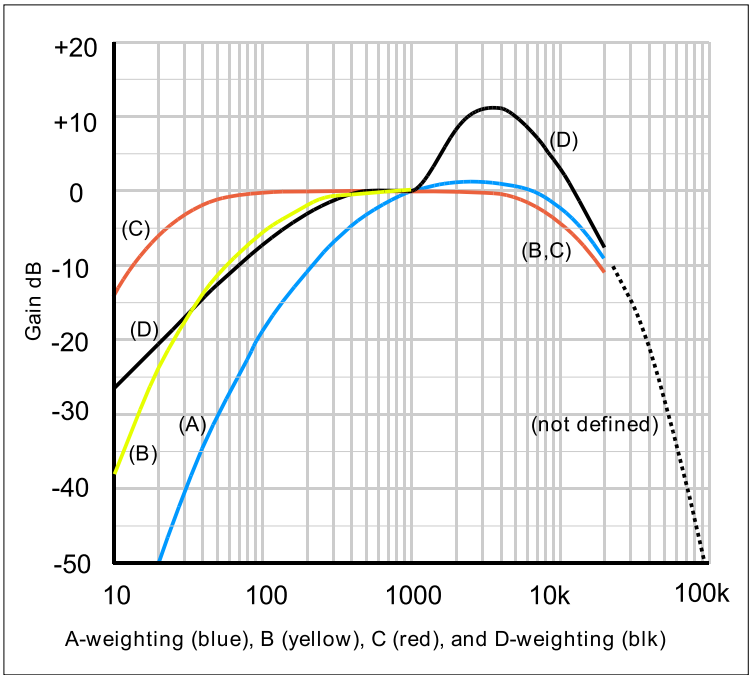


Figure 3.5: Several weighting curves used for measurement of sound level for the loudness perceived by human hearing. In this work, we use A-weighting. (Source: www.lindos.co.uk)

For *augmentation with noise*, we created three replicas of the original data. The first replica was modified by adding MUSAN[12] noises at SNR levels in the range of 0–15 dB. In this case, the noise was added as a *foreground* noise (that means several non-overlapping noises can be added to the input audio). The second replica was mixed with music at SNRs ranging from 5 to 15 dB as *background* noise (one noise per audio with the given SNR). The last noisy replica of training data was created by mixing in the babble noise. SNR levels were at 13–20 dB and we used 3–7 noises per audio. The augmented data were pooled and a random subset of 200k audios was selected and combined with clean data. The process of data augmentation is also described in [Snyder et al., 2018].

Apart from the original recipe, as described in the previous paragraph, we also added our own processing: real room impulse responses and stationary noises described above. The original RIR list was extended by our list of real RIRs and we kept one reverberated replica. Our stationary noises were used to create another replica of data with SNR levels in range 0–20 dB. We combined all replicas and selected a subset of 200k files. As a result, after performing all augmentations, we obtain 5 replicas for each original utterance. The whole process of creating the augmented training set is depicted in Figure 3.7.

3.3.5 Selected Benchmark Scenarios

Below, we list short descriptions and names of the benchmarks used to evaluate our systems:

[12]Musan was introduced in [Snyder et al., 2015] for training of models for voice activity detection (VAD) and for music/speech discrimination. The corpus consists of approximately 109 hours of audio consisting of 60.5 hours of speech, 42.5 hours of music, and 6 hours of noise.

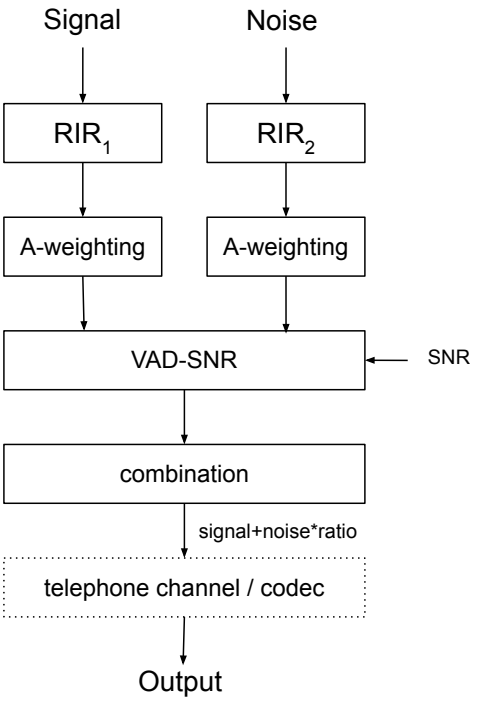


Figure 3.6: The process of data augmentation for autoencoder training, generating additional data for PLDA training, or system testing. The last step—filtering with the telephone channel—is used only when creating the denoising autoencoder training data.

- *tel-tel*: SRE 2010 extended telephone benchmark involving normal vocal effort conversational telephone speech in enrollment and test (known as "condition 5").
- *int-int*: SRE 2010 extended interview benchmark involving interview speech from different microphones in enrollment and test (known as "condition 2").
- *int-mic*: SRE 2010 extended interview-microphone benchmark involving interview enrollment speech and normal vocal effort conversational telephone test speech recorded over a room microphone channel (known as "condition 4").
- *sre16-all*: SRE 2016 non-English telephone benchmark, consist of Tagalog and Cantonese language
- *sre16-tgl*: The subset of *sre16-all* with Tagalog speakers only
- *sre16-yue*: The subset of *sre16-all* with Cantonese speakers only
- *prism,noi*: Clean and artificially noised waveforms from both interview and telephone conversations recorded over lavalier microphones. Noise was added at different SNR levels and recordings are tested against each other.
- *prism,rev*: Clean and artificially reverberated waveforms from both interview and telephone conversations recorded over lavalier microphones. Reverberation was added with different reverberation times (RTs) and recordings are tested against each other.

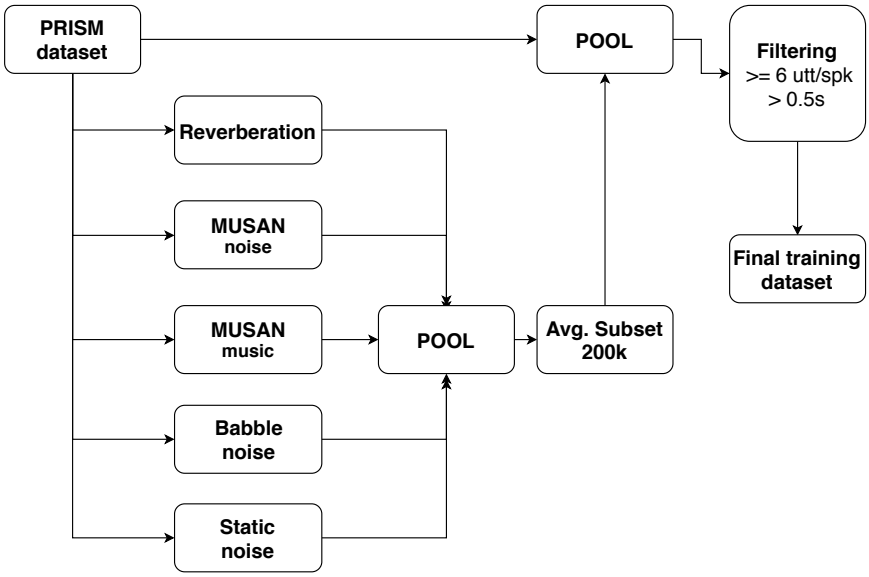


Figure 3.7: Preparation of the x-vector extractor training dataset.

- *prism,chn*: English telephone conversation with normal vocal effort recorded over different microphones from both SRE 2008 and 2010. Recordings are tested against each other.
- *prism,lan*: The language-language benchmark defined in the PRISM set, which comprises data from previous NIST evaluations in five different languges.
- *prism,chin*: Chinese subset of *prism,lan*.
- *sitw-core-core*: Interview conversation with an adequate vocal effort to the real environment during audio recording. Enrollment and test segments contain between 6 and 180 seconds of continuous speech from a single speaker.
- *BUT-RET-orig*: Original (non-retransmitted) subset of NIST SRE 2010 data.
- *BUT-RET-merge*: Retransmitted version subset of NIST SRE 2010 data, with trials from all 14 microphones pooled.

In the research community focusing on the SV field, we usually refer to the benchmarks as conditions, but to avoid misunderstanding and confusion with the acoustic conditions, we refer to them as "benchmarks".

Chapter 4

Embedding-Based Speaker Verification

In the field of speaker verification, it has always been necessary to deal with the problem of different utterance lengths. Finding a good representation of a recording independent of its length is one of the critical features of SV systems. One of the first solutions addressing this problem were systems based on GMM and likelihood comparison [Reynolds et al., 2000]. Compared to today's systems, these systems were in the order of magnitude worse [Matějka et al., 2020]. Also, the subsequent usage of the systems with different types of classifiers was problematic, because of absence of fixed-length vector representation.

Demand for an utterance representation that is easier to model results in a fixed-length vector (usually referred to as *embedding*) representation. A fixed-length vector representation was found useful also in other speech processing fields, such as language recognition, automatic speech recognition and a wide range of natural language processing (NLP) tasks.

As it was mentioned in Section 1.1, the embedding-based system (Figure 1.1) became a dominant approach in SV. In this chapter, we present two strong concepts of embeddings: i-vectors and x-vectors. Both differ in the primary paradigm, since *i-vectors* are based on a generative model and *x-vectors* on a discriminative one. A detailed overview of their history and architecture is given in the relevant sections.

4.1 Generatively Trained Embedding — i-vector

Before presenting the i-vectors, it is necessary to describe the essential prerequisites of Gaussian Mixture Models and their purpose in this concept of generative embedding extraction.

4.1.1 Gaussian Mixture Modeling of Acoustic Features

Gaussian Mixture Models (GMMs) represent a popular approach to acoustic feature space modeling. Due to the nature of the speech signal and its properties, it is necessary to consider their distribution multimodal. This assumption leads us to the use of a mixture of Gaussian distributions. The mixture is given by the weighted sum of C Gaussian distribution components, and we assume that each sample was generated by one component.

In the past, GMMs have proven to be a very effective method of modeling in all areas of general speech processing, such as speech transcription [Young et al., 2006], speaker recog-

nition [Reynolds et al., 2000], and language recognition [Torres-Carrasquillo et al., 2002]. The GMMs have also found their application in other areas of biometrics, such as facial recognition [Bredin et al., 2006] and signature recognition [Tolosana et al., 2015].

To formally describe GMM, let us define a speech segment as a set of F-dimensional features $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N]$. The probability density function of feature vector $\mathbf{o}_i$ given the GMM [Bishop, 2006]:

$$\mathcal{G}(\mathbf{o}_i; \boldsymbol{\theta}) = \sum_{c=1}^{C} w^{(c)} \mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}), \tag{4.1}$$

where $\mathcal{G}(\mathbf{o}_i; \boldsymbol{\theta})$ is likelihood of feature vector $\mathbf{o}_i$, $\boldsymbol{\theta}$ is a vector of all model parameters, $\boldsymbol{\mu}^{(c)}$ represents mean vector of a component c, $\boldsymbol{\Sigma}^{(c)}$ is covariance matrix of component c. Component weights $w^{(c)}$ must meet the conditions $w^{(c)} \geq 0$ and $\sum_{c=1}^{C} w^{(c)} = 1$.

For F-dimensional feature vector $\mathbf{o}_i$, single Gaussian model $\mathcal{N}(\mathbf{o}_i, \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)})$ is given by:

$$\mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}) = \frac{1}{(2\pi)^{F/2} |\boldsymbol{\Sigma}^{(c)}|^{1/2}} e^{-\frac{1}{2}(\mathbf{o}_i - \boldsymbol{\mu}^{(c)})^T \boldsymbol{\Sigma}^{(c)^{-1}} (\mathbf{o}_i - \boldsymbol{\mu}^{(c)})}. \tag{4.2}$$

The whole GMM is represented by parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = \left\langle \boldsymbol{w}^{(c)}, \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)} \right\rangle \text{ with } c = 1 \ldots C \tag{4.3}$$

or more conveniently as super vectors and the matrix of stacked parameters:

$$\boldsymbol{\theta} = \left\langle \boldsymbol{w}, \boldsymbol{\mu}, \boldsymbol{\Sigma} \right\rangle, \tag{4.4}$$

where $\boldsymbol{w}$ is the vector of component weights for the corresponding components, $\boldsymbol{\mu}$ is a super-vector of concatenated component mean vectors $\boldsymbol{\mu}^{(c)}$:

$$\boldsymbol{w} = \begin{bmatrix} w^{(1)} \\ \vdots \\ w^{(C)} \end{bmatrix}, \tag{4.5}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^{(1)} \\ \vdots \\ \boldsymbol{\mu}^{(C)} \end{bmatrix}, \tag{4.6}$$

and $\boldsymbol{\Sigma}$ is generally a block-matrix of component covariance matrices:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Sigma}^{(C)}. \end{bmatrix} \tag{4.7}$$

It should be mentioned that in practice, a simplified model with a diagonal covariance matrix is generally used. Furthermore, for computational and numerical reasons, the logarithm of likelihood is used, and (4.1) can be written as:

$$\begin{aligned} \log \mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}) = &-\frac{1}{2} F \log(2\pi) - \frac{1}{2} \log |\boldsymbol{\Sigma}^{(c)}| - \frac{1}{2} \boldsymbol{\mu}^{(c)^T} \boldsymbol{\Sigma}^{(c)^{-1}} \boldsymbol{\mu}^{(c)} \\ &- \frac{1}{2} \mathbf{o}_i{}^T \boldsymbol{\Sigma}^{(c)^{-1}} \mathbf{o}_i - \mathbf{o}_i{}^T \boldsymbol{\Sigma}^{(c)^{-1}} \boldsymbol{\mu}^{(c)} \end{aligned} \tag{4.8}$$

Data Alignment and Suffcient Statistic

The assignment of the feature vector $\boldsymbol{o}_i$ to component c is unknown and forms a hidden variable. However, for each feature vector $\boldsymbol{o}_i$ we can determine the degree of attribution of this vector $\boldsymbol{o}_i$ to component c from the GMM model, based on the posterior distribution. The posterior probability of the feature vector $\boldsymbol{o}_i$ (also referred to as occupation probability, shortly denoted as γ_i) can be computed using Bayes rule [Bishop, 2006]:

$$\gamma_i^{(c)} = \frac{w^{(c)}\mathcal{N}(\boldsymbol{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)})}{\sum_{j=1}^{C} w^{(j)}\mathcal{N}(\boldsymbol{o}_i; \boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)})}. \tag{4.9}$$

The configuration of the posterior probabilities for each feature vector $\boldsymbol{o}_i$ is referred to as the alignment of the data to the mixture components. In practice, it is advantageous to define Sufficient statistics:

$$N^{(c)} = \sum_{i=1}^{N} \gamma_i^{(c)} \tag{4.10}$$

$$\mathbf{f}^{(c)} = \sum_{i=1}^{N} \gamma_i^{(c)} \mathbf{o}_i \tag{4.11}$$

$$\mathbf{S}^{(c)} = \sum_{i=1}^{N} \gamma_i^{(c)} \mathbf{o}_i \mathbf{o}_i^T \tag{4.12}$$

We refer to these as the zero-, first- and second-order statistics for a sequence of F-dimensional features $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N]$, where N is a number of feature vectors. For simplicity, we define normalized statistics as:

$$\tilde{\mathbf{f}}^{(c)} = \mathbf{f}^{(c)} - N^{(c)}\boldsymbol{\mu}^{(c)} \tag{4.13}$$

$$\tilde{\mathbf{S}}^{(c)} = \mathbf{S}^{(c)} - \mathbf{f}^{(c)}\boldsymbol{\mu}^{(c)^T} - \boldsymbol{\mu}^{(c)}\mathbf{f}^{(c)^T} + N^{(c)}\boldsymbol{\mu}^{(c)}\boldsymbol{\mu}^{(c)^T} \tag{4.14}$$

For further simplification, the statistics can be expanded into the form of super-vector and a super-matrix as:

$$\mathbf{N} = \begin{bmatrix} N^{(1)}\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & N^{(2)}\mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & N^{(C)}\mathbf{I} \end{bmatrix}, \tag{4.15}$$

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \vdots \\ \mathbf{f}^{(C)} \end{bmatrix}, \tag{4.16}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{S}^{(C)} \end{bmatrix}. \tag{4.17}$$

Stacked centered statistic super vector $\tilde{\mathbf{f}}$ and super-matrix $\tilde{\mathbf{S}}$, can be produced in the same fashion as $\mathbf{f}$ and $\mathbf{S}$ but from centered variants $\tilde{\mathbf{f}}^{(c)}$ and $\tilde{\mathbf{S}}^{(c)}$.

Likelihood Function

Let us focus on the likelihood function and its different variants, which is used later in this text. Assume we have a set of statistically independent input data (feature vectors/frames) $\mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N,]$. The likelihood of the data is represented as:

$$p(\mathbf{O}|\theta) = \prod_{i=1}^{N} \mathcal{G}(\mathbf{o}_i; \boldsymbol{\theta}) \tag{4.18}$$

We usually do not work with the likelihood of this form but instead use the logarithm:

$$\log p(\mathbf{O}|\theta) = \sum_{i=1}^{N} \log \sum_{c=1}^{C} w^{(c)} \mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}) \tag{4.19}$$

For further usage, we modify the likelihood function into a slightly more complicated form. This modification allows us to get the logarithm out of the sum in front of the mixture model components. For any distribution $q(c)$ we can re-write the likelihood function $\log p(\mathbf{O}|\theta)$ as:

$$\begin{aligned}
\log p(\mathbf{O}|\theta) &= \sum_{i=1}^{N} \log p(\mathbf{o}_i|\theta) = \sum_{i=1}^{N} \underbrace{\sum_{c=1}^{C} q_i(c)}_{1} \log \frac{p(\mathbf{o}_i, c|\theta)}{p(c|\mathbf{o}_i, \theta)} \frac{q_i(c)}{q_i(c)} \\
&= \sum_{i=1}^{N} \sum_{c=1}^{C} q_i(c) \log \left(\mathcal{N}\big(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\big) \right) \\
&- \sum_{i=1}^{N} \sum_{c=1}^{C} q_i(c) \log \frac{q_i(c)}{w^{(c)}} \\
&+ \sum_{i=1}^{N} \underbrace{\sum_{c=1}^{C} q_i(c) \log \frac{q_i(c)}{\gamma_i^{(c)}}}_{\mathrm{D_{KL}}(q_i(c)||\gamma_i)} .
\end{aligned} \tag{4.20}$$

$\mathrm{D_{KL}}(q_i(c)||\gamma_i^{(c)})$ is Kullback-Leibler (KL) divergence, between $q_i(c)$ and the distribution $\gamma_i^{(c)} = p(c|\mathbf{o}_i, \boldsymbol{\theta})$. If we set $q(c)$ to the true posterior probability γ_i, $\mathrm{D_{KL}}(q_i(c)||\gamma_i^{(c)})$ vanishes and the likelihood function in (4.20) can be reformulated as:

$$\log p(\mathbf{O}|\theta) = \sum_{i=1}^{N} \sum_{c=1}^{C} \gamma_i^{(c)} \log \mathcal{N}(\mathbf{o}_i; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}) - \sum_{i=1}^{N} \sum_{c=1}^{C} \gamma_i^{(c)} \log \frac{\gamma_i^{(c)}}{w^{(c)}} \tag{4.21}$$

Using sufficient statistic in (4.15), (4.16), and (4.17), we can then rewrite the likelihood function as follows:

$$\begin{aligned}\log p(\mathbf{O}|\theta) = &\sum_{c=1}^{C} N^{(c)} \frac{1}{(2\pi)^{F/2}|\boldsymbol{\Sigma}^{(c)}|^{1/2}} \\ &- \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}) \\ &+ \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{f} - \frac{1}{2}\boldsymbol{\mu}^T\mathbf{N}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &- \sum_{i=1}^{N}\sum_{c=1}^{C} \gamma_i^{(c)} \log \frac{\gamma_i^{(c)}}{w^{(c)}}\end{aligned} \tag{4.22}$$

Maximum Likelihood Estimation of Parameters

When searching for the parameters $\boldsymbol{\theta}$ of GMM model, we maximize the expected likelihood $p(\mathbf{O}, \boldsymbol{\theta})$ over a given training set $\mathbf{O}$:

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} p(\mathbf{O}, \theta) \tag{4.23}$$

Usually, parameters can be found by solving $\frac{\mathrm{d}\, p(\mathbf{O},\theta)}{\mathrm{d}\,\theta} = 0$.

For a unimodal Gaussian model, the ML estimation of the parameters can be expressed directly as a closed-form solution. Unfortunately, there is no closed-form solution for GMM, because the assigment of training samples to the components of the model is not known. Probably the most used way to find the parameters of the GMM model is the Expectation-Maximization algorithm [Dempster et al., 1977] — an iterative procedure performing two steps in each iteration:

- **E**-step: We obtain posterior probabilities $\gamma_{\boldsymbol{\theta}_0}^{(c)}$ by fixing the alignment of the data $\mathbf{O}$ using current model $\boldsymbol{\theta}_0$ and collect statistic $\{N_{\boldsymbol{\theta}_0}^{(c)}, \mathbf{f}_{\boldsymbol{\theta}_0}^{(c)}, \mathbf{S}_{\boldsymbol{\theta}_0}^{(c)}\}$.
- **M**-step: The new ML estimate of the parameters is computed to maximize Q_{GMM}:

$$\boldsymbol{\theta}_{ML} = \arg\max_{\theta} Q_{\mathrm{GMM}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0) \tag{4.24}$$

For **E**-step, we need to construct auxiliary function Q_{GMM}. The true alignment is not available and is provided via a different model $\boldsymbol{\theta}_0$, then $q(c) = p(c|\mathbf{o}, \boldsymbol{\theta}_0)$. KL-divergence in (4.20) requires true alignment; therefore, the computation is impossible. If we assume Q_{GMM} only as an approximate of the true likelihood, we can omit KL-divergence. Then, Q_{GMM} can be constructed as:

$$\begin{aligned}Q_{\mathrm{GMM}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0) = &\sum_{c=1}^{C} N_{\boldsymbol{\theta}_0}^{(c)} \frac{1}{(2\pi)^{F/2}|\boldsymbol{\Sigma}^{(c)}|^{1/2}} \\ &- \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}_{\boldsymbol{\theta}_0}) \\ &+ \boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\mathbf{f}_{\boldsymbol{\theta}_0} - \frac{1}{2}\boldsymbol{\mu}^T\mathbf{N}_{\boldsymbol{\theta}_0}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ &- \sum_{i=1}^{N}\sum_{c=1}^{C} \gamma_{i_{\boldsymbol{\theta}_0}}^{(c)} \log \frac{\gamma_{i_{\boldsymbol{\theta}_0}}^{(c)}}{w^{(c)}}\end{aligned} \tag{4.25}$$

This approximation is a lower-bound of the correct likelihood function since the omitted KL-divergence is always non-negative. For **M**-step (4.24), a closed-form solution exists and update formulas of model parameters are given as:

$$\begin{aligned} w^{(c)} &= \frac{N^{(c)}}{N}, \\ \boldsymbol{\mu}_{ML}^{(c)} &= \frac{1}{N^{(c)}}\mathbf{f}^{(c)}, \\ \boldsymbol{\Sigma}_{ML}^{(c)} &= \frac{1}{N^{(c)}}\mathbf{S}^{(c)} - \boldsymbol{\mu}_{ML}^{(c)}\boldsymbol{\mu}_{ML}^{(c)\,T}. \end{aligned} \tag{4.26}$$

Repeating the **E**- and **M**-steps guarantees not to decrease the likelihood. The algorithm stops when the likelihood increase between two consecutive iterations is under a selected threshold.

Universal Background Model

In the SV, GMM is mainly used to train the Universal Background Model (UBM). The UBM serves as a representation of a space of acoustic features independent of the speaker. For a proper training of a UBM, we need large amounts of data [Burget et al., 2007], ideally from the target domain. If the domain is not known, it is desirable to provide the highest possible variability in the training data: speakers of different genders, different acoustic environments, different types of microphones. The amount of available training data then usually affects the number of model components. Improperly selected number of components or small amount of training data can lead to overtraining of the model. Initialization forms an integral part of GMM training. There are several ways to initialize the training and an excellent type of initialization is to use a K-means algorithm [Bishop, 2006], then the $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ parameters can be set on the basis of the data assigned to each cluster. Progressive Gaussian splitting represents another approach (used in this work and including diagonal covariance matrices). The model starts as a single Gaussian model. After a selected number of training iterations, each component is duplicated. The two components are shifted in the highest variability directions. This algorithm is described in more detail in [Glembek, 2012].

MAP Adaptation

Another approach of how to estimate parameters of the GMM model is the maximum a-posteriori criterium (MAP). In the ML approach, we consider model parameters as unknown but fixed. In MAP, model parameters are random variables. Generally, the parameters are computed as:

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\theta} p(\boldsymbol{\theta}|\hat{\mathbf{O}}), \tag{4.27}$$

where $p(\boldsymbol{\theta}|\hat{\mathbf{O}})$ is the posterior probability for the parameter $\boldsymbol{\theta}$, given by input data $\hat{\mathbf{O}}$:

$$p(\boldsymbol{\theta}|\hat{\mathbf{O}}) = \frac{p(\hat{\mathbf{O}}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\hat{\mathbf{O}})}. \tag{4.28}$$

$p(\hat{\mathbf{O}})$ does not depend on $\boldsymbol{\theta}$, therefore it can be omitted:

$$p(\boldsymbol{\theta}|\hat{\mathbf{O}}) \propto p(\hat{\mathbf{O}}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{4.29}$$

Equation (4.27) can be transformed into:

$$\boldsymbol{\theta}_{MAP} = \arg\max_{\theta} p(\hat{\mathbf{O}}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{4.30}$$

We can notice that ML is a particular case of MAP, where $p(\boldsymbol{\theta})$ is flat. MAP approach is very advantageous in moments when we have a small amount of training data that would not lead to a robust estimate of model parameters by the ML method. However, the prerequisite is the presence of a good a-priori distribution of parameters $p(\boldsymbol{\theta})$.

In SV, the UBM model, previously trained on a larger amount of data, usually serves as a source of prior information. In the MAP approach, usually, only part of the parameters are trained (mean $\boldsymbol{\mu}$); the rest of the parameters remain shared with the UBM. We often call this process MAP adaptation. A smaller amount of data would not lead to a robust estimates of covariance matrices $\boldsymbol{\Sigma}$.

MAP mean adaptation can be expressed as:

$$\boldsymbol{\mu}^{(c)}_{MAP} = \beta^{(c)}\boldsymbol{\mu}^{(c)}_{ML} + (1-\beta^{(c)})\boldsymbol{\mu}^{(c)}_{UBM}, \tag{4.31}$$

with

$$\beta^{(c)} = \frac{N^{(c)}}{N^{(c)}+\tau}, \tag{4.32}$$

where $\boldsymbol{\mu}^{(c)}_{UBM}$ is original UBM mean, $\boldsymbol{\mu}^{(c)}_{ML}$ is ML estimation of mean by given data and $N^{(c)}$ are zero-order statistics. The τ constant is usually referred to as a relevance factor [Reynolds et al., 2000]. It controls the trade-off between the original UBM model and ML estimation. With decreasing value of τ, $\beta^{(c)}$ increases, which means that more emphasis is put on $\boldsymbol{\mu}^{(c)}_{ML}$ than $\boldsymbol{\mu}^{(c)}_{UBM}$. With the increasing amount of adaptation data, $\beta^{(c)}$ increases, leading to more weight being put on $\boldsymbol{\mu}^{(c)}_{ML}$.

4.1.2 i-vectors

Super-vectors are the first concept of utilizing a generative model and representing variable-length utterance by a fixed-length vector. The term super-vector usually refers to a vector of concatenated mean values of GMM model components. The super-vector then represents an arbitrarily long recording with a fixed-length vector [Kinnunen and Li, 2010]. The disadvantage of the super-vector lies in its relatively large dimensionality. Assuming the C-components of the GMM model modeling the F-dimensional features, we obtain a CF-dimensional super vector for each utterance, see (4.6). GMM uses unsupervised training, and speaker labels are not necessary during the training. Regarding this fact, the super-vector does not represent only the desired information about the speaker. It also carries information about the entire domain (channel, language, etc.) in the recording. Considering that MAP adaptation is a method of extracting a super-vector, another problem arises in the case of short recordings. The short utterance may not cover all the components of the GMM model of acoustic space. Some sub-vectors $\boldsymbol{\mu}^{(c)}_{spk}$ from the speaker super vector $\boldsymbol{\mu}_{spk}$ could remain unchanged in comparison to $\boldsymbol{\mu}^{(c)}_{UBM}$. So, in case of the super-vector approach, we have some high-dimensional space, in which only subspace brings us some interesting variability. This fact gave rise to a considerable amount of work dealing with subspace modeling as eigen-voices in SV [Thyes et al., 2000, Kenny et al., 2003], eigen-channel adaptation [Brümmer, 2004], Joint Factor Analysis [Kenny, 2005], Within-Class Covariance

Normalization [Hatch et al., 2006], i-vectors [Dehak et al., 2010] or multinomial sub-space models in prosodic SV [Kockmann, 2012].

The i-vector approach has become a prevalent technique in speaker recognition. Over time, it has found a place in other fields of speech processing as well: ASR [Saon et al., 2013], LID [Martínez et al., 2011], or age estimation [Silnova et al., 2015].

For many years, the i-vectors have been the state-of-the-art technique in the speaker verification field. The beginning of i-vectors is dated to the John Hopkins University summer workshop on Robust Speaker Recognition in 2008 [Burget et al., 2008]. At the time, Joint Factor Analysis (JFA, for more details see [Kenny, 2005, Kenny et al., 2007]) was the state-of-the-art technology, and it was one of the topics to be investigated at the workshop. One of the research directions was to use the JFA as a feature extraction. Experiments were conducted with the speaker factors from JFA being used as low-dimensional features for SVM classifier. A similar experiment with channel factors of JFA also demonstrated acceptable performance, suggesting that even channel factors contained a fair amount of information about the speaker.

On the basis of this finding, [Dehak et al., 2010] proposed a reduction of JFA to a model with a single subspace that would represent a total variability information. Later, this system was simplified and optimized [Glembek et al., 2011b, Cumani and Laface, 2013, Cumani and Laface, 2014].

Theoretical Background

The main idea behind the i-vector model lies in a transformation of a high-dimensional super-vector $\boldsymbol{\mu}$ into a low-dimensional subspace while retaining significant variability of the original space. Speaker- and channel-dependent super vector $\boldsymbol{\mu}$ can be modeled as:

$$\boldsymbol{\mu} = \mathbf{m} + \mathbf{T}\boldsymbol{\phi}, \tag{4.33}$$

where $\mathbf{m}$ represents mean super-vector of UBM GMM, $\mathbf{T}$ is a low-rank matrix representing M bases spanning important variability in mean super vector space, and $\boldsymbol{\phi}$ is a random M-dimensional vector with a standard normal prior distribution:

$$p(\boldsymbol{\phi}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.34}$$

Prior distribution of $\boldsymbol{\mu}$ is given as

$$p(\boldsymbol{\mu}) = \mathcal{N}(\mathbf{m}, \mathbf{T}\mathbf{T}^T). \tag{4.35}$$

Likelihood Definition

In case of parameter estimation, we begin with a general log-likelihood function as defined in Section 4.1.1. For simplification, we consider a fixed alignment [Kenny, 2005] and sufficient statistics collected by UBM as defined in Section 4.1.1. With this assumption, the log-likelihood acts as the lower-bound to the real log-likelihood.

For all observations, $\boldsymbol{O}_i = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N]$ from utterance i, the log-likelihood can be decomposed as:

$$\begin{aligned}\log p(\mathbf{O}_i|\boldsymbol{\phi}_i, \mathbf{T}) &= H(\boldsymbol{\phi}_i) + G \\ H(\boldsymbol{\phi}_i) &= \boldsymbol{\mu}_i \boldsymbol{\Sigma}^{-1} \mathbf{f}_i - \frac{1}{2} \boldsymbol{\mu}_i^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_i \\ G &= \sum_{c=1}^{C} N^{(c)} \frac{1}{(2\pi)^{F/2} |\boldsymbol{\Sigma}^{(c)}|^{1/2}} - \frac{1}{2} \mathrm{tr}(\boldsymbol{\Sigma}^{-1} \mathbf{S}_i) - \sum_{i=1}^{N} \sum_{c=1}^{C} \gamma_i^{(c)} \log \frac{\gamma_i^{(c)}}{w^{(c)}},\end{aligned} \tag{4.36}$$

where $\mathbf{N}_i$, $\mathbf{f}_i$, $\mathbf{S}_i$ are stacked zero-, first-, and second-order statistic collected by the UBM. $\boldsymbol{\Sigma}$ is block diagonal covariance matrix, composed as in (4.7). The term G is independent of $\boldsymbol{\mu}_i$, and it is set to constant. By substituting $\boldsymbol{\mu}_i$ with (4.33), $H(\boldsymbol{\phi}_i)$ can be re-written as:

$$\begin{aligned}H(\boldsymbol{\phi}_i) = &\underbrace{\mathbf{m}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{f}}_i}_{0} + \boldsymbol{\phi}_i^T \mathbf{T}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{f}}_i \\ &- \frac{1}{2} \underbrace{\mathbf{m}^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \mathbf{m}}_{0} - \frac{1}{2} \boldsymbol{\phi}_i^T \mathbf{T}^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \mathbf{T} \boldsymbol{\phi}_i - \underbrace{\mathbf{m}^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \mathbf{T} \boldsymbol{\phi}_i}_{0}.\end{aligned} \tag{4.37}$$

With first-order statistics centered around UBM means:

$$\begin{aligned}\tilde{\mathbf{f}}_i^{(c)} &= \mathbf{f}_i^{(c)} - N_i^{(c)} \mathbf{m}^{(c)} \\ \mathbf{m}^{(c)} &\leftarrow \mathbf{0}.\end{aligned} \tag{4.38}$$

This allows us to effectively treat the UBM means $\mathbf{m}$ as a vector of zeros, leading us to another simplification:

$$H(\boldsymbol{\phi}_i) = \boldsymbol{\phi}_i^T \mathbf{T}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{f}}_i - \frac{1}{2} \boldsymbol{\phi}_i^T \mathbf{T}^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \mathbf{T} \boldsymbol{\phi}_i. \tag{4.39}$$

The joint log-likelihood of hidden variable $\boldsymbol{\phi}_i$ and oservation $\mathbf{O}_i$ is given as:

$$\begin{aligned}p(\mathbf{O}_i, \boldsymbol{\phi}_i|\mathbf{T}) &= \log p(\mathbf{O}_i|\boldsymbol{\phi}_i, \mathbf{T}) + \log p(\boldsymbol{\phi}_i) \\ \log p(\boldsymbol{\phi}_i) &= \log \mathcal{N}(\boldsymbol{\phi}_i; \mathbf{0}, \mathbf{I}) = -\frac{1}{2} \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i + K \\ p(\mathbf{O}_i, \boldsymbol{\phi}_i|\mathbf{T}) &= \boldsymbol{\phi}_i^T \mathbf{T}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{f}}_i - \frac{1}{2} \boldsymbol{\phi}_i^T \mathbf{T}^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \mathbf{T} \boldsymbol{\phi}_i - \frac{1}{2} \boldsymbol{\phi}_i^T \boldsymbol{\phi}_i + K,\end{aligned} \tag{4.40}$$

where K is a constant, independent of $\boldsymbol{\mu}$ and $\mathbf{T}$. Omitting K, the posterior distribution of the hidden variable $\boldsymbol{\phi}_i$, given observation $\mathbf{O}_i$ for utterance i, is given as:

$$p(\boldsymbol{\phi}_i|\mathbf{O}_i) \propto \log p(\mathbf{O}_i, \boldsymbol{\phi}_i) \propto \boldsymbol{\phi}_i^T \mathbf{T}^T \boldsymbol{\Sigma}^{-1} \tilde{\mathbf{f}}_i - \frac{1}{2} \boldsymbol{\phi}_i^T (\mathbf{T}^T \mathbf{N}_i \boldsymbol{\Sigma}^{-1} \mathbf{T} - \mathbf{I}) \boldsymbol{\phi}_i. \tag{4.41}$$

Using completion of squares, the posterior distribution $p(\boldsymbol{\phi}_i|\mathbf{O}_i)$ is also Gaussian, for each set of input features $\mathbf{O}_i = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_N]$:

$$p(\boldsymbol{\phi}_i|\mathbf{O}_i) = \mathcal{N}(\boldsymbol{\phi}_i; \hat{\boldsymbol{\phi}}_i, \mathbf{L}_i^{-1}), \tag{4.42}$$

where "i-vector" is the MAP point estimate of the variable $\boldsymbol{\phi}_i$, with mean vector $\hat{\boldsymbol{\phi}}_i$ and precision matrix $\mathbf{L}_i$):

$$\hat{\boldsymbol{\phi}}_i = \mathbf{L}_i^{-1}\mathbf{T}^T\boldsymbol{\Sigma}^{-1}\mathbf{f}_i \tag{4.43}$$

$$\mathbf{L}_i = \mathbf{I} + \sum_{c=1}^{C} N_i^{(c)}\mathbf{T}^{(c)^T}\boldsymbol{\Sigma}^{(c)^{-1}}\mathbf{T}^{(c)}. \tag{4.44}$$

The input data representing an observation $\boldsymbol{O}$ are zero- and first-order statistics (as defined in Section 4.1.1). $\mathbf{T}^{(c)}$ represents sub-matrix of $\mathbf{T}$ corresponding to mixture component c:

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}^{(1)} \\ \mathbf{T}^{(2)} \\ \vdots \\ \mathbf{T}^{(C)} \end{bmatrix}. \tag{4.45}$$

Projection Matrix Estimation

When estimating $\mathbf{T}$, the objective is to maximize the likelihood $p(\mathbf{O}_i|\boldsymbol{\phi}_i, \mathbf{T})$ over the training data. Similar to GMM, we use the EM algorithm for $\mathbf{T}$-matrix estimation. It comprises two steps that are repeated iteratively. Again, for E-step, we compose the auxiliary function [Brümmer, 2009] (to simplify writing, $\langle \dots \rangle$ represent expectation $\mathbb{E}[\dots]$):

$$Q(\mathbf{T}, \mathbf{T}_0) = \sum_i \left\langle \log p(\boldsymbol{O}_i, \boldsymbol{\phi}_i|\mathbf{T}_0) \right\rangle \tag{4.46}$$

$$\log p(\mathbf{O}_i, \boldsymbol{\phi}_i|\mathbf{T}_0) = \log p(\boldsymbol{O}_i|\boldsymbol{\phi}_i, \mathbf{T}_0) + \log p(\boldsymbol{\phi}_i), \tag{4.47}$$

$\log p(\boldsymbol{\phi}_i)$ is set to the standard normal distribution and kept fixed. There is no need to re-estimate parameters of $p(\boldsymbol{\phi}_i)$ as any change in prior distribution can be equivalently accomplished by appropriately changing $\boldsymbol{\mu}$ and $\mathbf{T}$. We can simplify the auxiliary function:

$$\begin{aligned} Q(\mathbf{T}, \mathbf{T}_0) &= \sum_i \left\langle \log p(\boldsymbol{O}_i|\boldsymbol{\phi}_i, \mathbf{T}_0) \right\rangle_{\mathcal{N}(\boldsymbol{\phi};\hat{\boldsymbol{\phi}}_i,\mathbf{L}_i^{-1})} \\ &= \sum_i \left\langle \boldsymbol{\phi}_i^T\mathbf{T}^T\boldsymbol{\Sigma}^{-1}\tilde{\mathbf{f}}_i - \frac{1}{2}\boldsymbol{\phi}_i^T\mathbf{T}^T\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\mathbf{T}\boldsymbol{\phi}_i \right\rangle \\ &= \sum_i \left[\operatorname{tr}\left(\langle\boldsymbol{\phi}_i\rangle^T\mathbf{T}^T\boldsymbol{\Sigma}^{-1}\tilde{\mathbf{f}}_i \right) - \frac{1}{2}\operatorname{tr}\left(\langle\boldsymbol{\phi}_i\boldsymbol{\phi}_i^T\rangle\mathbf{T}^T\mathbf{N}_i\boldsymbol{\Sigma}^{-1}\mathbf{T} \right) \right]. \end{aligned} \tag{4.48}$$

In this section, we defined the normalized statistics around the GMM mean (4.38). This operation allows us to treat GMM mean super-vector $\mathbf{m}$ as a vector of zeros. Next, we can normalize first-order statistic $\mathbf{f}_i$ and $\mathbf{T}$ matrix by GMM covariance:

$$\bar{\mathbf{f}}_i^{(c)} = \boldsymbol{\Sigma}^{(c)^{-\frac{1}{2}}}\mathbf{f}_i^{(c)} \tag{4.49}$$

$$\bar{\mathbf{T}}^{(c)} = \boldsymbol{\Sigma}^{(c)^{-\frac{1}{2}}}\mathbf{T}^{(c)}, \tag{4.50}$$

where $\boldsymbol{\Sigma}^{(c)^{-\frac{1}{2}}}$ is a symmetrical decomposition (such as Cholesky) of an inverse of the GMM UBM covariance matrix $\boldsymbol{\Sigma}^{(c)}$. As a result, GMM covariance is equal to an identity matrix: $\boldsymbol{\Sigma}^{(c)} = \mathbf{I}$. This leads to further simplification:

$$\begin{aligned} Q(\mathbf{T}, \mathbf{T}_0) &= \sum_i \left[\operatorname{tr}\left(\tilde{\mathbf{f}}_i \langle \boldsymbol{\phi}_i \rangle^T \mathbf{T}^T \right) - \frac{1}{2} \operatorname{tr}\left(\langle \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T \rangle \mathbf{T}^T \mathbf{N}_i \mathbf{T} \right) \right] \\ &= \sum_i \left[\operatorname{tr}\left(\mathbf{C}_i \mathbf{T}^T \right) - \frac{1}{2} \operatorname{tr}\left(\mathbf{A}_i \mathbf{T}^T \mathbf{N}_i \mathbf{T} \right) \right], \end{aligned} \tag{4.51}$$

where

$$\begin{aligned} \mathbf{C}_i &= \tilde{\mathbf{f}}_i \langle \boldsymbol{\phi}_i \rangle^T, \\ \mathbf{A}_i &= \langle \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T \rangle = \langle \boldsymbol{\phi}_i \rangle \langle \boldsymbol{\phi}_i^T \rangle + \mathbf{L}_i^{-1}. \end{aligned} \tag{4.52}$$

Setting the derivation to zero, we can get a closed-form solution for estimation of hyperparameters:

$$\frac{\mathrm{d}Q(\mathbf{T}, \mathbf{T}_0)}{\mathrm{d}\mathbf{T}^{(c)}} = \sum_i \left[\mathbf{C}_i^{(c)} - \left(n_i^{(c)} \mathbf{A}_i \mathbf{T}^{(c)} \right) \right] = 0, \tag{4.53}$$

with accumulators, collected during the E-step:

$$\mathbf{T}^{(c)} = \sum_i \mathbf{C}_i^{(c)^T} \left(n_i^{(c)} \mathbf{A}_i \right)^{-1}. \tag{4.54}$$

The presented framework allows for different settings of dimensionalities for $\mathbf{T}$. Note that training of $\mathbf{T}$ does not require any speaker labels, i.e. it is trained in an unsupervised way. As a consequence, the i-vector contains both wanted and unwanted variability of the target classification task.

4.2 Discriminatively Trained Embedding — x-vector

The *x-vectors* represent an evolutionary step in the extraction of fixed-length feature vectors from recordings of any length. In Section 4.1, we described this issue from the point of view of generative models that resulted in the i-vector approach. With the impending mass deployment of neural networks, efforts to use this technology have intensified. The first experiments have originated in the GMM and i-vectors approach, in which the NN replaced or improved a single individual component in the i-vector extractor pipeline. This gave rise to techniques such as bottleneck features [Song et al., 2013, Yaman et al., 2012], DNN alignment [Lei et al., 2014], or Discriminatively Trained PLDA (DPLDA, described in [Burget et al., 2011]).

The individual parts were always trained for their given sub-task and they were not fully optimized for SV. The work of NN-based End-to-End SV by [Rohdin et al., 2018] was one of the attempts addressing this shortcoming. In this work, the authors proposed to replicate each component of the i-vector system with feed-forward NN. Each NN component was individually trained to mimic the original i-vector component. After the initial component-by-component training, the connected sub-networks were re-trained together. This resulted in a relatively large NN with a binary output for target/non-target trials.

The logical step was to experiment with discarding of the original concept of i-vectors and using NN independently of the original topology. Several works dealing with the

issue were published, such as Sequence summarizing neural network (SSNN) for speaker adaptation in ASR [Veselý et al., 2016], SSNN for LID [Pešán et al., 2016] or End-to-End SV by [Snyder et al., 2016], which came up with a new approach based purely on NNs.

Again, the systems have to deal with different lengths of utterance (sequences of input feature vectors). Most of the NNs in the mentioned works can be represented by the general model in Figure 4.2, in which the NN can be divided into three key parts:

The first part comprises processing of the feature vectors or their contexts and still generating a sequence of internal representations of variable length. This sequence represents the input to the next part of the network; we refer to it as the *pooling layer*, in which the sequence is reduced to a vector of a fixed length. Various methods can be used: In [Pešán et al., 2016], this was achieved by a calculation of mean over temporal dimension. In [Snyder et al., 2016], mean and standard deviation were used. This is followed by the part of the network with an **embedding extractor**. In [Pešán et al., 2016], mean values were used without any change and in [Snyder et al., 2016], several hidden dense layers were used. The last part of the network is the **classifier**, defined by the target task. For example, in [Pešán et al., 2016], the authors used multi-class logistic regression (directly targeting the task of language classification), on the other hand, [Snyder et al., 2016] used discriminative Probabilistic Linear Discriminant Analysis (PLDA)-like classifier.

Snyder's approach was an exciting concept, in which a neural network with a fraction of parameters compared to the i-vector model was able to achieve very competitive results. The disadvantage of neural nets lies in the need for in-domain data to make the system efficient (because of DPLDA). The binary output also requires paying more attention to batch creation for proper training.

Later, this system was modified in [Snyder et al., 2017]. The concept of the essential parts of NN was retained, but the classifier was replaced with a multi-class logistic regression with a class for each speaker in the training set, which deviates from the original end-to-end concept. The new concept does not allow direct use in speaker verification. The NN only serves as an embedding extractor. For the comparison itself, other techniques are used, usually Heavy-tailed PLDA (HPLDA) or cosine-similarity scoring (described in Section 5).

The research community has quickly adopted this approach and called the output embedding *x-vector*. Later, the term "x-vectors" has been widely adopted to describe any embedding that had been extracted using a NN, regardless of its exact topology.

4.2.1 Time Delayed Neural Network

Before we describe the x-vector topology itself, let us take a closer look at the type of network that handles the context of input features. Time Delay Neural Network (TDNN) is a type of multi-layer feed-forward deep neural network. First introduced in [Waibel et al., 1989], it was subsequently improved for better training efficiency in [Peddinti et al., 2015].

The motivation for using TDNN was to replace the Recurrent Neural Networks (RNN) in learning of long-term time dependencies. The training time of RNN is higher than feed-forward networks because of the sequential nature of the learning algorithm. TDNN is used to learn long-term time dependencies from longer contexts of standard features and it achieves similar learning speed to other forward networks [Peddinti et al., 2015].

An example of a TDNN scheme is shown in Figure 4.2. A standard DNN learns an affine transformation of the whole input context during processing of a wide time context in the first layer of the DNN. On contrary, TDNNs are learning to transform narrow context only in the first layer. Deeper layers process hidden activations from a broader temporal context,

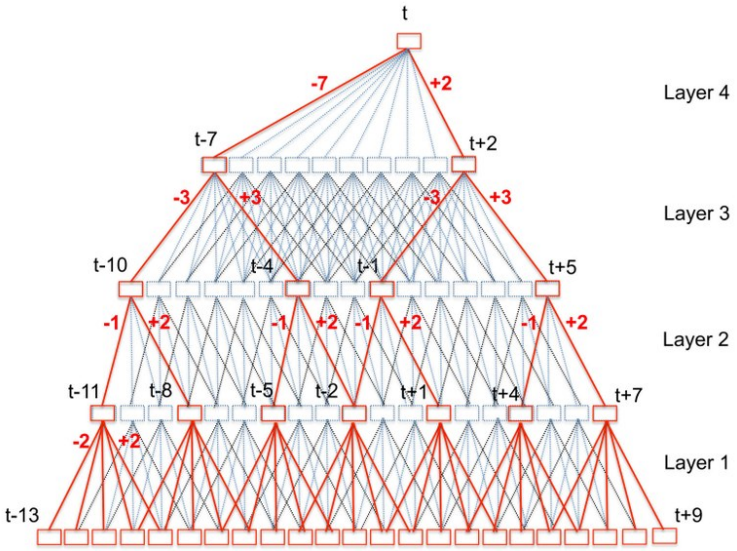


Figure 4.1: Original TDNN used for ASR. (Source: [Peddinti et al., 2015])

hence the ability of the higher layers to learn broader temporal relationships. Since each layer works with a different width of context, the layers have varying temporal resolution, which further increases with every deeper layer.

The transformations in the TDNN architecture are tied across time steps, therefore, during back-propagation, the lower layers of the network are updated by a gradient accumulated over all the time steps of the input temporal context. The TDNN is defined by the context of each layer that it uses to calculate its activation.

4.2.2 Original x-vector Network

An x-vector extractor is a feed-forward neural network that can compute a speaker embedding from variable-length series of input features. Let us now mainly focus on the original network design, published in [Snyder et al., 2017]. The topology of the neural network is depicted in Figure 4.2. As mentioned earlier, the network can be divided into three essential parts:

- Frame-level layers
- Statistics pooling layer
- Additional layers with classifiers

The frame-level part operates on speech (feature) frames. The primary purpose of this part of the network is to collect frame context information into frame-level network representation. A statistics pooling layer then aggregates over the frame-level representations into a segment representation. Additional layers then operate at the segment-level, and finally a soft-max output layer provides the means for training the whole network via multi-class cross-entropy. Throughout the whole network, Rectified linear units (ReLUs) are used as activation functions.

The original x-vector network consists of five frame-level TDNN-layers. Let us suppose t is the current time step. At the input, we splice together frames at $\{t-2, t-1, t, t+1, t+2\}$.

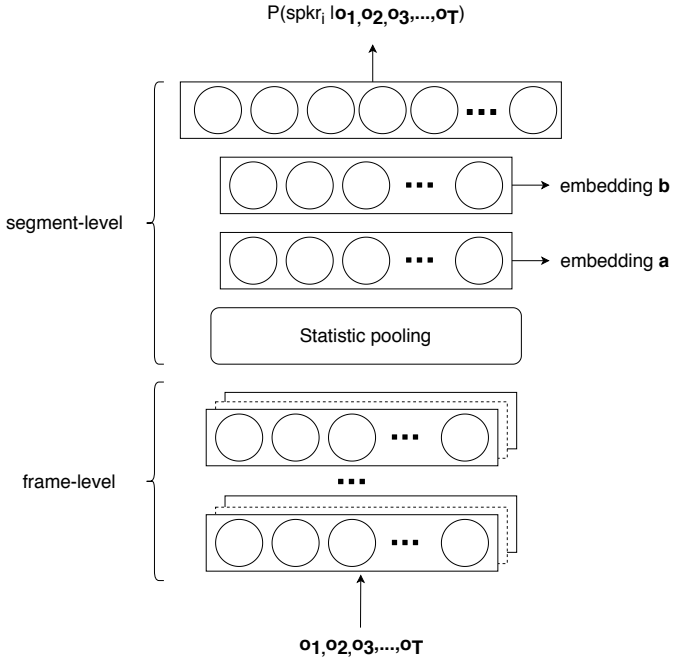


Figure 4.2: Diagram of an x-vector NN with two variants of embeddings (embedding **a** and embedding **b**) extracted from the network layers after the statistics pooling layer.

The next two layers splice the output of the previous layer together at times $\{t-2, t, t+2\}$ and $\{t-3, t, t+3\}$, respectively. The following two layers also operate on the frame-level, but without any added temporal context. The size of the layers varies from 512 to 1536 based on the used splicing context. Overall, the frame-level part of the network prepares frame-level representation from the time context of $t-8$ to $t+8$.

The statistics pooling layer receives the output of the final frame-level layer as an input, aggregates input segments over the time, and computes their mean and standard deviation.

These segment-level statistics are concatenated and forwarded to two additional hidden layers with dimension of 512. The two layers may be used to compute the embedding (x-vector). The last layer consists of a soft-max output. Initially, there were two variants of embeddings based on the layer used for extraction, *embedding-a*, and *embedding-b* (as depicted in Figure 4.2). Later, embedding-b became the standard.

The network is trained to classify training speakers using a multiclass cross-entropy objective function:

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \log P(spk_k|\mathbf{o}_T^n), \tag{4.55}$$

where $P(spk_k|\mathbf{o}_T^n)$ is the probability of speaker k given T input frames $\mathbf{o}_T$. The quantity d_{nk} is 1 if the speaker label for segment is k, otherwise it is 0. We assume there are N training segments with K speakers.

4.2.3 Variants of x-vector Network

For the sake of completeness, let us mention further development. Since the conception of x-vectors, there have been numerous variants; let us mention a few. The original topology

Table 4.1: Example of ETDNN variant of x-vector architecture.

Layer	Layer Type	Context	Size
1	TDNN-ReLu	t-2:t+2	512
2	Dense-ReLu	t	512
3	TDNN-ReLu	t-2,t,t+2	512
4	Dense-ReLu	t	512
5	TDNN-ReLu	t-3,t,t+3	512
6	Dense-ReLu	t	512
7	TDNN-ReLu	t-4,t,t+4	512
8	Dense-ReLu	t	512
9	Dense-ReLu	t	1500
10	Pooling (mean + std)	1:T	2×1500
11	Dense(Embedding)-ReLu	-	512
12	Dense-ReLu	-	512
13	Dense-Softmax	-	K

from [Snyder et al., 2017] was extended into multiple variants. In [Snyder et al., 2019], the authors introduce a variant, in which the frame-level part of the network is modified. Each TDNN-layer is followed with a single dense layer (also referred to as ETDNN). The final topology is summarized in Table 4.1.

Later, TDNN frame-level part of the network was replaced with a factorized TDNN with skip connections [Povey et al., 2018]. The FTDNN reduces the number of parameters of the network. The idea is to factor the weight matrix of each TDNN layer into a product of two low-rank matrices. The first of these factors is constrained to be semi-orthogonal. Semi-orthogonal constraints are expected to ensure that we do not lose information when projecting to a low-rank dimension. The final topology is summarized in Table 4.2.

In another variant, TDNN frame-level part is replaced with a residual network with 2D convolutions (ResNet) introduced in [He et al., 2016]. This topology was also used in a modified variant, in which the pooling layer was replaced with a learnable dictionary encoding (LDE) layer [Cai et al., 2018]. The LDE pooling assumes that frame-level representations are distributed in C clusters. The network learns a dictionary with cluster centers. This brings to mind the GMM i-vector paradigm.

There has also been a development in the objective function. Studies were concluded using Large Margin Cosine Loss [Deng et al., 2019]: this objective function was proven to be very successful, especially when using cosine similarity as a method of calculating the score (as described in Section 5.1). Formally, we define the Large Margin Cosine Loss as:

$$L = \frac{1}{N} \sum_{i=1}^{N} \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j \neq y_i} e^{s\cos(\theta_{j,i})}} \tag{4.56}$$

subject to

$$\begin{aligned} \mathbf{W} &= \frac{\mathbf{W}^*}{\|\mathbf{W}^*\|}, \\ \mathbf{x} &= \frac{\mathbf{x}^*}{\|\mathbf{x}^*\|}, \\ \cos(\theta_{j,i}) &= \mathbf{W}_j^T \mathbf{x}_i, \end{aligned} \tag{4.57}$$

Table 4.2: Example of F-TDNN variant of x-vector architecture.

Layer	Layer Type	Context factor 1	Context factor 2	Skip conn. from layer	Size	Inner size
1	TDNN-ReLu	t-2:t+2			512	
2	FTDNN-ReLu	t-2,t	t, t+2		725	180
3	FTDNN-ReLu	t	t		725	180
4	FTDNN-ReLu	t-3,t	t, t+3		725	180
5	FTDNN-ReLu	t	t	3	725	180
6	FTDNN-ReLu	t-3,t	t, t+3		725	180
7	FTDNN-ReLu	t-3,t	t, t+3	2,4	725	180
8	FTDNN-ReLu	t-3,t	t, t+3		725	180
9	FTDNN-ReLu	t	t	4,6,8	725	180
10	Dense-ReLu	t	1500			
11	Pooling (mean + std)	1:T	2×1500			
12	Dense(Embedding)-ReLu	-	512			
13	Dense-ReLu	-	512			
14	Dense-Softmax	-	K			

where N is the number of training samples, $\mathbf{x}_i$ is the i-th feature vector corresponding to the class label of y_i, $\mathbf{W}_j$ is the weight vector of the class j and $\theta_{j,i}$ is the angle between $\mathbf{W}_j$ and $\mathbf{x}_i$.

4.3 x-vectors vs. i-vectors

Let us briefly look at the fundamental difference between i-vectors and x-vectors. It lies in the principle of training. I-vector training is unsupervised with the goal to be the best possible low-dimensional representation of the original acoustic space (more precisely, representation of the GMM means hyperparameter subspace). As a result, i-vectors carry information not only about the speaker, but about all acoustic and linguistic conditions influencing the recording. From the speaker verification point of view, this represents an unwanted variability. From a broader data mining point of view, this is relatively advantageous — i-vectors represent a universal embedding, and one i-vector extractor system can also be used for other applications (e.g. language recognition, gender recognition). In contrast, x-vectors are trained to discriminate speakers. During their training, labels of speaker's identities in the training data are needed. Therefore, x-vector training is supervised. During the training, the x-vectors are forced to represent the speaker as best as possible, and, ideally, to eliminate any unwanted variability from the representation. This can result in a higher accuracy of speaker verification. However, they can be less suitable in other areas (language recognition) if trained for SV.

Chapter 5

Scoring

In this section, we describe the techniques used for comparison of two embeddings. We do this w.r.t. i-vectors due to historical context, however, the techniques are general and suitable for use with any type of embedding described in this work.

5.1 Cosine Similarity Scoring

In one of the first ways using i-vectors for speaker verification [Dehak et al., 2010], the authors used the Support Vector Machine (SVM) classifier. They used cosine distance as the kernel function for this classifier, for two i-vectors $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$, given as:

$$s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = \frac{\boldsymbol{\phi}_1^T \boldsymbol{\phi}_2}{\|\boldsymbol{\phi}_1\| \|\boldsymbol{\phi}_2\|}, \tag{5.1}$$

where $\boldsymbol{\phi}_1^T \boldsymbol{\phi}_2$ is an inner-product (dot product) of i-vectors, and $\|\cdot\|$ is an Euclidean length. However, better results were obtained by using the cosine distance directly as a scoring method. Note that such a score cannot be interpreted as a log-likelihood ratio (LLR) and must be calibrated [Brümmer, 2010] if needed. The advantage of the cosine scoring lies in low computational complexity. Note that the scores obtained in this way are symmetric and independent of possible swapping of enrollement and test i-vectors.

In the case of raw i-vectors, this method cannot be expected to be very accurate as it does not take into account the inter-session variability. PLDA explicitly models this variability (as discussed in Section 5.3). Experiments have been performed with several techniques that try to solve this modeling-related problem, such as Linear Discriminant Analysis (LDA), followed by Within-class Covariance Normalization (WCCN). Later, we discuss the use of cosine scoring in more detail for systems based on discriminative embeddings.

5.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) has found its place as a commonly performed step in the i-vector scoring pipeline. LDA aims at finding a linear subspace of the embedding space in which the classes (speakers) are best linearly separable.

We are looking for a linear transformation, where the original vector $\mathbf{x}$ from the $m-$dimensional space is transformed into the $n-$dimensional space using the $n \times m$ dimensional projection matrix $\mathbf{A}$:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i. \tag{5.2}$$

The best class separability can be achieved by a transformation that maximizes variance between classes while minimizing variance within classes (see Figure 5.1). This can be achieved using the Fisher discriminant ratio optimization criterion—a ratio between across-class and within-class variability [Bishop, 2006]:

$$J(\mathbf{A}) = \mathrm{tr}\{(\mathbf{A}\boldsymbol{\Sigma}_{wc}\mathbf{A}^T)^{-1}(\mathbf{A}\boldsymbol{\Sigma}_{ac}\mathbf{A}^T)\}, \tag{5.3}$$

where $\boldsymbol{\Sigma}_{ac}$ and $\boldsymbol{\Sigma}_{wc}$ are across-class and within-class covariance matrices, defined as:

$$\boldsymbol{\Sigma}_{ac} = \sum_{k=1}^{K} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{5.4}$$

$$\boldsymbol{\Sigma}_{wc} = \sum_{k=1}^{K} \sum_{n \in C_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T, \tag{5.5}$$

with class-dependent mean vectors

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in C_k} \mathbf{x}_n \tag{5.6}$$

$$\tag{5.7}$$

and global mean vector

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k. \tag{5.8}$$

The solution for $\mathbf{A}$ is given by m eigenvectors of $\boldsymbol{\Sigma}_{wc}^{-1}\boldsymbol{\Sigma}_{ac}$ corresponding to the m largest eigenvalues. For m-dimensional subspace, the data must contain at least $m + 1$ classes; in other words, $m \leq k - 1$.

5.3 Probabilistic Linear Discriminant Analysis

Initially, this technique was introduced for the purpose of face recognition [Prince and Elder, 2007]. In recent years, PLDA has become a state-of-the-art technique for comparison of embeddings in verification tasks.

In the PLDA model, i-vector $\boldsymbol{\phi}$ is considered to be a realization of a random variable, whose generation process can be described in terms of hidden variables. A number of various related models exist. Some of these models use different number of hidden variables, some use different priors. The two most popular variants are Gaussian PLDA, which assumes Gaussian priors, and Heavy-tailed PLDA, in which Student t-distribution is imposed on the priors. This heavy-tailed model can provide an improvement in terms of accuracy, but it is computationally more demanding [Kenny, 2010]. However, there is no close-form solution for the posterior probability. The advantage of the Heavy-tailed PLDA lies in the ability of the model to cope with raw i-vectors without previous normalization. It was shown that length-normalization helps Gaussian PLDA to achieve results comparable to Heavy-tailed PLDA [Garcia-Romero and Espy-Wilson, 2011]. This improvement is caused by the

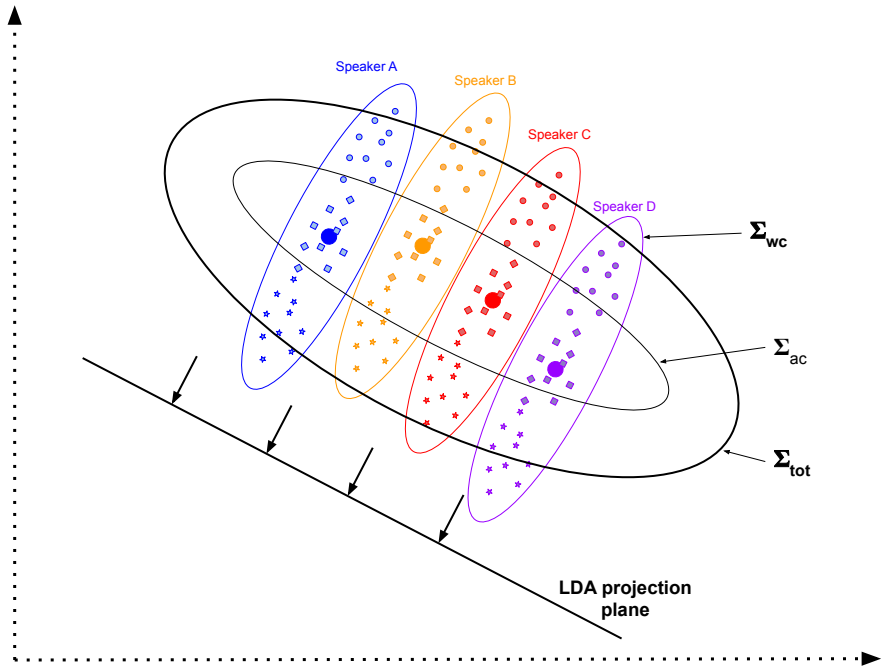


Figure 5.1: Demonstration of LDA (and also PLDA) assumptions about data. The bold dots correspond to the speaker means in the embedding space. The conditional distribution of embeddings around the speaker means sharing within the covariance matrix $\boldsymbol{\Sigma}_{\text{wc}}$. The distribution of all speaker means is then shown as a Gaussian with a covariance matrix $\boldsymbol{\Sigma}_{\text{ac}}$.

concentration of the probability mass of the Gaussian distribution in high-dimensional space. The length-normalization forces the i-vector to lie on a unit sphere, which brings them to the mass concentration in the Gaussian distribution shell.

5.3.1 General PLDA

The original PLDA model [Prince and Elder, 2007] assumes hat the i-vector $\boldsymbol{\phi}$ is decomposed as:

$$\boldsymbol{\phi} = \boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{\epsilon}, \tag{5.9}$$

where $\boldsymbol{\mu}$ is i-vector mean value, $\boldsymbol{V}$ represents the speaker subspace, $\boldsymbol{U}$ represents channel subspace, $\boldsymbol{y}$ and $\boldsymbol{x}$ are hidden variables representing speaker and channel. $\boldsymbol{U}$, $\boldsymbol{V}$ are typically low-rank matrices, representing speaker- and channel-subspace, respectively. $\boldsymbol{\epsilon}$ is a random variable representing a residual noise in data ($\boldsymbol{\epsilon}$ can be calculated once $\boldsymbol{y}$ and $\boldsymbol{x}$ are known). Aforementioned across- and within- class covariance matrix would be given as $\boldsymbol{\Sigma}_{ac} = \boldsymbol{V}\boldsymbol{V}^T$ and $\boldsymbol{\Sigma}_{wc} = \boldsymbol{U}\boldsymbol{U}^T$. For hidden variables, we assume a-priori normal distributions:

$$\begin{aligned} p(\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, \boldsymbol{I}) \\ p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x}; \boldsymbol{0}, \boldsymbol{I}) \\ p(\boldsymbol{\epsilon}) &= \mathcal{N}\left(\boldsymbol{\epsilon}; \boldsymbol{0}, \boldsymbol{D}^{-1}\right), \end{aligned} \tag{5.10}$$

where $\boldsymbol{D}$ is $M \times M$ diagonal covariance matrix of residual data variability.

5.3.2 Two Covariance PLDA

Another special kind of PLDA is *two-covariance* model, presented in [Brümmer and Villiers, 2010]. In this PLDA model, we assume that speaker- and channel-variability can be modeled by two Gaussians, with covariance matrices $\boldsymbol{\Sigma}_{ac}$ (sometimes called as across class covariance matrix) and $\boldsymbol{\Sigma}_{wc}$ (sometimes called as within class covariance matrix).

In the two covariance model, speaker identity is represented as a hidden variable $\boldsymbol{y}$ with normal distribution:

$$p(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}_{ac}\right), \tag{5.11}$$

where $\boldsymbol{\mu}$ is global mean vector and $\boldsymbol{\Sigma}_{ac}$ is the across-class covariance matrix. I-vector distribution of a known speaker represented by vector $\hat{\boldsymbol{y}}$ is given as:

$$p(\boldsymbol{\phi}|\hat{\boldsymbol{y}}) = \mathcal{N}\left(\boldsymbol{\phi}; \hat{\boldsymbol{y}}, \boldsymbol{\Sigma}_{wc}\right). \tag{5.12}$$

Visualization is similar to LDA presented in Figure 5.1.

5.3.3 Trial Scoring

In speaker recognition tasks, we usually assume a pair of sets of recordings $\mathbf{O}_1 = \{\boldsymbol{\phi}_1^{\mathbf{O}_1}, \boldsymbol{\phi}_2^{\mathbf{O}_1}, \cdots, \boldsymbol{\phi}_N^{\mathbf{O}_1}\}$ and $\mathbf{O}_2 = \{\boldsymbol{\phi}_1^{\mathbf{O}_2}, \boldsymbol{\phi}_2^{\mathbf{O}_2}, \cdots, \boldsymbol{\phi}_M^{\mathbf{O}_2}\}$ referred to as a *trial* (as described in Section 3.1). We always know that the recordings within a given set come from a single speaker. The goal is to decide whether the speaker is identical in both sets. The evaluation is symmetrical. There is no difference between enrollment and test set. Let us consider two hypotheses, the first hypothesis $\mathcal{H}_1$ states that both sets of recordings come from the same speaker, and the second hypothesis $\mathcal{H}_2$ represents a situation in which the speakers are different. The trial score $s(\mathbf{O}_1, \mathbf{O}_2)$ is then defined as a log of a ratio of joint conditional probabilities (also referred as log-likelihood ratio or LLR) given the two respective hypotheses:

$$s(\mathbf{O}_1, \mathbf{O}_2) = \log \frac{p(\mathbf{O}_1, \mathbf{O}_2|\mathcal{H}_1)}{p(\mathbf{O}_1, \mathbf{O}_2|\mathcal{H}_2)}. \tag{5.13}$$

Since the hidden variables in $\mathcal{H}_2$ associated with the speaker identity (speaker factors) $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_2$ are independent, we can break down the likelihood of the data and rewrite (5.13) as:

$$s(\mathbf{O}_1, \mathbf{O}_2) = \log \frac{p(\mathbf{O}_1, \mathbf{O}_2|\mathcal{H}_1)}{p(\mathbf{O}_1|\mathcal{H}_2)p(\mathbf{O}_2|\mathcal{H}_2)}. \tag{5.14}$$

In the case of hypothesis $\mathcal{H}_1$, both sets $\mathbf{O}_1$ and $\mathbf{O}_2$ are from the same speaker. So they share the same speaker factor $\hat{\boldsymbol{y}}$. Therefore, both $\mathbf{O}_1$ and $\mathbf{O}_2$ are generated with a $p(\boldsymbol{O}|\hat{\boldsymbol{y}})$ distribution. The product of likelihoods for $\mathbf{O}_1$ and $\mathbf{O}_2$ gives the joint likelihood that the speaker $p(\boldsymbol{O}|\hat{\boldsymbol{y}})$ generates two independent sets of i-vectors:

$$p(\mathbf{O}_1, \mathbf{O}_2|\hat{\boldsymbol{y}}) = p(\mathbf{O}_1|\hat{\boldsymbol{y}})p(\mathbf{O}_2|\hat{\boldsymbol{y}}). \tag{5.15}$$

The likelihood that both sets come from single speaker $\boldsymbol{y}$ is calculated by marginalization across all possible speakers:

$$
\begin{aligned}
p(\mathbf{O}_1, \mathbf{O}_2|\mathcal{H}_1) &= \int p(\mathbf{O}_1, \mathbf{O}_2|\boldsymbol{y}) p(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} \\
&= \int p(\mathbf{O}_1|\boldsymbol{y}) p(\mathbf{O}_2|\boldsymbol{y}) p(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} \\
&= \int \Big[\prod_{n=1}^{N} p(\boldsymbol{\phi}_n^{\mathbf{O}_1}|\boldsymbol{y}) \Big] \Big[\prod_{m=1}^{M} p(\boldsymbol{\phi}_m^{\mathbf{O}_2}|\boldsymbol{y}) \Big] p(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}.
\end{aligned}
\tag{5.16}
$$

In the case of hypothesis $\mathcal{H}_2$, both set $\mathbf{O}_1$ and $\mathbf{O}_2$ in the trial are generated from two different speakers, with two speaker factors $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_2$. Sets $\mathbf{O}_1$ and $\mathbf{O}_2$ are thus generated with distributions $p(\mathbf{O}_1|\hat{\boldsymbol{y}}_1)$ and $p(\mathbf{O}_2|\hat{\boldsymbol{y}}_2)$. As the hidden speaker factors are independent, the combined likelihood for both sets can be written as:

$$
p(\mathbf{O}_1, \mathbf{O}_2|\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2) = p(\mathbf{O}_1|\hat{\boldsymbol{y}}_1) p(\mathbf{O}_2|\hat{\boldsymbol{y}}_2). \tag{5.17}
$$

The likelihood for both sets being individually generated by any two different speakers $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ is calculated by marginalization across all possible speakers:

$$
\begin{aligned}
p(\mathbf{O}_1, \mathbf{O}_2|\mathcal{H}_2) &= \iint p(\mathbf{O}_1, \mathbf{O}_2|\boldsymbol{y}_1, \boldsymbol{y}_2) \, \mathrm{d}\boldsymbol{y}_1 \mathrm{d}\boldsymbol{y}_2 \\
&= \int p(\mathbf{O}_1|\boldsymbol{y}_1) p(\boldsymbol{y}_1) \, \mathrm{d}\boldsymbol{y}_1 \int p(\mathbf{O}_2|\boldsymbol{y}_2) p(\boldsymbol{y}_2) \, \mathrm{d}\boldsymbol{y}_2 \\
&= \int \Big[\prod_{n=1}^{N} p(\boldsymbol{\phi}_n^{\mathbf{O}_1}|\boldsymbol{y}_1) \Big] p(\boldsymbol{y}_1) \, \mathrm{d}\boldsymbol{y}_1 \int \Big[\prod_{n=1}^{N} p(\boldsymbol{\phi}_n^{\mathbf{O}_1}|\boldsymbol{y}_2) \Big] p(\boldsymbol{y}_2) \, \mathrm{d}\boldsymbol{y}_2.
\end{aligned}
\tag{5.18}
$$

By substitution of the conditional likelihoods (5.16) and (5.18) into (5.13), we get:

$$
s(\mathbf{O}_1, \mathbf{O}_2) = \log \frac{\int \Big[\prod_{n=1}^{N} p(\boldsymbol{\phi}_n^{\mathbf{O}_1}|\boldsymbol{y}) \Big] \Big[\prod_{m=1}^{M} p(\boldsymbol{\phi}_m^{\mathbf{O}_2}|\boldsymbol{y}) \Big] p(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}}{\int \Big[\prod_{n=1}^{N} p(\boldsymbol{\phi}_n^{\mathbf{O}_1}|\boldsymbol{y}_1) \Big] p(\boldsymbol{y}_1) \, \mathrm{d}\boldsymbol{y}_1 \int \Big[\prod_{n=1}^{N} p(\boldsymbol{\phi}_n^{\mathbf{O}_1}|\boldsymbol{y}_2) \Big] p(\boldsymbol{y}_2) \, \mathrm{d}\boldsymbol{y}_2}. \tag{5.19}
$$

For simplification, let us assume a single-session trial, i.e. a single i-vector in both datasets $\mathbf{O}_1 = \{\boldsymbol{\phi}_1\}$ and $\mathbf{O}_2 = \{\boldsymbol{\phi}_2\}$. In this case, we can rewrite the log-likelihood ratio as:

$$
s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = \log \frac{\int p(\boldsymbol{\phi}_1|\boldsymbol{y}) p(\boldsymbol{\phi}_2|\boldsymbol{y}) p(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y}}{p(\boldsymbol{\phi}_1) p(\boldsymbol{\phi}_2)}. \tag{5.20}
$$

More detailed description can be found in [Kenny, 2010], [Villalba and Brummer, 2011] and [Rajan et al., 2014].

Chapter 6

Multi-Conditional Training

Very often, models achieve high recognition accuracies on clean (or controlled) audio data in laboratory conditions, while their performance degrades when used in real-world applications, where the audio may come from a different channel and be affected by the environment noise and reverberation. In theory, optimal recognition performance would be obtained in matched conditions, i.e., for system training data coming from the same domain as the test data. Unfortunately, test conditions are often unknown or unpredictable.

Generally, when we speak about multi-conditional training — especially in connection to generative frameworks, such as i-vectors or PLDA — we often refer to a set of techniques rather than a single specific one. Typically, this term refers to training one or more parts of a generative framework on multi-conditional data to improve system robustness against some conditions. In this case, "condition" usually indicates the acoustic environment under which the test recording was made (channel, ambient noise, reverberation, etc.), usually unknown. Sometimes, this way of increasing the robustness of the system is also referred to as multi-style training.

The effect of noise and reverberation on an SV system based on i-vectors is described in [Mandasari et al., 2012] and [Ferrer et al., 2011b]. In [Mandasari et al., 2012], the authors discuss the effect of babble and car noise on i-vector based SV system with cosine and PLDA scoring. Among other things, they try to apply a Wiener filter [Wiener, 1964] to suppress the effect of noise. Wiener filtering has the ability to reduce the perceived noise in a signal by removing the spectral average of the noise. However, this requires having some knowledge of the models of the signal and noise.

In [Garcia-Romero et al., 2012], the authors focus on a wider range of noises. They propose an extension in which the final verification score is a soft mixture of scores produced by a collection of K subsystems. Each of these subsystems is trained according to a multi-condition training scheme. Each subsystem consists of a separate PLDA model for one of the K conditions. The rest of the i-vector framework is shared by all K subsystems. However, this method has a major disadvantage — the requirement to know the probable K-conditions occurring in the test. This also increases the memory and computational demands during scoring by the factor of K.

A simplification of this techniques was described in [Rajan et al., 2013]. The authors consider multi-conditional PLDA training or even multi-conditional trial scoring with more multi-conditional recordings on the enrolment side. Their system assumes K-training enrollment of recordings representing different conditions in the test. The authors achieved the best results when the multi-condition approach was used both during enrollment and PLDA training. However, it is also necessary to take into account single-session trials, in

which we have only one i-vector on the enrollment and test side. In such a case, it may be inappropriate to reproduce the enrollment i-vector with an artificial augmentation, especially if it already contains degradation in the form of other noise from the time of its acquisition.

The idea of multi-conditional training is very interesting. However, it is necessary to think about the correct implementation of such training. Otherwise, an incorrectly chosen procedure could lead to a degradation of the whole system. Let us assume more training recordings per speaker, with some form of degradation (noise, reverberation, channel). When each speaker in training is degraded with a different type of degradation, we face the risk that even unwanted noise information is used to distinguish between speakers and ends up in across-class variability during training. We need to determine the restriction that it is necessary to suppress this variability or ensure that this variability is included in the within-class variability of speakers.

In [McLaren and van Leeuwen, 2012], the authors studied different ways of estimating and averaging the between-class and within-class covariance matrices. In [Lei et al., 2012], the PLDA model is trained using pooled clean and noisy speech.

In short, the multi-condition training deals with the issue of training the system on a dataset that includes as diverse conditions as possible, such as different speaking styles [Lippmann et al., 1987], languages [Ghoshal et al., 2013] and others. This training approach generally contributes to an improvement in performance, even if the exact test conditions have never been observed before [Benesty et al., 2008].

The previously cited works have focused more on the multi-conditional training of mostly a single component throughout the pipeline — mostly PLDA. In [Ribas et al., 2015], the authors extend the idea of multi-conditional training to the entire pipeline of the i-vector and PLDA framework. Hence, it consists not only of re-training PLDA hyperparameters but also of the UBM model and i-vector extractor $\mathbf{T}$ matrix. Finally, the study in [Martínez et al., 2014] has assessed single-channel feature-domain noise compensation methods in combination with multi-conditional training.

In the following experiments, we try to evaluate the multi-conditional training and the scope of its effectiveness (also presented in [Novotný et al., 2019c]).

6.1 Experimental Setup

In this section, we experiment with multi-conditional training. We focus mainly on training an i-vector extractor and a PLDA (multi-conditional training of discriminative x-vectors is discussed later in this work). Our goal is to show that multi-conditional training is a technique that can increase the robustness of the system against the negative influences mentioned in Chapter 2. However, we try to answer the question of whether this technique makes sense in all steps of the generative SV system based on i-vectors. In the experiments, we focus mainly on noise and reverberation, because these are the two most common factors that significantly contribute to recognition performance degradation.

6.1.1 Evaluation Set

The results of our experiments are reported on a subset of benchmarks listed in Section 3.3.5. The evaluation metric is EER presented in Section 3.1.2.

6.1.2 System Description

In this system, we used MFCCs features, extracted using a 25 ms Hamming window. We used 24 Mel-filters and we limited the bandwidth to the 120–3800 Hz range. 19 MFCCs together with zero-*th* coefficient were calculated every 10 ms. This 20-dimensional feature vector was subjected to short time mean- and variance-normalization using a 3 s sliding window. Delta and double delta coefficients were then calculated using a five-frame window, resulting in a 60-dimensional feature vector.

The modelling in this system is based on i-vectors. To train the i-vector extractor, we use 2048-component diagonal-covariance Universal Background Model (GMM-UBM), and we set the dimensionality of i-vectors to 600. We then apply LDA to reduce the dimensionality to 200. Such i-vectors are then centered around a global mean followed by length normalization [Dehak et al., 2010, Garcia-Romero and Espy-Wilson, 2011].

For augmenting the PLDA training set, we created new artificially corrupted training sets from the PRISM training data. We used noises and RIRs described in Section 3.3. To mix the reverberation, noise and signal at given SNR, we followed the procedure outlined in Figure 3.6, but omitting the last step of applying the telephone channel. We trained the four PLDAs with the following abbreviations used further in the text:

- **Clean**: PLDA was trained on original PRISM data, without augmentation.
- **N**: PLDA was trained on i) original PRISM data, and ii) portion (24k segments) of the original training data corrupted by noise.
- **RR**: PLDA was trained on i) original PRISM data, and ii) portion of the original training data corrupted by reverberation using real room impulse responses.
- **RR+N**: PLDA was trained on i) original PRISM data, ii) noisy augmented data, and iii) reverberated data.

Note that the sizes of all 3 augmentation sets are the same. The **RR+N** augmentation set was also used to train the augmented variant of the i-vector extractor **T** matrix.

6.1.3 Results

The comparison of our MFCC i-vector extractor trained on the original clean data and augmented data is shown in Table 6.1. We see some improvement in some of the benchmarks, but mostly a degradation. For each benchmark, the best results are highlighted in **bold**. Based on highlighted resutls, we can see that multi-conditional training of only PLDA dominates compared to multi-conditional training of **T** matrix or **T** matrix and PLDA. A closer look at the best results in the case of multi-conditional training of **T** matrix shows that the gain compared to **T** matrix trained on original data is minimal. I-vector extractor trained on original data with multi-condition training of PLDA provides more stable and better performance over most test benchmarks.

We believe that the reason for such a behavior is that i-vector extraction training is unsupervised (w.r.t. speaker labels). When we add augmented data to the training list, i-vector extractor training is forced to reserve a portion of the parameters to represent the variability of noise and reverberation, limiting parameters for speaker variability. In the supervised discriminative x-vector approach, we are forcing the x-vector extractor to do the opposite: it is forced to distinguish among the speakers, and data augmentation in training can be beneficial as we will see in Chapter 10. In PLDA, multi-conditional

Table 6.1: Results (EER [%]) of i-vector extractor trained on clean data (iX ORIG) compared to i-vector extractor trained on augmented data (iX AUG). Blocks are divided into columns corresponding to systems trained in multi-condition fashion (with noised and reverberated data in PLDA). Each column corresponds to a different PLDA multi-condition training set: "—" - clean condition, N - noise, RR - real reverberation, RR+N - real reverberation + noise. Each value set in bold is the minimum in the particular benchmark.

	iX ORIG				iX AUG			
Benchmark	—	N	RR	RR+N	—	N	RR	RR+N
tel-tel	1.99	2.39	1.99	2.74	**1.98**	2.44	1.96	2.86
sre16-tgl-f	21.85	**21.37**	21.84	21.88	22.33	21.95	22.06	22.62
sre16-yue-f	11.20	**10.52**	11.15	11.53	11.32	10.59	11.26	11.20
int-int	4.57	4.70	4.49	4.55	4.52	4.88	**4.44**	4.71
int-mic	**1.85**	2.09	1.86	2.00	2.11	2.17	2.04	2.02
prism,chn	1.03	1.29	0.99	0.97	**0.92**	1.20	0.95	1.04
sitw-core-core	10.11	10.13	**10.06**	10.32	10.28	10.38	10.17	10.34
prism,noi	3.72	**3.02**	3.65	3.42	3.79	3.03	3.73	3.26
prism,rev	2.51	2.67	2.40	2.23	2.74	2.80	2.55	**2.22**
BUT-RET-orig	**2.29**	2.56	2.30	2.33	2.56	2.68	2.47	2.64
BUT-RET-merge	14.43	14.33	13.79	11.22	11.16	11.08	10.80	**9.06**

training supposedly has the benefit that channel subspace is trained in such a manner that it captures the variability provided by the condition variety, therefore it has the ability to compensate for the channels from different conditions.

Multi-condition training of an x-vector extractor system is a more complex problem. For generative models, such as i-vector systems, there are looser requirements on the number of recordings in a dataset, the number of speakers in the dataset, or the number of recordings per single speaker. X-vector systems are significantly more sensitive to these properties, and the data-set composition must be taken into account in routine training. For this reason, dedicated Chapter 10 is devoted to the influence of data on x-vector training.

Chapter 7

Speech Enhancement

Speech signals can be contaminated with many factors, e.g. noise or reverberation (as in Section 2). By the term "speech enhancement" we generally refer to all techniques leading to improved perceived quality of the captured speech signal.

One of the most important properties of a robust system is the ability to cope with the distortions caused by noise, reverberation and the transmission channel itself. In Chapter 6, we described one solution on how to tackle this problem in the last stage of SV, where we used multi-condition training [Martínez et al., 2014, Lei et al., 2012] of PLDA, which introduced artificial noise and reverberation variability into the within-class variability of speakers. This approach can be further combined with a domain adaptation [Glembek et al., 2014] or variability compensation [Aronowitz, 2014], which requires having a certain amount of usually unlabelled target data. In the very last stage of the system, SV outputs can be adjusted via various kinds of adaptive score normalization [Sturim and Reynolds, 2005, Matějka et al., 2017, Swart and Brümmer, 2017].

Another way to increase the robustness is to focus on the quality of the input acoustic signal/features and to enhance it before it enters the SV system. Several techniques were introduced in the field of microphone arrays, such as active noise canceling, beamforming, and filtering [Kumatani et al., 2012]. For single microphone systems, front-ends utilize signal preprocessing methods, for example, weighted prediction error (WPE) [Nakatani et al., 2010], Wiener filtering, adaptive voice activity detection (VAD), gain control, etc. [ETSI, 2007]. Various designs of robust features [Plchot et al., 2013] can also be used in combination with normalization techniques such as cepstral mean and variance normalization or short-time Gaussianization [Pelecanos and Sridharan, 2006].

At the same time, when the NNs were finding their way into basic components of the SV systems, the interest in the NNs has also increased in the field of signal pre-processing and speech enhancement. An example of a classical approach to remove a room impulse response is proposed in [Dufera and Shimamura, 2009], in which the RIR filter is estimated by an NN. NNs have also been used for speech separation in [Yanhui et al., 2014]. A NN-based autoencoder for speech enhancement was proposed in [Xu et al., 2014a] with optimization in [Xu et al., 2014b] and finally, reverberated speech recognition with a signal enhancement by a deep autoencoder was tested in Chime Challenge and presented in [Mimura et al., 2014].

In this chapter, we focus on improving the robustness of SV via DNN autoencoder as an audio pre-processing front-end. The autoencoder is trained to learn how to map a noisy and reverberated speech to a clean speech. The advantage of Speech enhancement as a signal-preprocessing step can be used in various tasks, such as ASR, LID, or SV, independently on

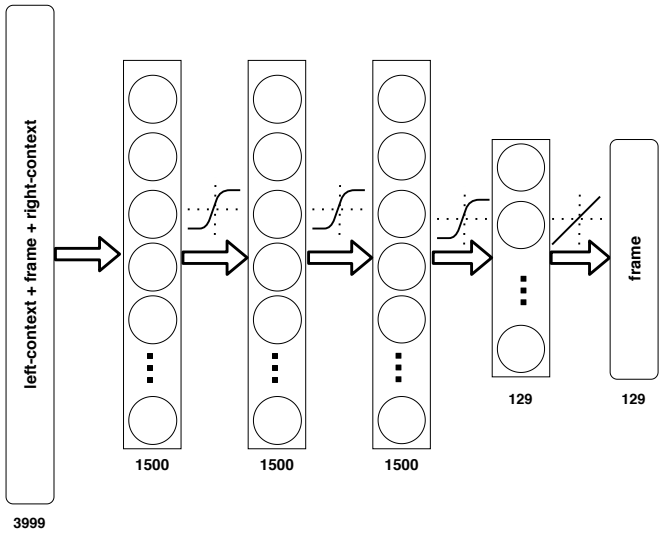


Figure 7.1: Topology of the autoencoder: three hidden layers each with 1500 neurons and hyperbolic tangent activation functions, output layer with 129 neurons and linear activation functions representing a single frame of dennoised (cleaned) log-magnitude spectrum. The input of the network are 31 concatenated frames of the 129-dimensional log-magnitude spectrum.

the used feature extraction pipeline. No interventions are necessary in the original system. This chapter is based on [Novotný et al., 2018a, Novotný et al., 2019c].

7.1 Signal Enhancement Autoencoder

Our autoencoder, introduced in [Plchot et al., 2016a] and in [Novotný et al., 2018a], consists of three hidden layers with 1500 neurons in each layer. The input of the autoencoder is a 129-dimensional log-magnitude spectrum with a context of +/- 15 frames (in total $31 \times 129 = 3999$-dimensional input). The output is a 129-dimensional enhanced central frame log-magnitude spectrum, see the topology in Figure 7.1.

It was necessary to perform feature normalization during the training and then repeat a similar process during actual denoising. We used the mean and variance normalization with mean and variance estimated per input utterance. At the output layer, de-normalization with parameters estimated on a clean variant of the file was used during training while during denoising, the mean and variance were global and estimated on the cross-validation set. Using log on top of the magnitude spectrum decreases the dynamic range of the features and leads to a faster convergence.

As an objective function for training the autoencoder, we used the Mean Square Error (MSE) between the autoencoder outputs from training utterances $\mathbf{x}_i^n$ and spectra of their clean variants $\mathbf{x}_i^c$:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i^n - \mathbf{x}_i^c)^2 \tag{7.1}$$

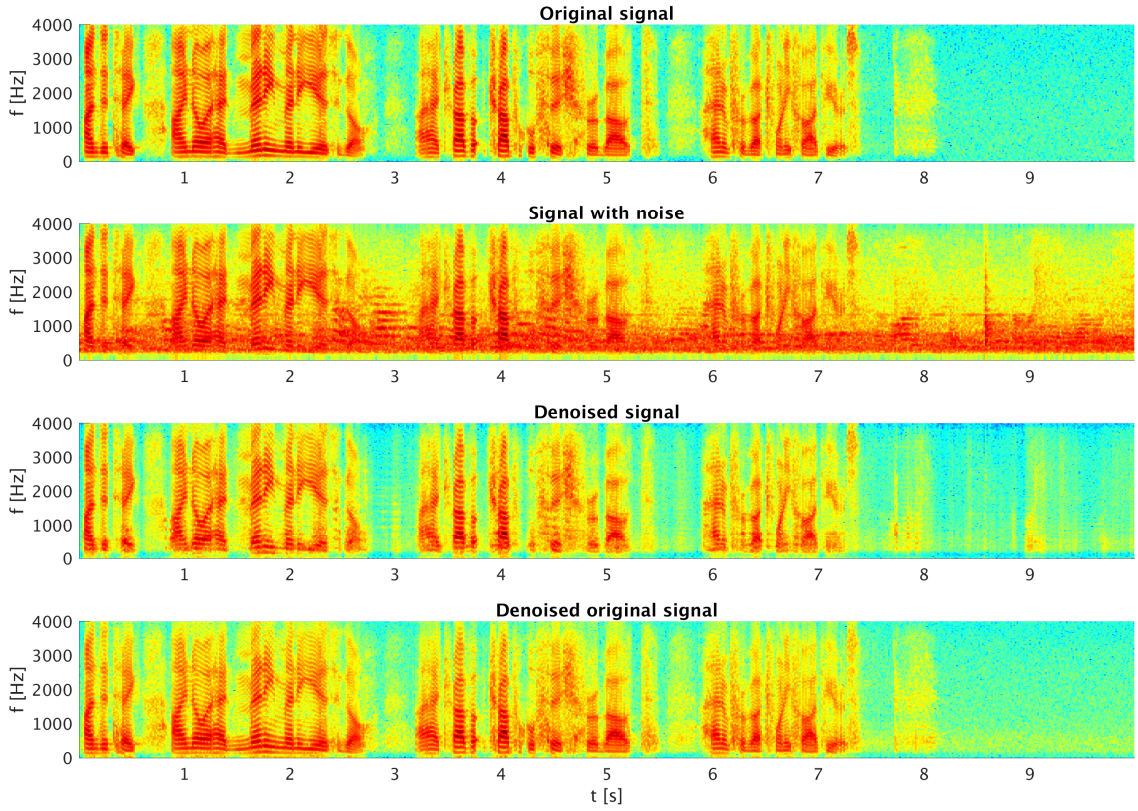


Figure 7.2: From top to bottom: spectrogram of original clean signal, spectrogram of signal with street noise with noise added at 10dB SNR, spectrum of enhanced corrupted signal. The last spectrogram is from enhanced original clean signal.

We were using both clean and augmented (noisy) recordings during the training as we wanted the autoencoder to be both robust and producing good results on relatively clean data. An example of autoencoder input and output is shown in Figure 7.2.

7.2 Multi-Conditional Training vs. Speech Enhancement

Let us now present an experiment that focuses on the comparison of speech enhancement using an NN autoencoder and multi-conditional training of PLDA, presented in Chapter 6, where it was already shown that performing multi-conditional training with added noisy and reverberated data helps significantly in SV based on i-vectors. We also discuss the influence of quantity and type of autoencoder training data on the performance of the analyzed SV system.

7.2.1 Experimental Setup

Evaluation Set

We evaluated our systems on the *female* portions of NIST SRE 2010 and PRISM benchmarks listed in Section 3.3.5.

Additionally, we created new artificially corrupted evaluation sets from the NIST 2010 *tel-tel* benchmark. The process was the same as described in Section 3.3.3 while using the tests portion of our noise and reverberation sets. We created seven new benchmarks:

- **rev-tel-tel**: SRE 2010 *tel-tel* benchmark corrupted by real room impulse responses (reverberation).
- **noi-∗-tel-tel**: SRE 2010 *tel-tel* benchmark corrupted by noise. We used three ranges of noise: 0-7dB, 7-14dB, 14-21dB (range is writen on position of ∗, e.g. noi-0-7-tel-tel).
- **rev-noi-∗-tel-tel**: SRE 2010 *tel-tel* benchmark corrupted by noise and real room impulse responses. Again, we used three ranges of noise: 0-7dB, 7-14dB, 14-21dB.

The difference between these new benchmarks and the benchmarks based on the PRISM set is in more realistic reverberation. Benchmark **prism,rev** is created from clean microphone data corrupted with artificially generated RIRs. The new benchmarks focus on adding a real reverberation to the telephone data. Similarly, while the **prism,noi** benchmark is created from microphone data by adding the noise at three levels of SNR (8dB, 15dB, 20dB), the new benchmarks use telephone data and randomly chosen SNR levels from the given intervals. Additionally, the selected telephone data tend to be more difficult than the microphone data used in the PRISM benchmarks. The recognition performance is evaluated in terms of the EER.

System Setup

In our experiments, we used cepstral features, extracted using a 25 ms Hamming window. We used 24 Mel-filters and we limited the bandwidth to the 120–3800 Hz range. 19 MFCCs together with zero-*th* coefficient were calculated every 10 ms. This 20-dimensional feature vector was subjected to short time mean- and variance-normalization using a 3 s sliding window. Delta and double delta coefficients were then calculated using a five-frame window giving a 60-dimensional feature vector. After feature extraction, voice activity detection (VAD) was performed by VAD described in Section 1.3.

To train i-vector extractors, we always use 2048-component diagonal-covariance Universal Background Model (GMM-UBM) and we set the dimensionality of i-vectors to 600. We apply LDA to reduce the dimensionality to 200. Such processed i-vectors are then transformed by global mean normalization and length-normalization.

We used the PRISM training dataset definition (in Section 3.2.4) without added noise or reverb to train UBM and i-vector extractor. Five variants of gender-independent PLDA were trained: one only on the clean training data, the rest included also artificially added different mixtures of noises and reverb. Artificially added noise and reverb segments totaled approximately twenty-four thousand segments or 30 % of total number of clean segments for PLDA training.

Autoencoder Variants

Fisher English database parts 1 and 2 were used for training the autoencoder. We trained five different autoencoders for signal enhancement. Two were trained only for dereverberation. The first was trained with artificially generated reverberation (**AR**) and the second used real reverberation (**RR**). The third autoencoder was trained only for denoising (**N**). The last two autoencoders were trained for both denoising and dereverberation. Again, one of them used artificially generated RIRs (**N+AR**) and the second one used the real ones (**N+RR**). Data and algorithm used for augmentation was described in Section 3.3.

PLDA Variants

Similarly, we created five different multi-condition training sets for PLDA. This approach is the same as in the autoencoder training. We used exactly the same noises and reverberation for segment corruption as in autoencoder training, which allowed us to compare the performance when using the autoencoder or multi-condition training.

Results

We provide a set of results for answering two questions: (i) How does the speaker recognition performance depend on the type of the enhancement (denoising, dereverberation, both) and amount or type (real, artificial) of the autoencoder training data? (ii) How does using the autoencoder compare to using the multi-condition data for SRE system training? In the end we also combine the autoencoder with the multi-condition training and find the best performing combination.

Our results are listed in Table 7.1. They are separated into two main blocks: PLDA trained on the clean data and PLDA trained on the multi-condition data. Each block is additionally separated to highlight whether the autoencoder enhancement is used or not.

In the first block, the baseline corresponds to the system where the PLDA was trained only on the clean data without any enhancement. The next five columns represent results when using different autoencoders: N - autoencoder trained only on the noised data, AR - autoencoder trained on the data corrupted with artificially generated RIRs, RR- autoencoder trained on the data corrupted with the real RIRs. N+(A/R)R - autoencoder simultaneously trained on the data with both types of distortion (noise and reverberation).

In the second block, we list the results for multi-condition training. We trained five different PLDAs, every time using a different mix of corrupted data added to the training list. PLDA or autoencoder on its own cannot fully profit from the added corrupted data. Autoencoder is able to partially remove the noise and reverberation from the data, while PLDA can learn the effect these data have for within- and across- speaker variability. Combining both techniques naturally brings the most improvement as we can see from the last block in Table 7.1. In these experiments, we were again modifying the data for the multi-condition PLDA training, but all of this data was previously processed by a single autoencoder. We decided to use the autoencoder simultaneously trained on the noisy and reverberated data (using real RIRs). This autoencoder was chosen based on its good and consistent performance in various benchmarks and we believe that it could represent a universal preprocessing step as there is only a negligible drop in performance on clean data (see for example the performance on tel-tel benchmark of baseline system versus the N+RR column in the first block in Table 7.1).

Now, let us focus on comparing the baseline system and the system trained on enhanced data (PLDA is trained only on clean and enhanced data). In these experiments, we study which autoencoder training dataset is the best for given benchmark. If we look at these results globally, we can see that for most of the reverberation benchmarks (*prism,rev, int-int, int-mic and rev-tel-tel*, with exception of *prism,chn*), the autoencoder trained on the real reverberation provides the best results. Similar situation occurs for noisy benchmarks (*prism,noi, noise-∗-tel-tel*) and noisy end reverberated benchmarks (*rev-noise-∗-tel-tel*). These results confirm our intuition, that it is best to use the autoencoder trained on the matching distortion to remove its effect from the data. We can also observe that to remove the reverberation, it is best to train on data reverberated by real RIRs instead of those artificially generated. This holds even for the benchmark containing only artificial reverberation (*prism,rev*). In general, when looking at the first block in Table 7.1, all of the autoencoders trained using reverberation with real RIRs (columns RR, N+RR) are better than those trained using artificial RIRs (AR, N+AR). We can also see, that the difference in performance between the RR-autoencoder and the N+RR autoencoder is rather small more in favor of the latter, both in reverberation and noisy benchmarks. This indicates that using the N+RR autoencoder is a good universal choice and justifies its selection for the experiments when combining the audio enhancing with multi-condition training.

When focusing on the multi-condition training (first part of the second block in Table 7.1) and taking the global view, we can observe similar trends as in the pure enhancement task. If we want to remove some type of distortion, it is best to add the matching distortion type into the PLDA training. If we look more closely, we can see the difference in reverberation benchmarks based on the PRISM set, where (as opposed to the enhancement) the multi-condition system using artificially generated RIRs has better results. This can indicate that it is easy for the PLDA to capture the channel variability caused by reverberating with the artificial RIRs which results in better performance in this matched-condition scenario. This hypothesis is further strengthened when comparing the AR with RR on rev-tel-tel benchmark: training on the matched-condition RR data almost halves the error rate.

If we analyze the difference in performance between the pure signal enhancement and the multi-condition training, we see that the multi-condition training has slightly better results, especially in the hardest benchmarks *rev-∗-tel-tel*. In the clean tel-tel benchmark, we can see that using autoencoder harms the performance less than multi-condition training. Additionally, in some PRISM-based benchmarks (*prism,rev, int-int, prism,chn*), the autoencoder is also better than multi-condition training.

Finally, we look at the combination of both techniques (the very last block in Table 7.1). Here, we are still having the same training lists for multi-condition PLDA training, but additionally, all data are enhanced by autoencoder trained on noised and reverberated data with real RIRs. We can see that in most benchmarks, we improve results compared to the pure multi-condition training. We suffer a significant degradation in clean tel-tel benchmark with respect to baseline for N+AR and N+RR training, but especially in the case of the latter, this degradation is compensated by excellent performance in other benchmarks, especially the most difficult *rev-noise-∗-tel-tel* where we gain more than 70 % relative improvement over the baseline.

The combination of both techniques can also eliminate the big difference between artificially generated reverberation and real reverberation as can be seen by comparing results of N+AR and N+RR systems. As we already saw for pure multi-condition training, the best results are again achieved by using the matched distortion for PLDA training, but the differ-

ence between the best possible results and multi-condition training with N+RR autoencoder are small. This justifies our recommendation to use the **combination of multi-condition training with N+RR data that were preprocessed by the N+RR autoencoder** as a universal and robust system, especially when expecting reverberated and/or noisy test data.

Table 7.1: Results (EER [%]) obtained in four scenarios. The first two blocks correspond to the system trained only with clean data (PLDA trained on clean data). In the left block, scores of baseline system are displayed. In the right block, the score of the clean system with autoencoder signal enhancement is displayed. Results of five autoencoders trained on: N - noise, (A/R)R- artificial/real reverberation, or both (+) are presented in each column. The last two blocks correspond to systems trained in multi-condition fashion (with noised and reverberated data in PLDA). Results in each column correspond to different PLDA multi-condition training set: N - noise, (A/R)R- artificial/real reverberation, or both (+). The very last block present results of the combination of both techniques. For combination, we select autoencoder trained on noised and reverberated data with real reverberation (N+RR). Each value set in bold is the minimum in the particular benchmark for combining the PLDA system and the best autoencoder.

	PLDA trained on **clean** data						PLDA trained on **multi-condition** data									
	baseline	Autoencoder training					PLDA extension data					Autoencoder (N+RR) + PLDA extension data				
Benchmark		N	AR	N+AR	RR	N+RR	N	AR	N+AR	RR	N+RR	N	AR	N+AR	RR	N+RR
tel-tel	2.06	2.06	2.09	2.07	**1.99**	2.06	2.46	2.07	2.73	**2.04**	2.79	2.48	2.07	2.68	2.14	2.75
prism,noi	2.95	**2.12**	2.50	2.26	2.47	2.19	**2.27**	3.08	2.52	2.93	2.46	**1.97**	2.24	2.04	2.24	2.06
prism,rev	2.07	1.75	1.62	1.61	**1.51**	1.56	2.22	**1.54**	1.62	1.61	1.63	1.58	**1.42**	1.39	1.42	**1.42**
int-int	1.76	1.79	1.69	1.77	**1.63**	1.79	1.86	**1.68**	1.76	1.67	1.71	1.81	**1.69**	1.71	1.71	1.76
int-mic	1.09	1.14	1.09	1.15	**1.01**	1.11	1.23	**0.77**	0.92	0.96	1.04	0.98	1.00	**0.84**	0.98	0.94
prism,chn	0.79	0.52	0.60	**0.40**	0.60	0.43	1.00	**0.54**	0.67	0.63	0.76	0.46	0.34	0.37	**0.27**	0.40
rev-tel-tel	19.37	14.76	11.18	13.45	**9.14**	9.37	17.84	9.46	10.15	**5.24**	6.60	8.29	6.14	5.85	**4.06**	4.76
noi-14-21-tel-tel	4.96	**3.30**	4.01	3.94	3.72	3.70	**2.90**	4.61	3.53	4.32	3.39	**2.68**	3.21	2.96	3.02	2.98
noi-7-14-tel-tel	8.29	**5.12**	6.81	5.71	6.66	5.75	**3.94**	8.03	4.92	7.54	4.72	3.53	5.08	3.72	4.60	**3.52**
noi-0-7-tel-tel	18.95	**10.68**	15.52	11.28	15.87	12.28	**8.83**	18.78	9.55	18.12	9.58	**6.08**	11.40	6.25	10.01	6.38
rev-noi-14-21-tel-tel	16.52	15.10	11.04	11.36	9.40	**7.63**	16.13	11.08	8.34	8.94	**6.39**	6.38	6.10	4.80	4.95	**4.14**
rev-noi-7-14-tel-tel	19.54	19.90	13.99	15.62	12.35	**9.61**	17.17	16.52	10.25	14.20	**8.31**	7.17	8.18	5.63	7.03	**5.12**
rev-noi-0-7-tel-tel	27.83	28.15	22.19	24.52	21.44	**16.84**	21.56	26.68	15.63	25.53	**14.66**	10.53	15.61	8.77	14.37	**8.15**

7.3 Speech Enhancement and Other Discriminative Approach in SV

In previous Section, we explored the benefits of DNN-based audio pre-processing with standard generative SV systems based on i-vectors and PLDA in the previous Section 7.2. In this section, we propose an additional improvement over a system, in which the SV is already improved by DNN (during embedding or feature extraction).

We use the x-vector architecture (described in Section 4.2), which already presents the *x-vector* (the embedding) as a robust feature for PLDA modeling, and provides state-of-the-art results across various acoustic conditions [Novotný et al., 2018b].

We experiment with the use of the signal enhancement autoencoder as a pre-processing step during training of the x-vector extractor or just during the embedding extraction. For further comparison with the best i-vector system, we also experiment with Stack Bottleneck (SBN, as described in Chapter 8) features concatenated with MFCCs to train our x-vector and i-vector extractor. A standard i-vector system with MFCC features is kept as the primary baseline.

We have mentioned systems trained with SBN. SBNs represent another way of integrating the discriminative approach in the originally generative model. This issue is discussed in more detail later in Chapter 8. Now we look at it merely as another, more robust feature extraction method.

We compare four SV systems, which combine two essential feature extraction techniques—MFCC, and Stack Bottle-neck features (SBNs) concatenated with MFCCs—and two front-end modelling techniques—the i-vectors and the x-vectors. Please note that each of the modeling techniques uses a slightly different MFCC extraction. All systems use the same voice activity detection from Section 1.3.

7.3.1 MFCC i-vector System

This system is the same as the system described in Section 7.2.1. We used 60-dimensional cepstral features, delta and double delta coefficients. To train the i-vector extractor, we use 2048-component diagonal-covariance GMM-UBM, and we set the dimensionality of i-vectors to 600, with LDA reduction to 200. Such i-vectors are then centered around a global mean and length-normalized

7.3.2 SBN-MFCC i-vector System

SBN is a type of features extracted from the internal state of NN. A detailed description of SBNs, their extraction, properties, and NN topology is discussed in Section 8.1 in Chapter 8 focused on discriminative approaches in generative SV.

The 30-dimensional SBN were concatenated with MFCC features (as used in the previous system) and used as an input to the conventional GMM-UBM i-vector system, with 2048 components in the UBM and 600-dimensional i-vectors.

To train the UBM and the i-vector extractor, we used the PRISM training dataset definition without added noise or reverberation.

7.3.3 x-vector Systems

These systems are based on the DNN architecture for the extraction of embeddings as described in Section 7.3.3. Specifically, we use the original Kaldi recipe [Snyder, 2017] and 512-

dimensional embeddings extracted from the first layer after the pooling layer (*embedding-a*), which is consistent with [Snyder et al., 2018].

Input features to the DNN were MFCCs, extracted using a 25 ms Hamming window. We used 23 Mel-filters and we limited the bandwidth to 20–3700 Hz range. 23 MFCCs were calculated every 10 ms. This 20-dimensional feature vector was subjected to short time mean- and variance-normalization using a 3 s sliding window. Note the differences to the MFCC features for i-vector system described above (mainly the number of Mel-filters, bandwidth, no delta/double delta coefficients).

In addition, we also trained an x-vector extractor on MFCC features concatenated with SBN from Section 7.3.2. Apart from changing the input features, we kept the architecture of the embedding DNN the same as for the MFCC system. Data used for NN training were described in Section 3.3.4.

7.3.4 Results

We provide a set of results, where we study the influence of DNN autoencoder signal enhancement on a variety of systems. Our autoencoder approach is also compared to the multi-condition training of PLDA, which can also improve the performance in corrupted acoustic environment. We also evaluate this approach with new features and embeddings. At the end, we combine the autoencoder with the multi-condition training, and we find the best performing combination.

We trained autoencoder for signal enhancement simultaneously for denoising and dereverberation, which provides better robustness towards an unknown form of signal corruption, compared to autoencoder trained on noise or reverberation only (as shown in [Novotný et al., 2018a] and in Section 7.2 of this thesis). The autoencoder we used corresponds to variant **N+RR**.

We also created different multi-condition training sets for PLDA (as described in Section 6.1.2). We used exactly the same noises and reverberation for segment corruption as in the autoencoder training, allowing to compare the performance of systems using the autoencoder and systems based on multi-condition training.

Our results are listed in Table 7.2 for the i-vector-based systems, and in Table 7.3 for the x-vector based ones. The results in each table are separated into four main blocks based on a combination of features and signal augmentation: i) system trained with MFCC without signal enhancement, ii) system trained with MFCC with signal enhancement, iii) system trained with SBN-MFCC without enhancement, iv) and system trained with SBN-MFCC and signal enhancement. In each block, the first column corresponds to the system where PLDA was trained only on clean data. The next three columns represent results with different multi-condition training: N, RR or N+RR (as described in Section 6.1.2).

Finally, the rows of the table are also divided into blocks based on the type of the benchmark, representing telephone channel, microphone and artificially created benchmarks. The last row denoted as *avg* gives the average EER over all benchmarks and each value set in bold is the minimum EER in the particular benchmark.

Table 7.2: Results (EER [%]) obtained in four scenarios. Each block corresponds to an **i-vector** system trained with either MFCC or SBN-MFCC features and with or without signal enhancement applied during i-vector extraction. Blocks are divided into columns corresponding to systems trained in multi-condition fashion (with noised and reverberated data in PLDA). Each column corresponds to a different PLDA multi-condition training set: "—" - clean condition, N - noise, RR - real reverberation, RR+N - real reverberation + noise. The last row denoted as *avg* gives the average EER over all benchmarks and each value set in bold is the minimum EER in the particular benchmark.

	MFCC ORIG				MFCC DENOISED				SBN-MFCC ORIG				SBN-MFCC DENOISED			
Benchmark	—	N	RR	RR+N	—	N	RR	RR+N	—	N	RR	RR+N	—	N	RR	RR+N
tel-tel	1.99	2.39	1.99	2.74	2.06	2.48	2.01	2.09	0.94	1.04	0.93	0.93	0.96	0.97	0.94	**0.91**
sre16-tgl-f	21.85	21.37	21.84	21.88	23.38	22.96	23.25	23.14	21.88	**21.24**	21.82	21.93	22.62	21.93	22.60	22.70
sre16-yue-f	11.20	**10.52**	11.15	11.53	11.76	11.47	11.76	11.79	13.45	13.02	13.45	13.44	14.60	13.69	14.54	14.52
int-int	4.57	4.70	4.49	4.55	4.34	4.59	4.21	4.00	3.88	4.07	3.77	3.73	3.44	3.69	**3.40**	**3.40**
int-mic	1.85	2.09	1.86	2.00	2.51	2.33	2.40	2.32	1.85	**1.69**	1.76	1.78	1.87	1.79	1.84	1.77
prism,chn	1.03	1.29	0.99	0.97	0.59	0.67	0.59	0.57	0.40	0.46	0.39	**0.36**	0.66	0.78	0.72	0.66
sitw-core-core	10.11	10.13	10.06	10.32	9.41	9.60	9.45	9.45	8.09	7.85	8.02	8.03	7.71	**7.59**	7.70	7.70
prism,noi	3.72	3.02	3.65	3.42	2.51	2.38	2.46	2.38	2.43	1.98	2.45	2.20	1.84	**1.73**	1.81	1.76
prism,rev	2.51	2.67	2.40	2.23	1.94	2.09	1.89	1.92	1.42	1.39	1.30	1.31	1.12	1.23	**1.07**	1.09
BUT-RET-orig	2.29	2.56	2.30	2.33	2.19	2.48	2.20	2.19	1.45	1.58	1.47	**1.43**	1.82	1.78	1.81	1.80
BUT-RET-merge	14.43	14.33	13.79	11.22	11.73	11.51	10.83	10.88	15.27	15.00	15.10	13.32	9.97	10.72	**9.38**	9.47
avg	6.87	6.82	6.77	6.65	6.58	6.60	6.46	6.43	6.46	6.30	6.41	6.22	6.06	5.99	**5.98**	**5.98**

i-vector Systems Results

We have already partially performed this analysis in the previous section, due to the added features and for the sake of completeness, we repeat some results here again.

Let us begin with comparing systems with and without signal enhancement. In this case, we focus on PLDA trained on clean data only. In the first case, the i-vector system was trained using the MFCC features. We see mixed results. In the first set of benchmarks representing a telephone channel, we see degradation. When we consider that this is a reasonably clean benchmark, this enhancement was expected not to be very effective.

In the second block of results (interview speech), the situation is better, except *int-mic* benchmark. We can notice an improvement in the system with signal enhancement. An interesting result can be spotted in benchmark *prism,chn*, where, with signal enhancement, we obtain more than 40 % relative improvement.

The next block of artificially corrupted benchmarks from PRISM also reports improvements and the last set of results with our retransmitted data too, in addition we can see that there is no degradation in original benchmark *BUT-RET-orig*.

Let us now focus on the i-vector system based on the SBN-MFCC features. In the past, the SBN-MFCC features provided good robustness in noisy benchmarks. We verify this statement comparing columns *MFFC-ORIG* and *SBN-MFCC-ORIG* in Table 7.2 (systems without signal enhancement). We see that except for the *SRE 2016* and *BUT-RET-merge* benchmarks, the system trained with stacked bottle-neck features yields better performance compared to the original MFCC system. When comparing systems with and without signal enhancement, the situation is similar to the MFCC case. We see degradation on the telephone channels and a portion of the interview speech benchmarks. We obtain 30 % relative improvement in *BUT-RET-merge* where the system without enhancement is even worse than the previous i-vector system. This could indicate that the bottle-neck features provide better robustness to noise than to reverberation.

Table 7.3: Results (EER [%]) obtained in four scenarios. Each block corresponds to an **x-vector** system trained with different type of features with or without signal enhancement. Blocks are divided into columns corresponding to systems trained in multi-condition fashion (with noised and reverberated data in PLDA). Each column corresponds to different PLDA multi-condition training set: "—" - clean condition, N - noise, RR - real reverberation, RR+N - real reverberation + noise. The last row denoted as *avg* gives the average EER over all benchmarks and each value set in bold is the minimum EER in the particular benchmark.

	MFCC ORIG				MFCC DENOISED				SBN-MFCC ORIG				SBN-MFCC DENOISED			
Benchmark	—	N	RR	RR+N	—	N	RR	RR+N	—	N	RR	RR+N	—	N	RR	RR+N
tel-tel	1.30	1.43	1.27	1.29	1.21	1.44	**1.18**	1.20	1.30	1.49	1.29	1.27	1.35	1.45	1.30	1.30
sre16-tgl-f	22.73	22.52	22.87	22.56	21.52	21.41	21.29	21.31	22.33	21.21	22.15	22.33	21.17	**20.74**	20.88	20.95
sre16-yue-f	10.36	9.61	10.45	10.61	8.86	**8.23**	8.75	8.66	9.60	8.71	9.56	9.88	8.89	8.38	8.67	8.64
int-int	3.36	3.72	3.29	3.22	2.92	3.34	2.90	**2.86**	3.24	3.66	3.16	3.20	3.16	3.42	3.08	2.97
int-mic	1.33	1.43	1.3	1.22	1.47	1.37	1.41	1.37	1.07	1.17	1.04	**1.03**	1.37	1.39	1.29	1.27
prism,chn	0.62	0.81	0.61	0.61	0.37	**0.27**	0.36	0.41	0.69	0.67	0.67	0.59	0.36	0.41	0.36	0.36
sitw-core-core	7.87	7.30	7.72	7.41	6.81	6.54	6.73	6.70	7.57	7.42	7.57	7.38	6.81	**6.40**	6.71	6.66
prism,noi	2.76	1.90	2.63	2.11	1.84	**1.50**	1.80	1.73	2.72	1.97	2.63	2.22	1.84	1.57	1.81	1.68
prism,rev	2.08	2.02	1.79	1.60	1.16	1.13	**1.10**	1.12	1.98	2.06	1.71	1.59	1.24	1.27	1.15	1.15
BUT-RET-orig	1.73	1.73	1.69	1.63	1.81	1.82	1.72	1.74	1.50	1.63	1.46	**1.43**	1.77	1.86	1.74	1.75
BUT-RET-merge	15.48	13.94	13.96	13.12	11.83	12.81	**10.07**	10.46	17.20	14.09	15.90	13.74	13.26	12.70	11.03	10.12
avg	6.33	6.04	6.14	5.94	5.44	5.44	5.21	5.23	6.29	5.83	6.10	5.88	5.57	5.42	5.27	**5.17**

x-vector Systems Results

We evaluated our speech enhancement autoencoder also with the system based on x-vectors, which is currently considered as state-of-the-art. In our experiments and system design, we have deviated from the original Kaldi recipe [Snyder et al., 2018]. For training the x-vector extractor, we extended the number of speakers and we also created more variants of augmented data. We extended the original data augmentation recipe by adding real room impulse responses and an additional set of stationary noises (the extension process is also described in [Novotný et al., 2018b], the x-vector network used here is labeled as Aug III. in the paper). In the PLDA backend training, we also added the augmented data for multi-condition training (see Section 6.1.2).

Let us point out, that the denoising autoencoder was trained on a subset of augmented data for training the x-vector DNN. The set of noises and real room impulse responses are therefore the same as in our extended set for training the x-vector extractor (as described in Section 3.3) and there is no advantage in autoencoder possibly seeing additional augmentations. It is also useful to refer the interested reader to our analysis in [Novotný et al., 2018b], where we show the benefit of having such a large augmentation set for x-vector extractor training.

Let us first compare the x-vector network trained with original MFCC and with SBN-MFCC features. In systems based on i-vectors, bottle-neck features mostly provided very significant improvement, but for x-vector-based systems, the gains are much lower, the performance stays the same, or even degrades for benchmark *BUR-RET-merge*. This degradation, however, completely disappears after using denoising in x-vector training and subsequently multi-condition training in PLDA. For the telephone data with low reverberation, we can observe either steady performance on *tel-tel* or slightly better performance on more challenging and non-English data in SRE'16 benchmarks. This is in contrast with i-vectors, where we only see either steady performance on easy *tel-tel* or degradation on more challenging SRE'16. In general, the positive effect of SBN-MFCC features on x-vector system is small, but more stable than in i-vector system.

When we focus on the effect of signal enhancement in the x-vector-based system, we see much higher improvement compared to i-vectors. There are still several cases where the enhancement causes mostly degradation (MFCC: *int-mic*, *BUT-RET-orig*; SBN-MFCC: *tel-tel*, *int-mic*, *BUT-RET-orig*—mostly clean benchmarks). Otherwise, the enhancement provides nice improvement across rest of the benchmarks and features used for system training. At this point, it is useful to point out that unlike with i-vectors, where denoising is applied only for i-vector extraction, we actually apply enhancement already on top of x-vector training data. The effect of applying enhancement only during x-vector extraction like with i-vectors can be seen in Table 7.4. We can observe that also here, we gain some improvements, but they are generally smaller than with enhancement deployed already during x-vector training (Table 7.3).

X-vector systems generally provide greater robustness across different signal corruptions. It was natural for us to expect, that x-vector systems should not need signal enhancement, and that they would implicitly learn it themselves, especially in the first part of DNN described in Section 7.3.3. To our belief, a reason why enhancement helped in our case is that denoising is not the target task of the x-vector DNN. Even though we did have multiple corrupted samples per speaker in the DNN training set, it may be possible that we simply didn't have enough. And since the x-vector training is generally known to be data-hungry,

Table 7.4: Results (EER [%]) of SV system with x-vector extractor trained on clean data and with signal enhancement used only for x-vector extraction. Blocks are divided into columns corresponding to systems trained in multi-condition fashion (with noised and reverberated data in PLDA). Each column corresponds to a different PLDA multi-condition training set: "—" - clean condition, N - noise, RR - real reverberation, RR+N - real reverberation + noise. The last row denoted as *avg* gives the average EER over all benchmarks and each value set in bold is the minimum EER in the particular benchmark.

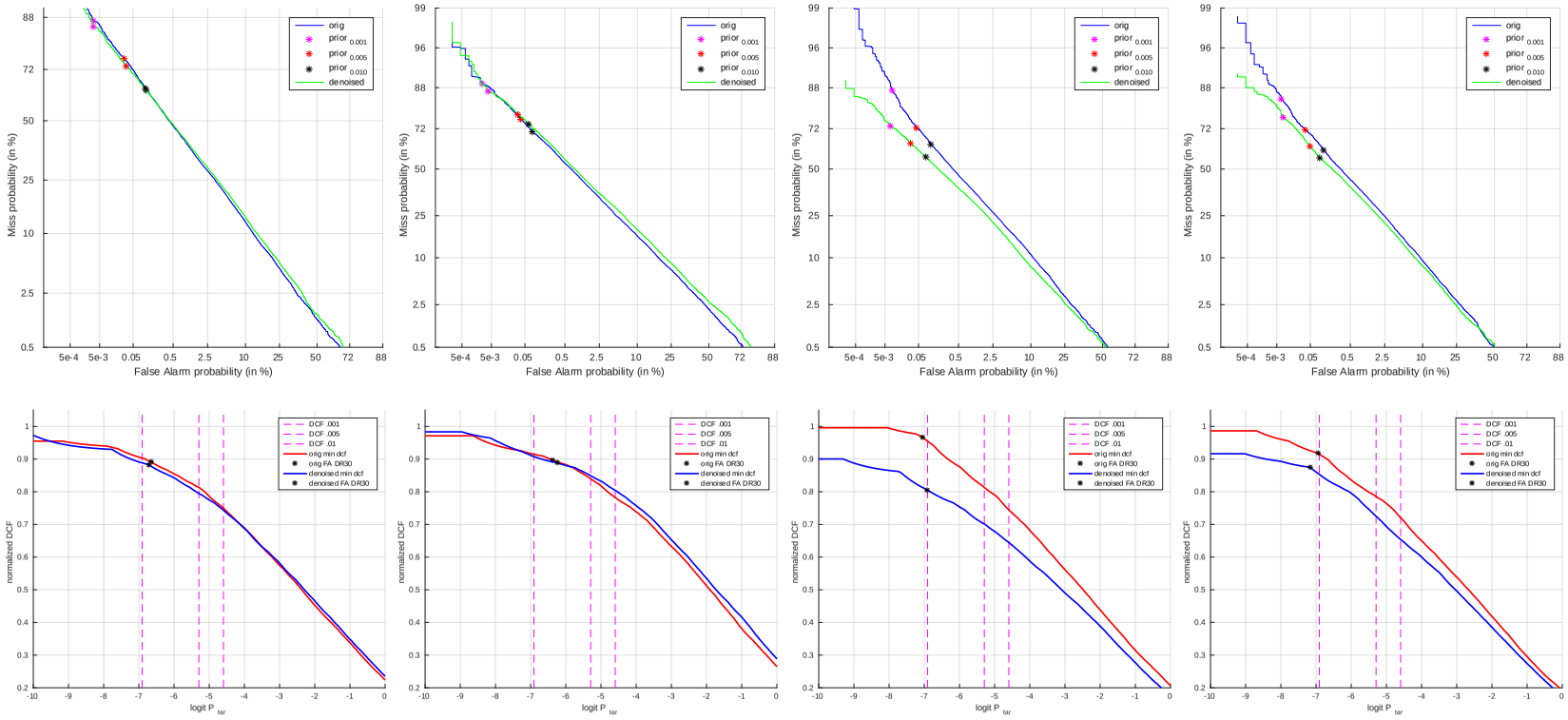
	MFCC				SBN-MFCC			
Benchmark	—	N	RR	RR+N	—	N	RR	RR+N
tel-tel	1.38	1.51	1.34	1.39	1.27	1.40	**1.21**	1.25
sre16-tgl-f	21.12	21.48	21.08	**20.94**	21.73	21.46	21.5	21.63
sre16-yue-f	9.76	**9.01**	9.7	9.69	9.38	9.07	9.41	9.16
int-int	3.15	3.32	3.12	**2.99**	3.19	3.40	3.14	3.05
int-mic	1.61	1.67	1.59	1.58	1.63	1.58	1.51	**1.39**
prism,chn	0.54	0.47	0.55	0.54	**0.40**	0.41	**0.40**	**0.40**
sitw-core-core	7.22	6.76	7.17	6.84	6.96	**6.52**	6.97	6.76
prism,noi	2.14	**1.64**	2.15	2.05	2.33	1.67	2.36	2.15
prism,rev	1.24	1.22	1.18	**1.20**	1.33	1.45	1.28	1.24
BUT-RET-orig	**1.87**	2.07	1.90	1.88	2.09	2.03	2.08	2.07
BUT-RET-merge	12.76	11.76	**10.71**	11.83	15.08	14.32	12.62	12.66
avg	5.71	5.54	**5.50**	5.54	5.94	5.76	5.68	5.61

it is therefore likely that if we had more corrupted samples per speaker, it would be in the DNN's natural capabilities to learn the task of de-noising.

Let us also point out that if a single type of noise (or channel in general) appears systematically with a concrete speaker, the noise becomes a part of the speaker identity and therefore the NN does not compensate for it.

So far, we have compared results on systems, where PLDA was trained on clean data only and we study possible improvements of enhancement across several systems. Multi-condition training of PLDA, where we add a portion of augmented data into PLDA training is another possible approach on how to improve system performance and its robustness.

From the results, we can see that multi-condition training, can provide improvement across all benchmarks and systems without signal enhancement. We can see that the ideal combination of the augmented data for multi-condition training of PLDA depends on a benchmark. In noisy benchmarks (*prism,noi*), it is more effective to use noise augmentation only. For reverberated benchmarks (*prism,rev*, *BUT-RET-merge*) we can see more benefits in using reverberated augmentation set compared to others.



(a) I-vector based systems: left column – MFCC features, right column – SBN-MFCC features.

(b) X-vector based systems: left column – MFCC features, right column – SBN-MFCC features.

Figure 7.3: Detection Error Trade-off (DET) plots (top row) and minDFC as a function of effective prior (bottom row) of all tested scenarios for *sre16-yue-f* benchmark. Intersection of minDCF curves with vertical dashed violet lines correspond from the left to the minDCF from NIST SRE 2010 and to the two operating points of DCF from NIST SRE2016. Similarly, the violet star in the DET plots corresponds to the minDCF from NIST SRE2010 and red and black stars correspond to the two operating points of NIST SRE 2016.

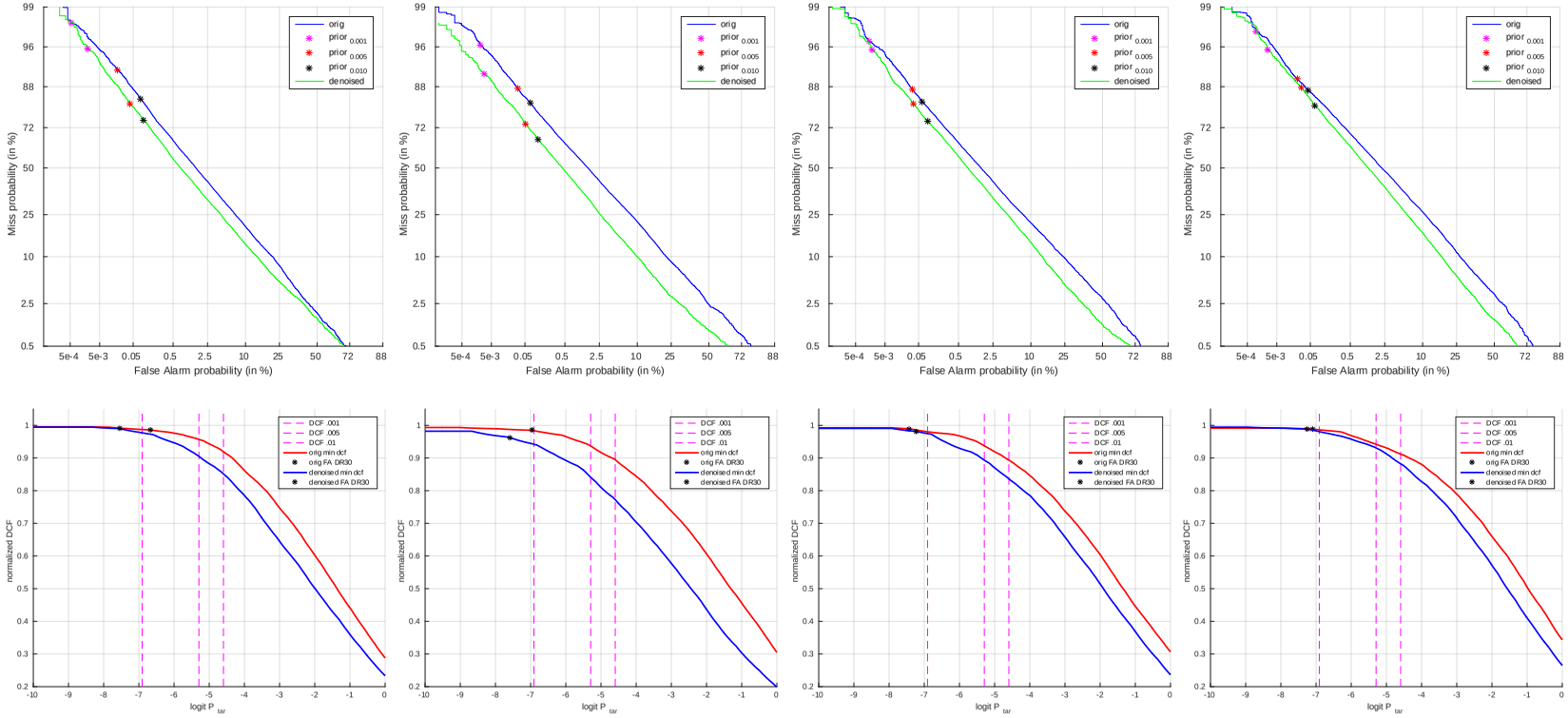
7.3.5 Analysis over the Range of Operating Points

Although EER (described in Section 3.1.2) is a common metric summarizing performance, it does not cover all operating points. In this section, we present the performance of various systems via DET (Section 3.1.1) and DCF (Section 3.1.3) curves as to see a more complete picture of systems' behavioral.

In order to summarize our observation without overwhelming the reader with too many plots, we have chosen two representative benchmarks, that are closest to the real-world scenario—*sre16-yue-f* (Figure 7.3) and *BUT-RET-merge* (see Figure 7.4). More specifically, the *sre16-yue-f* benchmark was chosen because a) it contains original noisy audio, and b) compared to the rest of the conditions, there is a high channel mismatch between the training and the evaluation data. The *BUT-RET-merge* benchmark was chosen because it realistically reflects real reverberation.

The graphs reveal that the benefit from using the studied techniques can be substantial. It is worth noting that according to the tables above, denoising may not be effective w.r.t. EER, however, when looking at the DET curves, we see that there are operating points that do benefit from denoising in a fairly large extent.

Apart from i-vector system on the *sre16-yue-f* benchmark, the DET or DCF curves corresponding to the denoised system are generally better than those using the original noisy data over the whole range of operating points.



(a) I-vector based systems: left column – MFCC features, right column – SBN-MFCC features.

(b) X-vector based systems: left column – MFCC features, right column – SBN-MFCC features.

Figure 7.4: Detection Error Trade-off (DET) plots (top row) and minDFC as a function of effective prior (bottom row) of all tested scenarios for *BUT-RET-merge* benchmark. Intersection of minDCF curves with vertical dashed violet lines correspond from the left to the minDCF from NIST SRE 2010 and to the two operating points of DCF from NIST SRE2016. Similarly the violet star in the DET plots corresponds to the minDCF from NIST SRE2010 and red and black stars correspond to the two operating points of NIST SRE 2016.

Chapter 8

Discriminative Techniques in Generative SV

In previous chapters, we mentioned SBN features, which led to increased robustness and better performance of the i-vector system. In this chapter, we take a closer look at these features. We also look at other techniques (such as DNN UBM alignment) that have helped to improve the original fully generative i-vector paradigm. These techniques can also be seen as developments that have led to the transition to fully discriminative embeddings x-vectors.

8.1 Stack Bottle-neck Features

Bottleneck Neural-Network (BN-NN) refers to such a topology of a NN, where one of the hidden layers has significantly lower dimensionality than the surrounding ones. A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through the BN-NN and reading off the vector of values at the bottleneck layer. We have used a cascade of two such NNs for our experiments. The output of the first network is *stacked* in time, defining context-dependent input features for the second NN, hence the term Stacked Bottleneck features (SBN, [Grézl et al., 2007], Figure 8.1).

The NN input features are 24 log Mel-scale filter bank outputs augmented with fundamental frequency features from 4 different f_0 estimators (Kaldi, Snack[1], and other two according to [Laskowski and Edlund, 2010] and [Talkin, 1995]). Together, we have 13 f_0 related features, see [Karafiát et al., 2014] for more details. Conversation-side based mean subtraction is applied on the whole feature vector, then 11 frames of log filter bank outputs and fundamental frequency features are stacked. Hamming window and DCT projection (0^{th} to 5^{th} DCT base) are applied on the time trajectory of each parameter resulting in $(24 + 13) \times 6 = 222$ coefficients on the first stage NN input.

The configuration of the first NN is $222 \times D_H \times D_H \times D_{BN} \times D_H \times K$, where $K = 9824$ is the number of target triphones. The dimensionality of the bottleneck layer, D_{BN} was set to 30. The dimensionality of other hidden layers D_H was set to 1500. The bottleneck outputs from the first NN are sampled at times $t-10$, $t-5$, t, $t+5$ and $t+10$, where t is the index of the current frame. The resulting 150-dimensional features are inputs to the second stage NN with the same topology as the first stage. The network was trained on the Fisher English corpus, and data were augmented with two noisy copies.

[1] http://kaldi.sourceforge.net, http://www.speech.kth.se/snack/

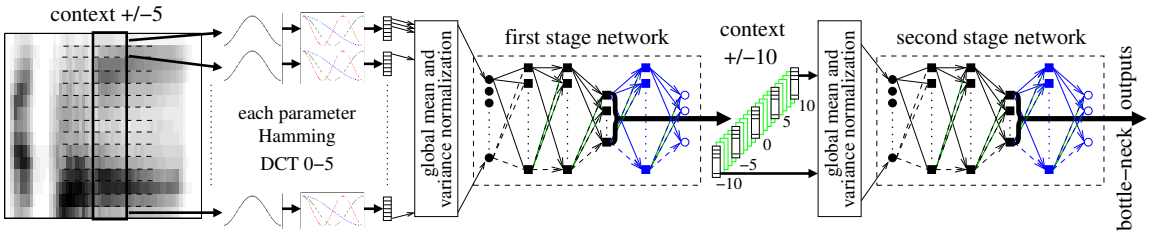


Figure 8.1: Block diagram of Stacked Bottle-Neck (SBN) feature extraction. The blue parts of neural networks are used only during the training. The green frames in context gathering between the two stages are skipped. Only frames with shift -10, -5, 0, 5, 10 form the input to the second stage NN. (Source: [Karafiát et al., 2014])

Finally, the 30-dimensional bottleneck outputs from the second NN (referred to as SBN), are used, usually concatenated with MFCC features.

8.2 DNN Aligment

GMM is a traditional part of the i-vector framework. The great advantage of GMMs is that no data annotations are needed and the training is fully unsupervised. The limitation of the GMM can be caused by a sub-optimal choice of the number of components (usually, 1024, 2048, or 4096 components). The number of components is chosen mostly regarding the available amount of training data, which is typically much lower than what is needed for training the discriminative systems.

The GMM can be seen as a model of acoustic units (or clusters) in the feature space. We intentionally talk about acoustic units in general, because the division of features into individual components of GMM is not optimized for any classification such as phonemes or senones. The generative SV system is therefore not forced to learn this information when trying to compare two speakers.

In the case of DNN-alignment [Hinton et al., 2012, Dahl et al., 2012], features are not grouped without supervision on the basis of position in feature space but based on their content and on the information they carry. DNN-alignment clusters features based on speech content in acoustic units such as senones. This can result in a partition of GMM components, as shown in Figure 8.2. In this example, Senone A is represented by several components, but at the same time, one component represents two senones.

The SBN (ASR NN, mentioned in Section 8.1) is most often used to obtain posterior probabilities used as component occupation $\gamma_t^{(c)}$.

This operation results in entering an a-priori information about what was said in a given recording into a text-independent speaker embedding (i-vector). Therefore, this can be considered a paradigm shift, in which the speakers are not compared blindly on the basis of voice, but the system is forced to compare them on the basis of the speech content as well.

Note the following possibility of DNN-alignment compared to GMM model. In GMM, the feature vectors $\mathbf{o}_t$ are expected to be generated by the GMM model with occupation probabilities $\gamma_t^{(c)}$ (as described in Section 4.1.1):

$$\mathbf{o}_t \sim \sum_{c=1}^{C} \gamma_t^{(c)} \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{8.1}$$

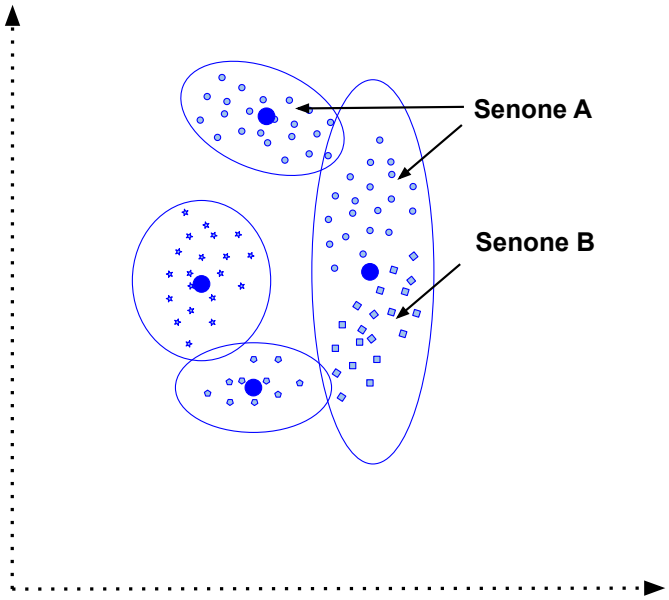


Figure 8.2: Example of GMM, where several components represent Senone A, but at the same time, one component represents two senones.

$$\gamma_t^{(c)} = p(c|\mathbf{o}_t). \tag{8.2}$$

Using the DNN-alignment, we can use the different features to determine occupation probability $\gamma_t^{(c)}$ (8.2) and sufficient statistics:

$$N^{(c)} = \sum_t \gamma_t^{(c)}, \tag{8.3}$$

$$\mathbf{f}^{(c)} = \sum_t \gamma_t^{(c)} \mathbf{o}_t, \tag{8.4}$$

$$\mathbf{S}^{(c)} = \sum_t \gamma_t^{(c)} \mathbf{o}_t \mathbf{o}_t^T. \tag{8.5}$$

8.2.1 Experimental Setup

In this experiment, we focus on the comparison of both mentioned approaches (using SBN features and DNN-alignment) and their influence on the robustness of SV with respect to language variability (based on [Novotný et al., 2016]).

We experimented with monolingual (English and Mandarin) and multilingual BN features. In the case of multilingual training, we adopted a training scheme with block-softmax, which divided the output layer into parts according to individual languages. During training, the only part of the output layer that corresponds to the language the given target belongs to is activated. See [Veselý et al., 2012, Fér et al., 2015] for detailed description.

Evaluation Set

We report our results on the "Language Set" pack of the PRISM set, referred to as *prism,lan*. Moreover, results on the Chinese subset of the *prism,lan* benchmark, referred

to as *prism,chin* are reported. To provide a contrastive view, we also report the results on the NIST SRE 2010 data extended core benchmark (telephone-telephone, "condition-5"), referred to as *tel-tel*, which is English.

8.2.2 System Definition

For training the Multilingual neural networks, the IARPA Babel Program data[2] were mainly used. This data set simulates the scenario of what one could collect in a limited time from a completely new language. It consists mainly of conversational telephone speech (CTS), but scripted recordings, as well as far field recordings, are present. We used 11 languages to train our multilingual SBN feature extractor. The *language list* (as referred to later in this paragraph) consists of Cantonese, Assamese, Bengali, Pashtu, Turkish, Tagalog, Vietnamese, Haiti, Lao, Tamil, and Zulu. More details about the characteristics of the languages can be found in [Harper, 2013]. The phone-state target labels were obtained using forced-alignment with our BABEL ASR system [Karafiát et al., 2013], with $471 + 141 + 147 + 216 + 126 + 252 + 303 + 99 + 411 + 102 + 219 = 2487$ phone states, respectfully to the *language list*.

For the monolingual English DNN variant, we have used a selection of 250 hours of data derived from the Fisher English Part 1 and 2 with 2423 tied tri-phone states.

For the monolingual Mandaring DNN, we have used total of 153 hours from the Mandarin HKUST, and the Mandarin CallHome/CallFriend collections [Karafiát et al., 2016], with 4941 tied tri-phone states.

As the baseline features, we used 19 MFCC coefficients + energy augmented with their delta and double delta coefficients, resulting in 60-dimensional feature vectors. The analysis window was 20 ms long with the shift of 10 ms. First, we removed silence frames according to our standard VAD (described in Section 1.3), after which we applied short-time (300 frames) cepstral mean and variance normalization.

The PRISM set was chosen as the training dataset. A gender-independent UBM was represented as a full or diagonal covariance 2048-component GMM. It was trained on a subset of the PRISM training set: 15602 files equally distributed between telephone and microphone benchmarks and male and female portions. The variance flooring was used in each iteration of the EM algorithm during the UBM training. Gender-independent i-vector extractor was trained using the entire PRISM set. The results are reported with 600-dimensional i-vectors. Gender-independent LDA and PLDA were trained on the same data as the i-vector extractor.

8.2.3 Results

Table 8.1 shows the overall results of all systems in terms of (calibration insensitive) $\mathrm{DCF}^{\mathrm{min}}_{\mathrm{old}}$, $\mathrm{DCF}^{\mathrm{min}}_{\mathrm{new}}$, and EER. For the *tel-tel* test, the best performing system is the DNN-alignment with the DNN trained on the Fisher English data, as expected. However, when looking at the *prism,lan* benchmark, there is no gain from switching from the Baseline system to English DNN (and only a negligible gain in switching to English SBN).

Our hypothesis was that this behavior would be fixed by using a more general DNN, such as the Multilingual DNN (only in the SBN variant, as explained in Section 8.2), since the test comprises of numerous languages. However, it turned out that Mandarin SBN suited this benchmark the best.

[2] Collected by Appen, http://www.appenbutlerhill.com

Looking at the *prism,chin* benchmark, we again expected the Mandarin DNN (or SBN) to significantly outperform the English and Multilang DNN's, which turned out not to be the case.

Our initial hypothesis was that the English training corpus is the largest, and therefore had to provide the best phone accuracy and thus a better acoustic space clustering. However, it was observed in many cases (e.g. in [Lozano-Diez et al., 2016]) that better phone accuracy does not necessarily imply better SV performance. Therefore, we leave this question open for future research.

Let us also note that the UBM/i-vector/PLDA training data are identical—i.e., mainly English—across the different systems. Our hypothesis is that even if the DNN matches the target language, the acoustic space clustering does not fit well to the observed data. Therefore, the first-order statistics (4.11) for the i-vector extractor computation are "warped", and the i-vector extractor captures a different "total" variability than is in fact used for the test. One of the possible indications for this hypothesis is the fact that the performance on the *tel-tel* benchmark does not vary dramatically across different systems. Similar hypothesis holds for the PLDA/LDA modeling, where the within/across variabilities are modeled using these "warped" i-vectors.

Table 8.3 shows the overall performance in terms of the actual vs. the minimum DCF values, i.e., it directly shows the calibration loss. We see that the *tel-tel* benchmark is well calibrated, i.e., the actual values are close enough to the minimum counterparts. However, looking at the *prism,chin* and *prism,lan* tests, and especially at the $\mathrm{DCF}_{\mathrm{new}}$ metric, the calibration losses are extremely high. This effect is even more pronounced for the female part of the tests.

In Table 8.2, we show the effect of a linear calibration on the English SBN system. Because of the lack of an independent held-out set, we performed a cheating (gender-independent) calibration trained using the *prism,lan* trial set, which contains both English and Chinese trials.

We see that although not perfect, the $\mathrm{DCF}_{\mathrm{new}}$ of the *prism,lan* and *prism,chin* remained almost constant across tasks, especially in the female case (which could be explained by having twice as many female trials compared to the male portion). It seems that even though English trials were in majority in the *prism,lan* set, the calibration still helped the non-English trials. The *prism,chin* calibration loss reduction was the most noticeable. The *tel-tel* benchmark got de-calibrated, as expected. All this behavior indicates a heavy language-dependent score modality.

Table 8.1: Comparison of the systems under the PRISM *prism,lan* and *prism,chin*, and the NIST *tel-tel* benchmarks. We expected (without result) the Multilang SBN to perform best in the *prism,lan* benchmark, and a variant of Mandarin to perform best in the *prism,chin* benchmark.

Test set	System	DCF^{min}_{new}		DCF^{min}_{old}		EER [%]	
		male	female	male	female	male	female
prism,chin	Baseline	0.1834	0.3019	0.0621	0.0894	1.44	2.27
	English SBN	0.1491	0.2251	**0.0418**	0.0838	**1.00**	1.99
	Mandarin SBN	0.1480	0.2368	0.0511	0.0755	1.45	2.47
	Multilang SBN	0.2121	**0.1907**	0.0439	**0.0670**	1.16	**1.93**
	English DNN	**0.1373**	0.3621	0.0616	0.1192	1.29	3.05
	Mandarin DNN	0.1688	0.2574	0.0516	0.1018	1.17	2.70
prism,lan	Baseline	0.2979	0.9836	0.1021	**0.2007**	2.60	5.05
	English SBN	0.2963	0.9848	0.0979	0.2305	2.45	4.93
	Mandarin SBN	**0.2734**	0.9787	**0.0685**	0.2282	**1.69**	**4.11**
	Multilang SBN	0.4008	0.9854	0.0898	0.2997	2.16	5.03
	English DNN	0.2963	0.9463	0.0914	0.2228	2.70	5.68
	Mandarin DNN	0.3705	**0.9234**	0.1450	0.3255	3.57	7.14
tel-tel	Baseline	0.3577	0.3387	0.0967	0.1013	1.84	1.94
	English SBN	0.1295	**0.1679**	0.0387	0.0471	1.17	1.11
	Mandarin SBN	0.1459	0.2087	0.0440	0.0604	1.20	1.11
	Multilang SBN	0.1280	0.1696	0.0416	0.0544	1.21	1.16
	English DNN	**0.1200**	0.2212	**0.0352**	**0.0449**	**0.71**	**0.93**
	Mandarin DNN	0.2732	0.3356	0.0702	0.0856	1.60	1.83

Table 8.2: The effect of calibration on the actual DCF's under the PRISM *prism,lan* and *prism,chin*, and the NIST *tel-tel* tests for the English SBN system.

Test	System	DCF^{act}_{new}		DCF^{act}_{old}	
		male	female	male	female
prism,chin	Uncal	1.5201	10.4024	0.0515	0.1857
	Cal	0.5278	0.5080	0.0642	0.0859
prism,lan	Uncal	2.1503	24.4566	0.0702	0.3476
	Cal	0.5519	1.2460	0.0950	0.2311
tel-tel	Uncal	0.1472	0.1750	0.0976	0.1098
	Cal	0.8349	0.8604	0.2087	0.2487

Table 8.3: Analysis of the actual DCF's under the PRISM *prism,lan* and *prism,chin*, and the NIST *tel-tel* benchmark. Note the system de-calibration on the *prism,lan* and *prism,chin* benchmarks. Also note that de-calibration is more emphasized for the female benchmarks. (Due to the dynamic range of the values, we prefer to report a table of numbers rather than a graph plot.)

Test	System	DCF_{new}				DCF_{old}			
		actual		min		actual		min	
		male	female	male	female	male	female	male	female
prism,chin	Baseline	5.7461	16.0798	0.1834	0.3019	0.1206	0.2785	0.0621	0.0894
	English SBN	1.5201	10.4024	0.1491	0.2251	0.0515	0.1857	0.0418	0.0838
	Mandarin SBN	8.4710	25.2394	0.1480	0.2368	0.1536	0.4003	0.0511	0.0755
	Multilang SBN	3.9156	12.3843	0.2121	0.1907	0.0863	0.2189	0.0439	0.0670
	English DNN	10.2419	46.4058	0.1373	0.3621	0.1856	0.6857	0.0616	0.1192
	Mandarin DNN	30.4309	75.9809	0.1688	0.2574	0.4683	0.9842	0.0516	0.1018
prism,lan	Baseline	3.5369	14.0482	0.2979	0.9836	0.1142	0.2812	0.1021	0.2007
	English SBN	2.1503	24.4566	0.2963	0.9848	0.0702	0.3476	0.0979	0.2305
	Mandarin SBN	5.8890	30.2647	0.2734	0.9787	0.1333	0.4363	0.0685	0.2282
	Multilang SBN	5.2089	38.1320	0.4008	0.9854	0.1121	0.4855	0.0898	0.2997
	English DNN	6.6261	36.8887	0.2963	0.9463	0.1427	0.5451	0.0914	0.2228
	Mandarin DNN	16.0119	58.9831	0.3705	0.9234	0.2856	0.7746	0.1450	0.3255
tel-tel	Baseline	0.4323	0.3442	0.3577	0.3387	0.1587	0.2171	0.0967	0.1013
	English SBN	0.1472	0.1750	0.1295	0.1679	0.0976	0.1098	0.0387	0.0471
	Mandarin SBN	0.1815	0.2139	0.1459	0.2087	0.1264	0.1428	0.0440	0.0604
	Multilang SBN	0.1530	0.1921	0.1280	0.1696	0.1171	0.1339	0.0416	0.0544
	English DNN	0.1234	0.2286	0.1200	0.2212	0.0800	0.1204	0.0352	0.0449
	Mandarin DNN	0.3320	0.3539	0.2732	0.3356	0.1231	0.1865	0.0702	0.0856

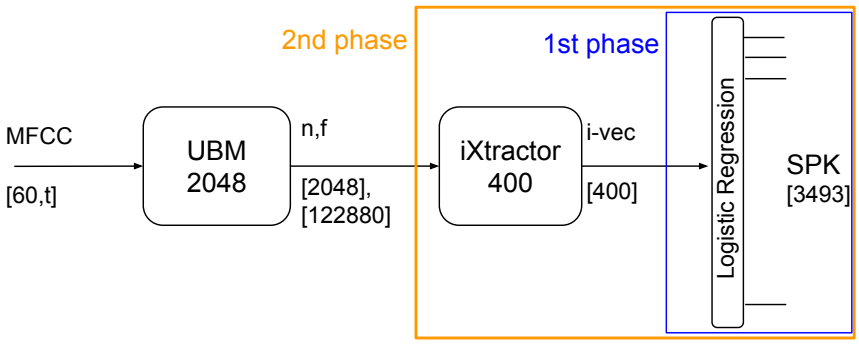


Figure 8.3: Training pipeline of i-vector extractor parameters re-estimation. During the initial phase of training, only the logistic regression is trained. During the second phase, the parameters of the logistic regression and the i-vector extractor ($\mathbf{T}$-matrix) are updated.

8.3 Discriminatively Re-trained i-vector Extractor

In this set of experiments, we keep the large parameter space from the generative i-vector extractor and we focus on discriminative retraining of such a model that still uses a fairly complex GMM-UBM to provide the training examples (sufficient statistics). I-vector model is generally very robust, which is a property that we want to retain, but at the same time we want the model to focus on important features with respect to the task at hand—discrimination between speakers. We do not want the model to waste parameters to represent the redundant variability in the data.

To obtain a standalone discriminative i-vector extractor, we used the same strategy as in the x-vector framework and we retrained the hyper-parameters of the original i-vector model to optimize the multi-class cross-entropy over a set of training speakers (4.55). This is in contrast with our previous research [Glembek et al., 2011a], where we optimized the binary cross-entropy over verification trials formed by pairs of i-vectors. We show that with such an approach we can achieve a signifficant improvement in performance.

8.3.1 T-matrix Re-estimation

Traditionally, matrix $\mathbf{T}$ is trained in a generative fashion using the EM algorithm. In this work, however, we focus on the i-vector extractor parameter re-estimation to better discriminate between speakers. Our experimental pipeline is plotted in Figure 8.3.

Multi-class logistic regression was used as a classifier, where the posterior probability of class (speaker) k given i-vector $\boldsymbol{\phi}_{\mathcal{X}_n}$ (as defined in (4.43)) is computed as:

$$p_{\mathbf{W}}(C_k \mid \boldsymbol{\phi}_{\mathcal{X}_n}) = \frac{\exp(\mathbf{w}_k^T \boldsymbol{\phi}_{\mathcal{X}_n})}{\sum_j \exp(\mathbf{w}_j^T \boldsymbol{\phi}_{\mathcal{X}_n})}, \tag{8.6}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K]$ are the parameters of logistic regression. Multi-class cross-entropy was used as the objective function:

$$E(\mathbf{W}, \mathbf{T}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} s_{nk} \log p_{\mathbf{W}}(C_k \mid \boldsymbol{\phi}_{\mathcal{X}_n}), \tag{8.7}$$

where, s_{nk} is k-th element of the target variable in 1-of-K coding, K is number of speakers (classes), and N is number of training samples. For the purpose of this work, let us treat the i-vector $\boldsymbol{\phi}_{\mathcal{X}_n}$ as a function of $\mathbf{T}$.

Table 8.4: Comparison of the different approaches of classifier initialization presented on multi-class cross-entropy loss ($E(\mathbf{W})$) and EER of selected benchmarks.

		EER [%]	
Init	$E(\mathbf{W})$	tel-tel	sitw-core-core
RAND	0.0821	1.97	9.89
$\mathbf{\Sigma}_{wc}^{\text{PLDA}}$	0.2516	2.17	10.38
COS	0.9470	3.56	11.70

Model Initialization

The proposed technique does not aim to be a replacement for the EM training algorithm. The aim is to push the resolution of i-vectors more towards the SV task, having speakers as classes in mind. For this purpose, the $\mathbf{T}$ matrix was not randomly initialized, but a $\mathbf{T}$ pre-trained by the classical EM algorithm was used to initialize it $\mathbf{T}_0 = \mathbf{T}_{\text{EM}}$.

Classifier Initialization

In the case of the classifier, there is no straightforward way of initializing its parameters (as defined by (8.6)), as in the case of the $\mathbf{T}$-matrix, but we also need to initialize the classifier for effective model retraining. The simplest method is random initialization. However, it is possible to use two-covariance PLDA (as described in Section 5.3.2), where we use its within-class covariance matrix $\mathbf{\Sigma}_{\text{wc}}$ (5.12) to define the new classifier:

$$\begin{aligned} \mathbf{w}_k &= \mathbf{\Sigma}_{\text{wc}}^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= \frac{1}{2}\boldsymbol{\mu}_k\mathbf{\Sigma}_{\text{wc}}^{-1}\boldsymbol{\mu}_k + \ln p(C_k), \end{aligned} \tag{8.8}$$

where $\boldsymbol{\mu}_k$ is mean value of set of speaker i-vectors $\boldsymbol{\phi}_{\mathcal{X}_k}$ and $p(C_k)$ is probability of speaker k in training set. Another initialization option is to use cosine similarity scoring (described in Section 5.1). The only condition is to ensure the unit length of the i-vectors entering to the classifier:

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i\in k}\boldsymbol{\phi}_i \tag{8.9}$$

$$\boldsymbol{w}_k = \frac{\boldsymbol{\mu}_k}{\|\boldsymbol{\mu}_k\|}. \tag{8.10}$$

Experiments with Classifier Initialization

In the sections above, we suggested three ways to initialize the classifier. Before the analysis of the $\mathbf{T}$ re-training, let us focus on the best initialization selection. For this purpose, we only use the *tel-tel* benchmark (as a representative of clean data) and *sitw-core-core* (as a representative of very noisy data). We propose a two-phased training (see Figure 8.3). In phase-1, we only trained the classifier (parameters $\mathbf{W}$). After several epochs, when the loss stopped improving, we initiated phase-2, in which the trained classifier and $\mathbf{T}$ were jointly trained. A stochastic gradient descent algorithm was used as an optimizer.

In Table 8.4, let us observe the objective loss (multi-class cross-entropy) $E(\mathbf{W})$ at the end of phase-1. We can clearly see that the random initialization dominates in the loss

values. In the table, we also present EER result for *tel-tel* and *sitw-core-core* benchmark at the end of phase-2. Again, we can see that the random initialization of the classifier led to the best improvement after phase-2. For this reason, the random initialization of the classifier is selected for further experiments.

Experiments with T-matrix Re-estimation

We conducted a set of experiments, all with three different approaches in i-vector extractor training, denoted with letters B, G, D further on:

B — In the first set of experiments, we trained a baseline i-vector extractor in the traditional generative way, using the original PRISM training corpus.

G — In the second variant, we still used generative training, but we augmented the training data with noise, reverberation, and cuts as described in Section 3.3.

D — In the last variant, we used the pre-trained generative i-vector extractor from G and we retrained it discriminatively. The training pipeline is shown in Figure 8.3.

For discriminative training, careful data preparation was necessary to avoid classifier overtraining. We used speakers with at least 5 utterances in original data only. This step limits the training data to 3493 speakers with 59112 utterances (177336 utterances including augmentation).

For all experiments, we kept the same PLDA configuration. The i-vectors are pre-processed with mean normalization, LDA (i-vectors are transformed into 200-dimensions) and finally they are length-normalized.

Our results (EER) are presented in Table 8.5, for 400- and 600-dimensional i-vectors. They are also divided based on PLDA, where we distinguish PLDA trained on clean data and multi-condition training. Finally, the table is also divided based on the type of the benchmark, for the telephone channel, microphone and artificially created benchmarks. We did not use any type of adaptation.

When we compare baseline systems (B column in the table) with discriminatively re-trained variant (D column in the table), we can see that except two cases (*sre16-yue-f* in 400-dimensional variant with "clean" PLDA and *int-mic* in 600-dimensional variant with "clean" PLDA) the discriminative training is better. The discriminative approach is also better compared to the generative approach with augmented data in the training, where it can be seen see that augmentation in generative training caused mostly degradation.

In *tel-tel* benchmark, we can see significant improvement with discriminative training, where 400-dimensional system has almost 12 % relative improvement compared to baseline and it also outperforms 600-dimensional baseline. Similarly, in the *prism,ch* benchmark, we have 22 % relative improvement in 400-dimensional variant, here we also outperform 600-dimensional discriminative variant of training.

The general improvement can be observed also when doing multi-condition training of the PLDA, but we can also see that it harmed the clean benchmark and helped more on the noisy one, which is an expected behavior.

Generative i-vector extraction training is unsupervised. When we add augmented data to the training list, i-vector extraction is forced to reserve a portion of parameters for representation of variability of noise, reverberation and so it limits parameters for speaker variability. In our supervised discriminative approach, we are pushing i-vector extractor to do the opposite. The extractor is forced to distinguish the speakers, so it should decrease

the unwanted variability and keep as many parameters of the **T**-matrix to the speaker variability. It can also help to limit usage GMM components which are not useful for speaker separation.

Table 8.5: Comparison of the i-vector baseline with different approaches used for i-vector extractor training. Both blocks are divided into columns corresponding to the dimensionality of i-vectors (400- and 600-dimensions). Results are also divided based on the training set of PLDA, where we used clean and multi-condition fashion (with noised and reverberated data). Results (EER [%]) in each column correspond to the different i-vector extractor training setup: B - generative baseline without augmented data, G- generative training with augmented data and D - augmented data used for discriminative retraining. The last row denoted as *avg* gives the average EER over all benchmarks and each value set in bold is the minimum EER in the particular benchmark.

	400-dim						600-dim					
	PLDA clean			PLDA extension data			PLDA clean			PLDA extension data		
Benchmark	B	G	D	B	G	D	B	G	D	B	G	D
tel-tel	2.23	2.43	1.97	3.36	3.73	3.25	1.99	1.98	**1.84**	2.74	2.86	2.70
sre16-yue-f	10.90	11.25	10.97	11.32	11.20	**10.87**	11.20	11.32	11.10	11.53	11.20	11.17
int-int	4.72	4.75	**4.37**	4.83	4.88	4.56	4.57	4.52	4.47	4.55	4.71	4.52
int-mic	2.15	2.24	2.11	2.02	2.28	1.91	**1.85**	2.11	1.91	2.00	2.02	1.94
prism,chn	1.13	1.48	**0.88**	1.14	1.40	1.14	1.03	0.92	0.95	0.97	1.04	0.94
sitw-core-core	10.51	10.75	10.29	10.57	10.62	10.21	10.11	10.28	**9.82**	10.32	10.34	10.17
prism,noi	4.43	4.51	3.97	3.66	3.90	3.44	3.72	3.79	3.58	3.42	3.26	**3.25**
prism,rev	2.81	3.06	2.54	2.45	2.47	2.34	2.51	2.74	2.35	2.23	2.22	**2.15**
avg	4.86	5.06	4.64	4.92	5.06	4.72	4.62	4.71	**4.50**	4.72	4.71	4.61

Experiment with T-matrix: Additional Observations

We found out, that robust classifier was necessary for proper **T**-matrix retraining. We have conducted experiments with different depth of NN multi-class classifier until we settled on a topology with no hidden layer, which effectively equals to logistic regression. With this setup, we avoid problems with overtraining (especially in the early stage of our endeavor, where we did not use augmented data), there are fewer parameters to train, and time and memory requirements are within reasonable limits, yielding an overall robust classifier.

For effective i-vector extractor re-training, a well-trained classifier was crucial. In stage-2 of the training (where classifier was jointly retrained with the **T**-matrix), a poorly trained classifier resulted in either negligible or even harmful update of the **T**-matrix.

Because of its size, matrix **T** was prone to overtraining, therefore, regularization was necessary. We have chosen L2 regularization centered around the initial ML matrix $\mathbf{T}_{\mathrm{EM}}$. This limits the estimate of **T** from moving too far from the initial (already well-estimated) state.

After several unsuccessful experiments, where the change of **T** was too rapid, we set learning rate during the full pipeline training to 10^{-3} (so far, 10^{-1} was used). After this change, the regularization was not necessary anymore, and we received stable training.

Fixing the parameters of the classifier during stage-2 (and retraining only **T**) led to minor effect on the system, compared to the joint training.

Retraining **T** from randomly initialized matrix rather than from a ML estimate did not lead to convergence.

8.3.2 T-matrix Factorization

We continue with our previous research, in which we kept the large parameter space away from the generative i-vector extractor and in which we focused on discriminative retraining of such a model. We were able to retain the model robustness and even increase the SV performance by optimizing the model for discrimination between speakers—a task closely related to the final speaker verification. However, memory requirements and sizeable computational cost during training have not only limited us in running experiments effectively, but more importantly, they prevent possible usage in a larger DNN scheme that would be closer to an end-to-end system.

To solve our problem, we had to drastically decrease the number of trainable model parameters, but, of course, without a major decrease in performance. In the past, researchers have dealt with the same issue and experimented with factorization of similar or even the same models as ours. In 2003, Subspace Precision and Mean model (SPAM) for acoustic modeling in speech recognition was introduced in [Axelrod et al., 2003] and later optimized by Daniel Povey in [Povey, 2006]. SPAM models are Gaussian mixture models with a subspace constraint, in which each covariance matrix is represented as a weighted sum of globally shared full-rank matrices. In 2014, Sandro Cumani proposed an i-vector extractor factorization [Cumani and Laface, 2014] for faster i-vector extraction and smaller memory footprint. Each row of the i-vector extractor matrix is represented as a linear combination of atoms from a common dictionary with the assumption that it is not necessary to store all rows of this matrix to perform i–vector extraction.

In our approach to factorization, we were inspired by [Cumani and Laface, 2014], but instead of factorizing each row, we perform factorization on the level of submatrices of the i-vector extractor that represent individual GMM-UBM components:

$$\bar{\mathbf{T}}^{(c)} = \sum_{q=1}^{Q} a_q^{(c)} \mathbf{U}_q, \tag{8.11}$$

where Q is number of factors, $\mathbf{U}_q$ are the base matrices, $a_q^{(c)}$ are scalar weights for each component $\mathbf{T}^{(c)}$ (see (4.45)). Note that bases $\mathbf{U}_q$ are shared across all components c. C represents number of all components, F feature dimensionality and D i-vector dimensionality. The number of parameters in this new model representation is $QC + QFD$, while the number parameters in the original i-vector extractor was CFD. Since individual matrices $\mathbf{T}^{(c)}$ in the original i-vector concept are theoretically linearly independent, the size of Q would have to be equal to C in order for the factorized model to fully describe the original subspace $\mathbf{T}$. However, our assumption is that there is, in fact, some level of linear dependency and therefore, Q can be chosen significantly smaller than C, therefore reducing the original model parameter space.

To finally obtain a discriminative i-vector extractor, we still use the same strategy as in Section 8.3.1 and we retrain the NN representation of our factorized generative model to optimize the multi-class cross-entropy (8.7) over a set of training speakers.

Generatively trained i-vector extractor was used as an initialization. The logical step would be to use this initialization here as well. To find a solution with respect to $\mathbf{U}_q$, the derivative in (4.53) can be extended using a chain rule to:

$$\frac{\mathrm{d}Q(\mathbf{T}, \mathbf{T}_0)}{\mathrm{d}\mathbf{U}_q} = \frac{\mathrm{d}Q(\mathbf{T}, \mathbf{T}_0)}{\mathrm{d}\mathbf{T}^{(c)}} \frac{\mathrm{d}\mathbf{T}^{(c)}}{\mathrm{d}\mathbf{U}_q} = 0. \tag{8.12}$$

Training and Initialization of Factorised T-matrix

There is no closed-form solution for $\mathbf{U}_q$ of (8.12). Therefore, we generalize the optimization objective by adding an L_2 regularizer:

$$E_{\text{reg}}(\mathbf{W}, \mathbf{T}) = E(\mathbf{W}, \mathbf{T}) + \lambda ||\mathbf{T}, \mathbf{T}_{\text{EM}}||, \tag{8.13}$$

where $||\mathbf{T}, \mathbf{T}_{\text{EM}}||$ is Euclidian distance between our factorized matrix $\mathbf{T}$, and the original generatively trained matrix $\mathbf{T}_{\text{EM}}$.

We used two training strategies which differ in initialization and in the λ regularizing factor.

In scheme-1 initialization, we select Q eigen-vectors (based on Q largest eigen-values) of covariance matrix of the vectorized $\mathbf{T}^{(c)}$'s (C vectors of FD-dimensionality). Parameters $a_q^{(c)}$ are computed as a solution of system of Q equations:

$$\bar{\mathbf{T}}^{(c)} = \sum_{q=1}^{Q} a_q^{(c)} \mathbf{U}_q. \tag{8.14}$$

For this scheme, we globally set $\lambda = 0$. In scheme-2, we started with random initialization, and for the first epoch (phase-0), λ was set to a large number. The rest of the training is two-stage training as described in Section 8.3.1. In general, we used stochastic gradient descent algorithm for parameter optimization.

Experiments with Factorized T-matrix

One of the issues we had to solve to even begin experimenting with the factorized model was its proper initialization. We present two different strategies for initialization and then we experiment with subsequent discriminative retraining of such models. We also provide comparisons with the generative baseline and with discriminative retraining of its full representation. In our experiments with factorization, we set the number of bases Q to 250. This means that the matrix $\mathbf{T}$ is represented by 7.5 times less parameters compared to the original model $\mathbf{T}_{\text{EM}}$, and when compared to the i-vector extractor block from [Rohdin et al., 2018], the number of parameters is almost half. In all of our experiments, we set the i-vector subspace dimensionality to 400.

For clarity, we denote different ways of obtaining the i-vector extractor by capital letter B, C, R and D:

B — We trained a baseline i-vector extractor in the traditional generative way, using the original PRISM training corpus without any augmentations.

C_0 — We initialized the bases $\mathbf{U}_b$ for factorized model by eigen-vectors.

C — We initialized the bases $\mathbf{U}_b$ for factorized model by eigen-vectors as in C_0 and then we continued training with the loss function from (8.7) and the two-phase training described in Section 8.3.1.

R_0 — We initialized the bases $\mathbf{U}_b$ randomly and then we ran a single epoch of training with the loss function in (8.13).

R — We initialized the bases $\mathbf{U}_b$ randomly and then we ran a single epoch of training with the loss function in (8.13) as in R_0, then we continued training with the loss function from (8.7) and the two-phase training described in Section 8.3.1.

D — We driscriminatively re-trained a full representation of the baseline generative i-vector extractor [Novotný et al., 2019b].

For training, we used augmented PRISM set (based on our recipe in Section 3.3). To avoid over-fitting of the classifier during discriminative training, it was necessary to filter the training data. From the original data, we selected speakers with at least 5 utterances. This step limits the training data to 3493 speakers with 59112 utterances (177336 utterances including augmentation).

For all experiments, we kept the same PLDA configuration. The i-vectors were pre-processed with mean normalization, LDA (i-vectors are transformed into 200-dimensions) and finally, they were length normalized.

Our results in terms of EER are presented in Table 8.6 which is divided into two vertical blocks to provide a comparison between PLDA trained on the clean data and multi-condition PLDA training, where we trained the PLDA also on augmented copies of its training data. Now we are interested in the general robustness of our methods and therefore we focus on overall performance across all conditions rather than looking closely into individual cases.

The table is also divided into three horizontal blocks based on the type of the benchmark: into telephone channel (*tel-tel*, *sre16-yue-f*), microphone (*int-int*, *int-mic*, *prism,ch*, *sitw-core-core*) and artificially created benchmarks (*prism,noi*, *prism,rev*). For systems C and R, we also present results for initialization, before $\mathbf{U}_b$ were retrained (in R after the first epoch with $\lambda||\mathbf{T}, \mathbf{T}_{\text{orig}}||$ penalty).

Table 8.6: Results in terms of EER [%] for different i-vector extractors: B - generative baseline without augmented data, C_0 and R_0 are mere initialized factorized models while C and R are their re-trained variants. D stands for a full representation of the original i-vector extractor that has been discriminatively re-trained. The table is also verticaly divided into two blocks which correspond to the training set of PLDA, where we used either only clean data or multi-condition style of training (with noised and reverberated data added to the training of PLDA). The last row denoted as *avg* gives the average EER over all benchmarks and each value set in bold is the minimum EER in the particular benchmark.

	PLDA clean						PLDA extension data					
Benchmark	B	C_0	C	R_0	R	D	B	C_0	C	R_0	R	D
tel-tel	2.23	8.39	3.90	2.47	2.20	**1.97**	3.36	9.72	4.91	3.52	3.30	3.25
sre16-yue-f	10.90	17.39	12.79	11.29	10.96	10.97	11.32	17.18	12.18	11.42	11.11	**10.87**
int-int	4.72	9.56	5.57	4.74	4.51	**4.37**	4.83	10.18	5.94	4.96	4.67	4.56
int-mic	2.15	5.27	2.69	2.23	2.18	2.11	2.02	5.67	2.65	2.28	2.10	**1.91**
prism,chn	1.13	5.63	2.25	0.92	**0.83**	0.88	1.14	5.95	1.98	1.11	1.12	1.14
sitw-core-core	10.51	17.97	12.35	10.92	10.40	10.29	10.57	17.54	12.33	10.84	10.47	**10.21**
prism,noi	4.34	11.74	6.15	4.60	4.29	3.97	3.66	10.73	5.27	4.04	3.73	**3.44**
prism,rev	2.81	8.59	3.67	2.84	2.49	2.54	2.45	7.25	3.17	2.49	**2.30**	2.34
avg	4.85	10.57	6.17	5.00	4.73	**4.64**	4.92	10.53	6.05	5.08	4.85	4.72

When we compare baseline systems (columns B in the table) with the results obtained with initialized models for discriminative training (columns C_0 and R_0), we can see that C_0 is always significantly worse than the baseline. Initialization R_0 is much better — the results as they are only slightly degraded compared to the baseline indicating that we were able to represent the original i-vector model well.

We can see, that starting from C_0, we reach significant improvements with discriminative re-estimation of parameters. Unfortunately, these results indicate, that the model got stuck in a local minimum and it was not able to improve to the level of the baseline.

Initialization variant R_0 proved to be a significantly better starting point. After discriminative parameter re-estimation, model R was able to obtain slight improvement across all benchmarks w.r.t. R_0. Model R has also achieved a slight improvement over the baseline B or almost reached its performance.

Observing results in columns D, we can compare with discriminative retraining of the full i-vector representation from previous Section 8.3.1 (or with [Novotný et al., 2019b]). With model D we achieve the best overall performance (slightly better than R), but the architecture with factorization offers approximately 4 times faster training with 7.5 times less parameters.

Chapter 9

Impact of Normalization on Language Robustness

In Section 1.4, we introduced normalization techniques from a theoretical perspective. Let us now focus on their practical application and impact on system performance. The score normalization is the last step, where it is possible to cope with unwanted variability caused by factors mentioned in Chapter 2. We focus on language variability as we did in Section 8.2, but now in the score space.

This chapter is based on research and publication from [Matějka et al., 2017], on which we worked closely with Pavel Matějka, Oldřich Plchot, Lukáš Burget, Mireia Diez Sánchez, and Jan Černocký, and included in this thesis with the kind permission of the first author.

For speaker verification systems, score normalization is one of the standard steps in producing well-calibrated speaker verification scores. Different distributions of target and non-target scores can be obtained for two different enrollment speaker models without normalization. This makes it impossible to set a single detection threshold for the different speaker models' scores. Similarly, for the same speaker model, the score distributions can vary depending on the test utterance condition (recording channel, acoustic conditions, or the utterance language), which calls for a condition-dependent threshold.

Typically, the normalization step shifts and scales the distributions for the individual models and/or conditions to allow for a single detection threshold. The shifts and scales are usually estimated using a set of utterances called normalization cohort.

In this chapter, we analyze several score normalization techniques (from Section 1.4) for evaluation benchmarks with multiple languages.

9.1 Experimental Setup

9.1.1 Evaluation Sets

For system evaluation, we used the *sre16-all*, *tel-tel* and *prism,lan* benchmarks. In addition to the original PRISM set files, we generated short cuts from these files, resembling the histogram of durations of the *sre16-all* set. The motivation for evaluating on this set is that it contains multiple languages and a channel similar to the training data.

9.1.2 System Description

Our system employs gender-independent i-vector extraction and PLDA scoring. The front-end operates on standard 19 Mel-Frequency Cepstrum Coefficients (MFCC) with C0, delta and double deltas, which are short-term mean- and variance-normalized over a 3 s sliding window. The universal background model (UBM) has 2048 diagonal-covariance Gaussians and the i-vector extractor produces 600-dimensional vectors. UBM was trained on approximately 8500 telephone files (313 hours of speech after VAD), i-vector extractor on 75000 files (3650 hours) and PLDA on 121000 files (5300 hours) defined by PRISM set. Additionally, we generated utterances with artificially added noise, reverberation (Section 3.3) and short cuts from non-English files which were added to PLDA training to simulate the properties of the data in NIST SRE 2016[1]. Results are reported as in terms of DCF^{min} as defined for NIST SRE 2016 (Section 3.1.3).

9.1.3 Normalization Cohorts

- *NIST* — contains one utterance per speaker from NIST SRE training data. It comes from different channels and contains only limited amount of data with the same languages as sre16evl. We experimented with different selection methods not to include many utterances of each speaker to the cohort, but there was no significant change in the results.
- *LID* — contains files in many languages from our development and evaluation data from the NIST Language Recognition Evaluation [Jančík et al., 2010, Plchot et al., 2016b]. This set was chosen to address the language mismatch problem. In total, there are 75k files from 57 languages with nominal durations longer than 30 s of speech.
- *SRE16 unlabeled* — provided as development data for NIST SRE 2016 as matched channel and language data. This set should be ideal for score normalization for *sre16-all*.
- *SRE16 minor* — this set mimics the scenario of similar recording channels but different languages.

9.2 Results

This section presents primarily results on the *sre16-all* benchmark (test set), as it introduced new variabilities which were not present in previous evaluations. Later, we complete the analysis on the well known *tel-tel* benchmark and also on the *prism,lan* benchmarks from PRISM set.

Figure 9.1 and Table 9.1 show the results on all pooled NIST SRE16 trials with different score normalization techniques and with three different cohort sets for normalization. NIST cohort set contain different languages and different channel than evaluation data, *SRE16 unlabeled* contains the same languages and channel as evaluation data and the last one is a pool of these two. The results without any score normalization are marked as "baseline". For all experiments with adaptive score normalization, we used the top 200 files. By inspecting Figure 9.1, we can make several conclusions:

[1]The numbers of files and hours of speech above already include these additional noisy and reverberated files.

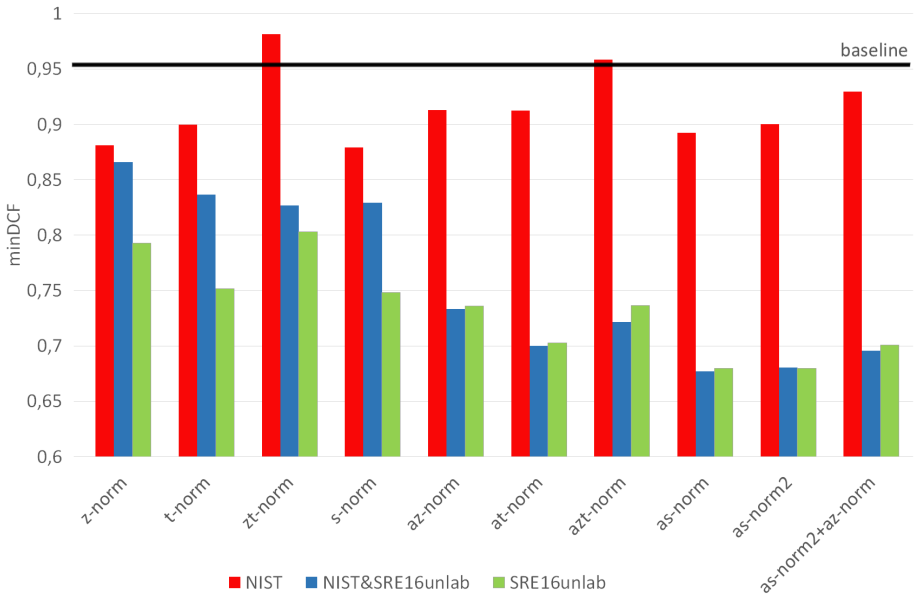


Figure 9.1: Comparison of different score normalization techniques - DCF^{min} on all pooled trials from NIST SRE 2016.

- S-norm produces the best results with 30 % relative improvement, T-norm is behind and Z-norm is worse. ZT-norm produces the worst results in this setup.
- if matched data are present in the cohort, the results are much better, if they are not present, there is only a slight improvement over the baseline.
- if matched data are present in the cohort then the adaptive score normalization is always better than using all data, because it selects the "correct" cohort.
- adaptive S-norm2 used in [Cumani et al., 2011] performs about the same as adaptive S-norm1. We also tried to apply adaptive Z-norm on the top of adaptive S-norm2 [Cumani et al., 2011], but our results did not show any improvement.

Figure 9.2 shows DCF^{min} as a function of the size of selected cohort in adaptive S-norm1. As before, there are three different cohort sets. All curves have nice flat minima between 200–500, we prefer 200 for practical reasons. The same trend was observed also on other benchmarks. It is also clear that using all data from the cohort yields worse results.

There are few tricks learned during these experiments to eliminate outliers from the cohort. The cohort set has an assumption to contain only one file per speaker, which might be hard to ensure in reality. When designing the cohort set on data without speaker labels, it is better to run unsupervised speaker clustering [Shum et al., 2014] and take only one file from each cluster. When selecting the cohort scores, it is also advantageous to eliminate/reject outlier scores by setting a "safe" interval from minus to plus 4–5 times standard deviation around the mean, and reject all cohort data outside.

Figure 9.3 compares adaptive S-norm1 with top 200 files in the cohort for different benchmarks and different cohorts. By examining all benchmarks and cohorts we conclude that:

- The cohort should contain the same languages as the evaluation set - channel match is not enough. This can be seen on the *sre16-all* benchmark with *SRE16 unlabeled*

Table 9.1: DCF^{min} for different score normalization techniques for pooled NIST SRE 2016 with different datasets in cohort.

norm. / cohort	NIST	NIST& SRE16unlab	SRE16unlab
baseline (no-norm)	0.9539	0.9538	0.9538
z-norm	0.8811	0.8661	0.7926
t-norm	0.8996	0.8366	0.7514
zt-norm	0.9814	0.8270	0.8029
s-norm	**0.8790**	0.8294	0.7483
az-norm1	0.9131	0.7335	0.7362
at-norm1	0.9124	0.6998	0.7026
azt-norm1	0.9584	0.7214	0.7365
as-norm1	0.8922	**0.6771**	**0.6797**
as-norm2	0.8999	0.6806	**0.6797**
as-norm2+az	0.9293	0.6954	0.7010

Table 9.2: Results for all pooled NIST SRE 2016 trials for different cohorts, and adaptive s-norm with top 200 selected files.

Cohort Set	DCF^{min}
Baseline (no norm)	0.9538
NIST	0.8922
LID	0.7712
NIST + LID	0.7739
NIST + SRE16unlabeled	0.6771
NIST + LID + SRE16unlabeled	**0.6733**
SRE16 unlabeled	0.6797
SRE16 minor	0.7418

(contains the same languages as the evaluation set) and *SRE16 minor* (similar channel as the evaluation data, but different languages).

- if the cohort is too different, the performance can even degrade (see *tel-tel* benchmarks and *SRE16 unlabeled* and *SRE16 minor* cohort sets).
- big set of LID data in the cohort is generally helpful especially in multilingual benchmarks.

Table 9.2 presents the details of DCF^{min} for SRE 2016 all trials.

The following step was to analyze which files were selected to form the cohort during the adaptive S-norm1 process. We ran this analysis with the top 50 files with *NIST+SRE16 unlabeled* cohort set. The first part of Table 9.3 shows the results on *sre16-all* and the languages with the largest coverage in the selected cohorts. *SRE16 unlabeled* is present in the cohorts with 51 % and 64 % for male and female trials respectively which is obvious because it contains matched languages (Cantonese and Tagalog) and channel data. The second is Cantonese which is the target language (in this case from NIST data). English is the third most probable because it has a lot of files in the cohorts under various benchmarks.

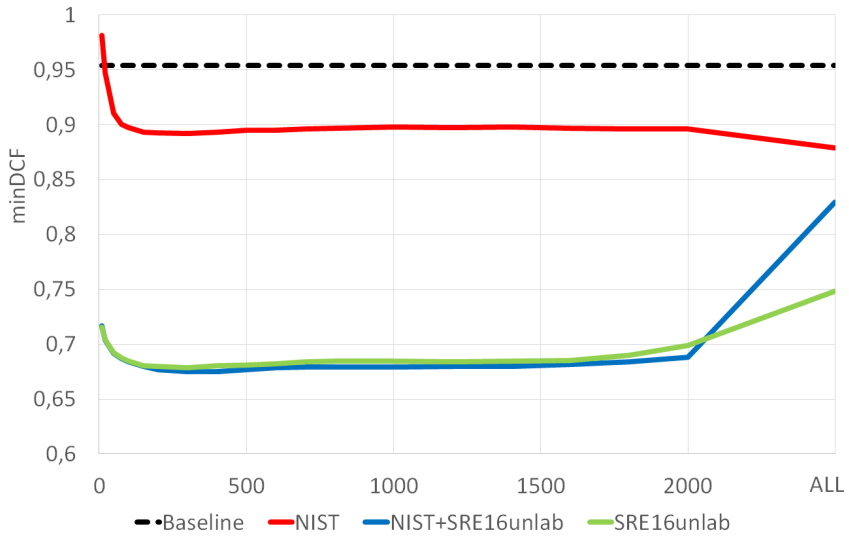


Figure 9.2: Numbers of files selected to the cohort in adaptive S-norm — results are in DCF^{min} on all pooled trials from NIST SRE 2016.

The following Mandarin, Vietnamese, Thai and Tagalog are languages quite close to target ones, with Tagalog being one of the target languages.

The *sre16-all* set contains 63.9 % of Cantonese and 36.1 % Tagalog data for male speakers. The selected cohort sets contain data with the same language from *SRE16 unlabeled*[2] (51 %), Cantonese (10.2 %) and Tagalog (0.3 %), the total evaluation language match is therefore 61.5 %. For females, this number is 74.5 %.

The second part of Table 9.3 describes PRISM *prism,lan* benchmark with true distribution of languages given in brackets. There is again a strong agreement between what language is selected to cohorts and the true percentage. The same is valid also for the last English benchmark *tel-tel* where more than 90 % of data selected to cohorts comes from English.

Globally, for all test data and all languages, we obtain in average 68 % agreement between the language of enrollment and test files and language selected to the cohorts. There is also strong agreement in gender – the files in selected cohorts match in 92 % cases the gender of the evaluation benchmark.

9.3 Summary

The analysis shows that using adaptive symmetric score normalization (s-norm) performs the best with 30 % relative improvement over baseline without any normalization. The best results were achieved by selecting 200 to 500 top scoring files to create a speaker-dependent cohort. Further analysis shows that the selected cohorts match in 68 % the language and in 92 % the gender of the enrollment and test recordings. Our experimental results also suggest that the general score normalization cohort should be a pool of several languages and channels, and, if possible, its subset should contain the data from the target domain (language and channel).

[2]Contains Cantonse and Tagalog data, but there are no labels for this data, and we do not know precise numbers.

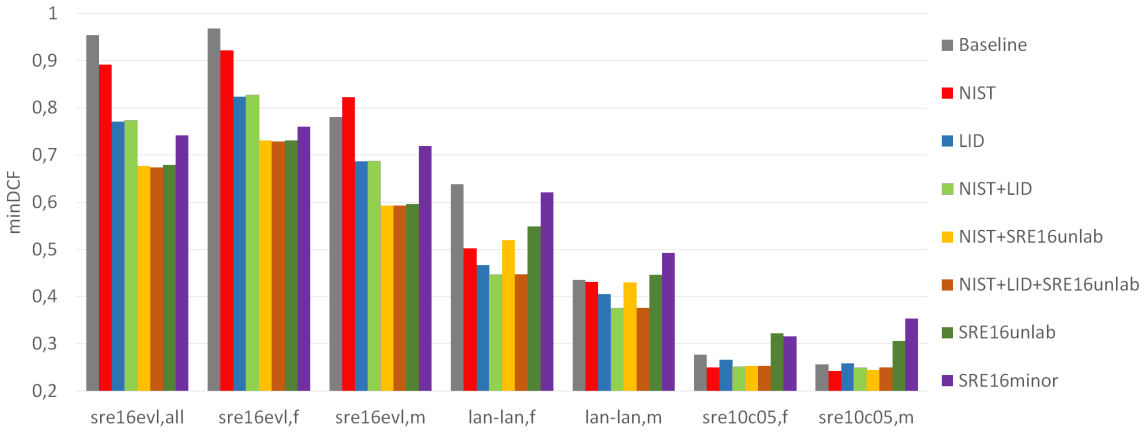


Figure 9.3: Comparison of adaptive s-norm with different cohort sets with top 200 selected files.

Table 9.3: Per-language analysis of files selected to the cohorts in adaptive score normalization. The numbers in brackets show real percentage distribution of languages in the set.

Benchmark	Language	Male DCF^{min}	Female DCF^{min}
sre16-all	SRE16unlab	51.1	64.3
	Cantonese	10.2 (63.9)	9.0 (43.7)
	English	7.7	4.6
	Mandarin	5.8	2.9
	Vietnamese	2.4	4.8
	Arabic	5.2	2.9
	Thai	1.0	3.3
	Tagalog	0.3 (36.1)	1.2 (56.3)
prism,lan	English	61.7 (72.3)	55.4 (71.9)
	Mandarin	11.1 (17.2)	14.5 (25.1)
	Arabic	6.3 (3.6)	2.4 (2.8)
	Russian	1.6 (2.6)	5.5 (9.4)
	Thai	1.0 (4.4)	6.0 (13.0)
tel-tel	English	91.0 (100)	93.8 (100)
	other	<1.0	<1.0

Chapter 10

Impact of Data on the Robustness of Discriminative Systems

In Chapter 6, we described a system training paradigm referred to as *multi-conditional* that addresses acoustic conditions such as noise and reverberation. We presented the techniques mainly in a context of generative models. If used correctly, multi-conditional training helped introduce unwanted variability into the model and increase its robustness. In the case of discriminative models, multi-conditional training is usually an intrinsic part of the training procedure. It is necessary not only for robustness of the systems but also for their generalization, especially on unseen data domains. Multi-condition data can also act as a regularization element during training. Discriminative models are significantly more sensitive to data selection. The data itself can affect the model's topology; for example, the number of speakers determines the size of the output layer of the x-vector NN. Generative models were less demanding in this respect, due to their simplicity and parameter sharing, for example, (P)LDA within-speaker variability represented by a single Gaussian shared by all speakers within the training set.

In this chapter, we use the x-vector model and extend the analysis from Section 7.3, which already presents the *x-vector* as a robust feature for PLDA modeling. We experiment with training data, separately analyzing the effect of augmentation and the number of training speakers. The analysis helps us identify the training data's critical attributes for the subsequent proper training of discriminative systems. Experiments are based on [Novotný et al., 2018b].

10.1 Experimental Setup

10.1.1 x-vectors System

As the x-vector embedding system, we used the original DNN topology presented in Section 4.2. Input features to the DNN were 20 MFCCs, extracted using a 25 ms Hamming window with 10 ms overlap. We used 23 Mel-filters, and we limited the bandwidth to 20–3700 Hz range. Resulting 20-dimensional feature vectors were subjected to short time mean- and variance-normalization using a 3 s sliding window. We used default energy-based VAD from the Kaldi recipe [Snyder, 2017].

10.1.2 PLDA Augmentation Sets

We prepared four variants of PLDA training sets with clean and augmented data, exactly as described in Section 7.2.

10.1.3 Embedding Extractor Augmentation Sets

We experimented with four sets of DNN training data. We kept most of the parameters from the original recipe. Every speaker had to have at least 6 utterances (set to 8 in the original recipe) and every utterance had to be at least 500 frames long. The consequence of this constraint was having fewer speakers, especially in training on clean data, because there, utterances had not been duplicated duplicated by the augmentation. It is worth noting that increasing the number of training speakers also increases the model size due to the larger output layer.

The statistics of the training data for all four models are listed in Table 10.1. Our first model was trained only on the "clean" original data without any augmentation. The second model (Aug I.) was trained on augmented data, but the number of speakers was limited to be the same as in the first model. The third model (Aug II.) was similar to the second but without any limitation.

In the original Kaldi recipe, training data were augmented with reverberation, noise, music, and babble and combined with original clean data. The package of all noises and room impulse responses can be downloaded from OpenSLR[1] [Ko et al., 2017], and includes MUSAN noise corpus (details in Section 3.3).

For *augmentation with reverberation* data, the total amount of Room Impulse Responses (RIRs) is divided into two lists for medium and small rooms. The probability of selecting a small or medium room is equal. We add reverberation to obtain a single replica of the original training data.

For *augmentation with noise*, we created three replicas of the original data. The first replica was augmented by adding the MUSAN noises at SNR levels in the range of 0–15 dB. In this case, the noise was added as a foreground noise (that means several non-overlapping noises can be added to the input audio). The second replica was augmented by music at SNRs from 5 to 15dB as background noise (one noise per audio with the given SNR). The last noisy replica of training data was created by augmentation with babble noise. SNR levels were at 13–20 dB and we used 3–7 noises per audio. All augmented data were pooled and a random subset of 128k audio recordings was selected and combined with clean data. The process of data augmentation is also described in [Snyder et al., 2018].

For the last model (Aug III.), we add our augmentation: real room impulse responses and stationary noises described in Section 3.3. The original RIR list was extended by our list of real RIRs and we kept one reverberated replica. Our stationary noises were used to create another replica of data with SNR levels in range 0-20 dB. We combined all replicas and selected a subset of 150k files. As a result, we obtained 11383 speakers after filtering.

10.2 Results

We conducted a set of experiments with embeddings to analyze their robustness in different data domains. We also performed an analysis with embedding DNN extractors aimed at

[1] http://www.openslr.org/resources/28/rirs_noises.zip

Table 10.1: Numbers of speakers, utterances and amounts of speech used for training the embedding DNN. (E=embedding)

Parameter	E-clean	E-Aug I.	E-Aug II.	E-Aug III.
speakers	3359	3359	9544	11383
utterances	58965	72371	211906	268219
speech duration [h]	2488	3494	10289	13288

answering the following question: How does the system performance depend on the amount and type of the training data for the embedding DNN?

We focused on the analysis with training of the embedding system. We varied the amount of training data (and also training speakers) by the means of augmentation. Results from all four embedding networks are listed in Table 10.2. The first block represents the system trained only on original ("clean") data, without any augmentation. Next blocks represent systems with increasing number of training utterances, speakers and types of noise in the augmentation (see Table 10.1).

The first two blocks represent the same network. The difference is in augmentation: while *Embedding-clean* was trained only on clean, *Embedding-Aug I.*, was trained with augmented data, but we kept the number of speakers identical for both. We can clearly see that just adding additional hours of training data consistently improved the performance and also that the trend of contribution of the multi-condition PLDA training to the performance is consistent.

The second and the third blocks represent the comparison of different numbers of speakers in training and therefore also change in the model size (size of the output layer of the network). *Embedding-Aug I.* has 3359 speakers as the output, while *Embedding-Aug II.* has seen 9544 speakers. Again, we can see the same trend as in the previous paragraph and additional improvements in performance. The exception is the Tagalog benchmark of SRE16, which seems to be different and possibly too much out-of-domain.

The last block represents the largest network (*Embedding-Aug III.*). We extended the number of speakers and we also added more augmented data to the training. We increased also the augmentation data for the PLDA multi-condition training (see Section 3.3 and Chapter 6): we added real room impulse responses and additional set of stationary noises. This brought more improvements on the *tel-tel* and *sitw-core-core* benchmarks. On *sre16-yue-f*, we can also see overall improvement, with the exception of the outstanding result achieved by the *Embedding-Aug II.* with PLDA+N. On the remaining benchmarks, the largest network has kept its robustness and similar performance.

We can conclude that the analyzed embedding architecture shows a robust performance under various benchmarks. We have verified that this architecture is indeed data-hungry — by extending the original recipe with our collection of augmentation data and, at the same time, further improving the performance. Based on Table 10.2, we see that *Aug III.*, a variant of the training set, provides the best results on PLDA without multi-conditional training across almost all evaluation benchmarks. This suggests it is a good starting point for future experiments and tuning of score estimation and normalization.

Table 10.2: Results (EER [%]) obtained in four scenarios. Each block corresponds to a system trained on different data (see Table 10.1). Blocks are divided into columns corresponding to the systems trained in multi-condition fashion (with noised and reverberated data in PLDA). Each column correspond to different PLDA multi-condition training set: N - noise, RR - real reverberation, or both (+). Each value set in bold is the minimum in the particular benchmark.

	Embedding-clean				Embedding-Aug I.				Embedding-Aug II.				Embedding-Aug III.			
	PLDA clean	PLDA extension data			PLDA clean	PLDA extension data			PLDA clean	PLDA extension data			PLDA clean	PLDA extension data		
Benchmark	—	N	RR	RR+N	—	N	RR	RR+N	—	N	RR	RR+N	—	N	RR	RR+N
tel-tel	1.38	1.73	1.36	1.39	1.18	1.46	1.15	1.16	1.04	1.28	1.05	1.08	0.94	1.09	**0.88**	0.94
sre16-tgl-f	24.45	24.20	24.53	24.07	22.08	21.84	21.71	**21.68**	23.16	21.83	23.2	22.94	22.89	21.92	22.84	22.48
sre16-yue-f	12.46	12.18	12.64	12.41	11.53	11.11	11.25	11.23	10.07	**8.61**	9.98	9.86	9.66	9.43	9.59	9.54
int-int	4.19	4.58	4.06	3.93	3.60	4.07	3.57	3.54	3.28	3.61	**3.22**	3.23	3.40	3.64	3.38	3.34
int-mic	1.49	1.62	1.54	1.44	1.41	1.39	1.34	1.30	1.08	1.24	1.09	**1.01**	1.17	1.30	1.14	1.07
sitw-core-core	10.33	10.41	10.25	9.76	8.62	8.35	8.37	8.22	8.18	8.15	8.12	7.95	7.87	7.71	7.62	**7.38**

Chapter 11

Conclusions

11.1 Summary

The field of speech recognition has undergone an unprecedented development in the recent years. The main driver for this development is the dramatic increase in the available computing power, especially in graphics cards, which have supported the possibilities of research in the area of discriminative training. This significant development has also been supported by the regular organization of international competitions (NIST evaluations), which allow for an objective comparison of the developed systems and techniques. In this work, we focused primarily on speaker verification systems, which are currently experiencing mass expansion in many fields, such as law enforcement, security, but also in various private-sector applications, such as banking authentication. Such systems achieve excellent results in laboratory conditions, but face many issues in practical use. In the first part of the work, we focused on a theoretical introduction of the state-of-the-art in SV and on the description of the factors that are responsible for performance degradation in various acoustic conditions. These factors include background noise, reverberation, or the speaker's language. We refer to these properties and factors as "domains".

After the presentation of evaluation datasets, we immersed ourselves in the evaluation metrics. We outlined some of the reasons for using multiple evaluation metrics and the problems of operating points of deployed systems.

In the subsequent chapters 4 and 5, we summarized the origins and history of two contemporary approaches to the extraction of an embedding from an audio recording—generative and discriminative—both of which have dominated the last decade, and which are still considered to be state-of-the-art. We tried to capture the fundamental differences between the two approaches and their use and comparison.

The work can be seen as a traverse through the subsequent blocks of speaker verification systems and offering or studying a solution of making each block more robust to the selected factors. We began with a presentation of speech enhancement as speech signal preprocessing. The fact that it is a signal preprocessing allows a wide deployment of this approach without a need to change the subsequently used system. This technique proved to be very useful in the presented experiments, regardless of the embeddings used; generative i-vectors or discriminative x-vectors. We also investigated what data should be used in speech enhancement training to achieve maximum improvement in terms of real deployment. We have presented that with speech enhancement, it is possible to achieve an average 12 % relative improvement of a (already robust) system based on x-vectors when tested in noisy or reverberated conditions or different communication channels.

In the next part of the work, we presented gradual arrival of discriminative techniques into generative systems. The primary motivation for these approaches lied again in the increase of the performance and robustness of these systems, especially the robustness of the extracted i-vectors as vector representations of speech recordings. In the first part of Chapter 8, we presented SBN features, i.e. features generated from a neural network. This approach was compared to the DNN alignment, in which the UBM was replaced with a neural network classifying the input feature into content-related acoustic units. Both approaches were then subjected to a test of robustness to the spoken language, and an almost 20 % relative improvement on the language-based benchmark was achieved. In this chapter, we also presented a hybrid approach to training of an i-vector extractor with the aim to combine the benefits of the generative and discriminative approaches to training. We also showed that it is possible to drastically reduce the memory requirements of an i-vector extractor without reducing its performance. The presented techniques can also be seen as parts of a paradigm shift, in which the generative elements of the system are gradually replaced, up to the actual transition to a purely discriminative embedding extractor in the form of an x-vector.

The presentation of the possibilities of score normalization then closes the passage of the speech signal through the system and shows the last step, where we can increase robustness. Again, we focused on language robustness, and in case of the SRE16 benchmark, we achieved almost 30 % relative improvement.

Throughout the work, we compared generative and discriminative approaches to embedding extraction. From a fundamental point of view, the discriminative x-vector can then be described as very sensitive to the quantitative but also qualitative use of training data. Wrong choice of data can result in a significantly more serious system degradation than one that would occur in case of a generative approach. In the last chapter, we show the dependence not only on the variety of recording conditions but also on the number of speakers in the training data. Discriminative SV proved that they can benefit from extensive training data sets with augmentation and a high number of speakers.

11.2 Future Work

11.2.1 Possible Improvement of x-vectors

Current trends in SV research are clearly focused on the discriminative approach. There is still a large number of unexplored aspects in this field, which gives room for further progress. However, it should be noted that current efforts are largely focused on the embedding extractor. X-vectors come with a fully discriminative approach to embedding extraction, but the generative PLDA is still very often used for the trial score estimator. Discriminative PLDA has been already presented [Burget et al., 2011], but it still has not been fully deployed. This indicates a possible direction for further research. Discriminative PLDA could be involved in the training pipeline in order to form an end-to-end system allowing the application of new strategies in training and batch creation compared to the current situation. However, the possibilities are broader, and one can expect a future duel of the N-class classification approach when training the x-vector extractor, with the 2-class one, addressing directly the SV objective.

Another possibility lies in a significant disadvantage of x-vectors, specifically in the absence of information about the quality or uncertainty of their estimation. In the case of i-vectors, it was possible to predict this information from its precision matrix, which could

then be used, for example, in full-posterior PLDA. In the case of x-vectors, this information is missing, although there are already attempts to achieve it [Silnova et al., 2020]; so far, this area is still open to further progress.

11.2.2 Possible Improvement of i-vectors

Here we need to ask the question: do i-vectors have a chance to compete with better and more robust x-vectors in the future? In their original generative form, most likely not. The hybrid approach presented in this work could provide hope. However, it would mean an even more significant transfer from generative to discriminative training. The motivation for this effort "not to leave the i-vector approach" is the already mentioned uncertainty estimation, which is missing in the case of x-vectors. However, as it has already been shown, the simple discriminative overtraining of the i-vector extractor will most likely not bring a sufficient increase in recognition accuracy. The solution could lie in introducing the concept of i-vectors into the x-vector network, for example, as a kind of pooling layer using attention mechanism instead of UBM.

Bibliography

[Apsingekar and Leon, 2011] Apsingekar, V. R. and Leon, P. L. D. (2011). Speaker verification score normalization using speaker model clusters. *Speech Communications*, 53(1):110–118.

[Aronowitz, 2014] Aronowitz, H. (2014). Compensating Inter-Dataset Variability in PLDA Hyper-Parameters for Robust Speaker Recognition. In *Proceedings of Odyssey 2014*.

[Aronowitz and Aronowitz, 2010] Aronowitz, H. and Aronowitz, V. (2010). Efficient score normalization for speaker recognition. In *Proceedings of ICASSP 2010*.

[Aronowitz et al., 2005] Aronowitz, H., Irony, D., and Burshtein, D. (2005). Modeling intra-speaker variability for speaker recognition. In *Proceedings of Interspeech 2005*.

[Arslan and Engin, 2019] Arslan, Ö. and Engin, E. Z. (2019). Noise robust voice activity detection based on multi-layer feed-forward neural network.

[Auckenthaler et al., 2000] Auckenthaler, R., Carey, M., and Lloyd-Thomas, H. (2000). Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10(1):42 – 54.

[Axelrod et al., 2003] Axelrod, S., Goel, V., Kingsbury, B., Visweswariah, K., and Gopinath, R. A. (2003). Large vocabulary conversational speech recognition with a subspace constraint on inverse covariance matrices. In *Proceedings of Interspeech 2003*.

[Ben et al., 2002] Ben, M., Blouet, R., and Bimbot, F. (2002). A Monte-Carlo method for score normalization in automatic speaker verification using kullback-leibler distances. In *Proceedings of ICASSP 2002*, pages 689–692.

[Benesty et al., 2008] Benesty, J., Sondhi, M. M., and Huang, Y., editors (2008). *Springer Handbook of Speech Processing*. Springer Handbooks. Springer, Berlin.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Boll, 1979] Boll, S. (1979). Suppression of acoustic noise in speech using spectral subtraction. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(2):113 – 120.

[Bredin et al., 2006] Bredin, H., Dehak, N., and Chollet, G. (2006). GMM-based SVM for face recognition. volume 3, pages 1111–1114.

[Brümmer, 2004] Brümmer, N. (2004). Spescom DataVoice NIST 2004 system description. In *Proc. NIST Speaker Recognition Evaluation 2004*, Toledo, Spain.

[Brümmer, 2009] Brümmer, N. (2009). EM for JFA (EM4JFA). Technical report, Agnitio Research, Sount Africa.

[Brümmer, 2010] Brümmer, N. (2010). *Measuring, Refining and Calibrating Speaker and Language Information Extracted from Speech*. PhD thesis, University of Stellenbosch.

[Brümmer et al., 2007] Brümmer, N., Burget, L., Černocký, J., Glembek, O., Grézl, F., Karafiát, M., van Leeuwen, D., Matějka, P., Schwarz, P., and Strasheim, A. (2007). Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006. *IEEE Transactions on Audio, Speech and Language Processing*, 15(7):2072–2084.

[Brümmer and du Preez, 2006] Brümmer, N. and du Preez, J. (2006). Application-independent evaluation of speaker detection. *Computer Speech & Language*, 20(2-3):230–275.

[Brümmer and Villiers, 2010] Brümmer, N. and Villiers, E. D. (2010). The speaker partitioning problem. In *Proceedings of Odyssey 2010*.

[Burget, 2004] Burget, L. (2004). *Complementarity of Speech Recognition Systems and System Combination*. PhD thesis, Brno University of Technology.

[Burget et al., 2008] Burget, L., Brummer, N., Reynolds, D., Kenny, P., Pelecanos, J., Vogt, R., Castaldo, F., Dehak, N., Dehak, R., Glembek, O., Karam, Z., Noecker, J. J., Na, Y. H., Costin, C. C., Hubeika, V., Kajarekar, S., Scheffer, N., and Černocký, J. (2008). Robust speaker recognition over varying channels. Technical report, Johns Hopkins University.

[Burget et al., 2007] Burget, L., Matejka, P., Schwarz, P., Glembek, O., and Cernocky, J. H. (2007). Analysis of Feature Extraction and Channel Compensation in a GMM Speaker Recognition System. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):1979–1986.

[Burget et al., 2011] Burget, L., Plchot, O., Cumani, S., Glembek, O., Matějka, P., and Brümmer, N. (2011). Discriminatively Trained Probabilistic Linear Discriminant Analysis for Speaker Verification. In *Proceedings of ICASSP 2011*, pages 4832–4835. IEEE Signal Processing Society.

[Cai et al., 2018] Cai, W., Cai, Z., Zhang, X., Wang, X., and Li, M. (2018). A novel learnable dictionary encoding layer for end-to-end language identification. pages 5189–5193.

[Cieri et al., 2004] Cieri, C., Miller, D., and Walker, K. (2004). The fisher corpus: A resource for the next generations of speech-to-text.

[Cumani et al., 2011] Cumani, S., Batzu, P. D., Colibro, D., Vair, C., Laface, P., and Vasilakakis, V. (2011). Comparison of speaker recognition approaches for real applications. In *Proceedings of Interspeech 2011*, Florence, Italy.

[Cumani and Laface, 2013] Cumani, S. and Laface, P. (2013). Memory and computation trade-offs for efficient i-vector extraction. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 21(5):934–944.

[Cumani and Laface, 2014] Cumani, S. and Laface, P. (2014). Factorized Sub-Space Estimation for Fast and Memory Effective I-vector Extraction. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(1):248–259.

[Dahl et al., 2012] Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

[Davis and Mermelstein, 1980] Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4).

[Dehak et al., 2010] Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2010). Front-End Factor Analysis For Speaker Verification. *IEEE Transactions on Audio, Speech and Language Processing*, pages 1 –1.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

[Deng et al., 2019] Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694.

[Doddington et al., 2000] Doddington, G. R., Przybocki, M. A., Martin, A. F., and Reynolds, D. A. (2000). The NIST speaker recognition evaluation – Overview, methodology, systems, results, perspective. *Speech Communication*, 31(2):225–254.

[Dufera and Shimamura, 2009] Dufera, B. and Shimamura, T. (2009). Reverberated speech enhancement using neural networks. In *Proc. International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2009.*, pages 441–444.

[ETSI, 2007] ETSI (2007). Speech Processing, Transmission and Quality Aspects (STQ). Technical Report ETSI ES 202 050, European Telecommunications Standards Institute (ETSI).

[Fér et al., 2015] Fér, R., Matějka, P., Grézl, F., Plchot, O., and Černocký, J. (2015). Multilingual Bottleneck Features for Language Recognition. *Interspeech 2015*.

[Ferrer et al., 2011a] Ferrer, L., Bratt, H., Burget, L., Cernocky, H., Glembek, O., Graciarena, M., Lawson, A., Lei, Y., Matejka, P., Plchot, O., and Scheffer, N. (2011a). The PRISM set. https://code.google.com/p/prism-set/.

[Ferrer et al., 2011b] Ferrer, L., Bratt, H., Burget, L., Cernocky, H., Glembek, O., Graciarena, M., Lawson, A., Lei, Y., Matejka, P., Plchot, O., and Scheffer, N. (2011b). Promoting robustness for speaker modeling in the community: the PRISM evaluation set. In *Proceedings of SRE11 analysis workshop*, Atlanta.

[Fortuna et al., 2004] Fortuna, J., Sivakumaran, P., Ariyaeeinia, A. M., and Malegaonkar, A. (2004). Relative effectiveness of score normalization methods in open-set speaker identification. In *Proceedings of Odyssey 2004*, Toledo, Spain.

[Furui, 1986] Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34:52–59.

[Garcia-Romero and Espy-Wilson, 2011] Garcia-Romero, D. and Espy-Wilson, C. Y. (2011). Analysis of i-vector length normalization in Gaussian-PLDA speaker recognition systems. In *Proceedings of Interspeech 2011*.

[Garcia-Romero et al., 2012] Garcia-Romero, D., Zhou, X., and Espy-Wilson, C. Y. (2012). Multicondition training of Gaussian PLDA models in i-vector space for noise and reverberation robust speaker recognition. In *Proceedings of ICASSP 2012*, pages 4257–4260.

[Ghoshal et al., 2013] Ghoshal, A., Swietojanski, P., and Renals, S. (2013). Multilingual training of deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7319–7323.

[Glembek, 2012] Glembek, O. (2012). *Optimization of Gaussian Mixture Subspace Models and Related Scoring Algorithms in Speaker Verification*. Ph.D. thesis, Brno University of Technology, Faculty of Information Technology.

[Glembek et al., 2011a] Glembek, O., Burget, L., Brümmer, N., Plchot, O., and Matějka, P. (2011a). Discriminatively Trained i-vector Extractor for Speaker Verification. In *Proceedings of Interspeech 2011*, number 8, pages 137–140. International Speech Communication Association.

[Glembek et al., 2014] Glembek, O., Ma, J., Matějka, P., Zhang, B., Plchot, O., Burget, L., and Matsoukas, S. (2014). Domain Adaptation via Within-class Covariance Correction in I-Vector Based Speaker Recognition Systerms. In *Proceedings of ICASSP 2014*, pages 4060–4064.

[Glembek et al., 2011b] Glembek, O., Matějka, P., and Burget, L. (2011b). Simplification and optimization of i-vector extraction. In *Proceedings of ICASSP 2011*, Prague, CZ.

[Graff et al., 2001] Graff, D., Miller, D., and Walker, K. (2001). Switchboard cellular part 1 audio. *Linguistic Data Consortium, Philadelphia*.

[Graff et al., 2002] Graff, D., Miller, D., and Walker, K. (2002). Switchboard-2 phase III. *Linguistic Data Consortium, Philadelphia*.

[Graff et al., 2004] Graff, D., Miller, D., and Walker, K. (2004). Switchboard cellular part 2 audio. *Linguistic Data Consortium, Philadelphia*.

[Graff et al., 1999] Graff, D., Walker, K., and Canavan, A. (1999). Switchboard-2 phase II. *Linguistic Data Consortium, Philadelphia*.

[Grézl et al., 2007] Grézl, F., Karafiát, M., Kontár, S., and Černocký, J. (2007). Probabilistic and bottle-neck features for LVCSR of meetings. In *Proceedings of ICASSP 2007*, pages 757–760.

[Habets, 2006] Habets, E. A. (2006). Room impulse response generator. Technical report. https://www.researchgate.net/profile/Emanuel-Habets/publication/259991276_Room_Impulse_Response_Generator/links/5800ea5808ae1d2d72eae2a0/Room-Impulse-Response-Generator.pdf.

[Harper, 2013] Harper, M. (2013). The BABEL program and low resource speech technology. In *ASRU 2013*.

[Hatch et al., 2006] Hatch, A. O., Kajarekar, S., and Stolcke, A. (2006). Within-Class Covariance Normalization for SVM-based speaker recognition. In *Proc. ICSLP, Pittsburgh, USA*, pages 1471–1474.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Hinton et al., 2012] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[Hirsch, 2005] Hirsch, H. G. (2005). *Fant-filtering and noise adding tool*. Niederrhein University of Applied Sciences. http://dnt.-kr.hsnr.de/download.html.

[ITU, 1994] ITU (1994). *ITU-T Recommendation O.41*. International Telecommunication Union. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-O.41-199410-I!!PDF-E&type=items.

[Jančík et al., 2010] Jančík, Z., Plchot, O., Brummer, N., Burget, L., Glembek, O., Hubeika, V., Karafiát, M., Matějka, P., Mikolov, T., Strasheim, A., and Černocký, J. (2010). Data selection and calibration issues in automatic language recognition - investigation with BUT-AGNITIO NIST LRE 2009 system. In *Proceedings of Odyssey 2010*, pages 215–221.

[Karafiát et al., 2016] Karafiát, M., Baskar, M. K., Grézl, F., Veselý, K., and Černocký, J. H. (2016). Multilingual BLSTM and SSNN adaptation in Babel. In *submitted to SLT*.

[Karafiát et al., 2013] Karafiát, M., Grézl, F., Hannemann, M., Veselý, K., and Černocký, J. H. (2013). BUT BABEL System for Spontaneous Cantonese. In *Proceedings of Interspeech 2013*, pages 2589–2593, Lyon, France.

[Karafiát et al., 2014] Karafiát, M., Grézl, F., Veselý, K., Hannemann, M., Szőke, I., and Černocký, J. (2014). BUT 2014 Babel system: Analysis of adaptation in NN based systems. In *Proceedings of Interspeech 2014*, pages 3002–3006.

[Kenny, 2005] Kenny, P. (2005). Joint factor analysis of speaker and session variability: Theory and algorithms - technical report CRIM-06/08-13. Montreal, CRIM, 2005. https://www.crim.ca/perso/patrick.kenny/FAtheory.pdf.

[Kenny, 2010] Kenny, P. (2010). Bayesian speaker verification with heavy–tailed priors. keynote presentation, Proceedings of Odyssey 2010.

[Kenny et al., 2007] Kenny, P., Boulianne, G., Ouellet, P., and Dumouchel, P. (2007). Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1435–1447.

[Kenny et al., 2003] Kenny, P., Mihoubi, M., and Dumouchel, P. (2003). New map estimators for speaker recognition. In *Proceedings of Interspeech 2003*.

[Kinnunen and Li, 2010] Kinnunen, T. and Li, H. (2010). An overview of text-independent speaker recognition: from features to supervectors. *Speech Communication*, 52:12–40.

[Kirill et al., 2009] Kirill, S., Ekaterina, V., and Simak, B. (2009). Approach for energy-based voice detector with adaptive scaling factor. *IAENG International Journal of Computer Science*, 36.

[Ko et al., 2017] Ko, T., Peddinti, V., Povey, D., Seltzer, M. L., and Khudanpur, S. (2017). A study on data augmentation of reverberant speech for robust speech recognition. In *Proceedings of ICASSP 2017*, pages 5220–5224.

[Kockmann, 2012] Kockmann, M. (2012). *Subspace Modeling of Prosodic Features for Speaker Verification*. PhD thesis, Brno University of Technology.

[Kumatani et al., 2012] Kumatani, K., Arakawa, T., Yamamoto, K., McDonough, J., Raj, B., Singh, R., and Tashev, I. (2012). Microphone Array Processing for Distant Speech Recognition: Towards Real-World Deployment. In *APSIPA Annual Summit and Conference*, Hollywood, CA, USA.

[Laskowski and Edlund, 2010] Laskowski, K. and Edlund, J. (2010). A Snack implementation and Tcl/Tk Interface to the Fundamental Frequency Variation Spectrum Algorithm. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

[Lei et al., 2012] Lei, Y., Burget, L., Ferrer, L., Graciarena, M., and Scheffer, N. (2012). Towards noise-robust speaker recognition using probabilistic linear discriminant analysis. In *Proceedings of ICASSP 2012*, Kyoto, JP.

[Lei et al., 2014] Lei, Y., Scheffer, N., Ferrer, L., and McLaren, M. (2014). A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *Proceedings of ICASSP 2014*.

[Lippmann et al., 1987] Lippmann, R., Martin, E., and Paul, D. (1987). Multi-style training for robust isolated-word speech recognition. In *Proceedings of ICASSP 1987*, volume 12, pages 705–708.

[Lozano-Diez et al., 2016] Lozano-Diez, A., Silnova, A., Matejka, P., Glembek, O., Plchot, O., Pesan, J., Burget, L., and Gonzalez-Rodriguez, J. (2016). Analysis and Optimization of Bottleneck Features for Speaker Recognition. In *Proceedings of Odyssey 2016*, pages 352–357, Bilbao, Spain.

[Mandasari et al., 2012] Mandasari, M. I., McLaren, M., and van Leeuwen, D. A. (2012). The effect of noise on modern automatic speaker recognition systems. In *Proceedings of ICASSP 2012*, pages 4249–4252.

[Martin et al., 1997] Martin, A. F., Doddington, G. R., Kamm, T., Ordowski, M., and Przybocki, M. A. (1997). The DET curve in assessment of detection task performance. In *Proceedings of Eurospeech*.

[Martínez et al., 2014] Martínez, D. G., Burget, L., Stafylakis, T., Lei, Y., Kenny, P., and LLeida, E. (2014). Unscented Transform For Ivector-based Noisy Speaker Recognition. In *Proceedings of ICASSP 2014*, Florencie, IT.

[Martínez et al., 2011] Martínez, D. G., Plchot, O., Burget, L., Glembek, O., and Matějka, P. (2011). Language Recognition in iVectors Space. In *Proceedings of Interspeech 2011*, volume 2011, pages 861–864, Florence, IT.

[Mason and Zhang, 1991] Mason, J. S. and Zhang, X. (1991). Velocity and acceleration features in speaker recognition. In *Proceedings of ICASSP 1991*, pages 3673–3676 vol.5.

[Matějka et al., 2017] Matějka, P., Novotný, O., Plchot, O., Burget, L., Diez, M. S., and Černocký, J. (2017). Analysis of Score Normalization in Multilingual Speaker Recognition. In *Proceedings of Interspeech 2017*, volume 2017, pages 1567–1571.

[Matějka et al., 2006] Matějka, P., Burget, L., Schwarz, P., and Černocký, J. (2006). Brno University of Technology system for NIST 2005 language recognition evaluation. In *Proceedings of Odyssey 2006: The Speaker and Language Recognition Workshop*, pages 57–64.

[Matějka et al., 2020] Matějka, P., Plchot, O., Glembek, O., Burget, L., Rohdin, A. J., Zeinali, H., Mošner, L., Silnova, A., Novotný, O., Diez, M. S., and Černocký, J. (2020). 13 years of speaker recognition research at BUT, with longitudinal analysis of NIST SRE. *Computer Speech and Language*, 63:1–15.

[McGovern, 2009] McGovern, S. G. (2009). Fast image method for impulse response calculations of box-shaped rooms. *Applied Acoustics*, 70(1):182–189.

[McLaren et al., 2016] McLaren, M., Ferrer, L., Castan, D., and Lawson, A. (2016). The Speakers in the Wild (SITW) Speaker Recognition Database. In *Proceedings of Interspeech 2016*, pages 818–822.

[McLaren and van Leeuwen, 2012] McLaren, M. and van Leeuwen, D. (2012). Source-Normalized LDA for Robust Speaker Recognition Using i-Vectors From Multiple Speech Sources. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):755–766.

[Mimura et al., 2014] Mimura, M., Sakai, S., and Kawahara, T. (2014). Reverberant speech recognition combining deep neural networks and deep autoencoders. In *Proc. Reverb Challenge Workshop*, Florence, Italy.

[Mošner et al., 2018] Mošner, L., Matějka, P., Novotný, O., and Černocký, J. (2018). Dereverberation and beamforming in far-field speaker recognition. In *Proceedings of ICASSP 2018*.

[Nakatani et al., 2010] Nakatani, T., Yoshioka, T., Kinoshita, K., Miyoshi, M., and Juang, B. (2010). Speech dereverberation based on variance-normalized delayed linear prediction. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7):1717–1731.

[NIST, 2010] NIST (2010). The NIST year 2010 Speaker Recognition Evaluation Plan. https://www.nist.gov/system/files/documents/itl/iad/mig/NIST_SRE10_evalplan-r6.pdf.

[NIST, 2016] NIST (2016). The NIST year 2016 Speaker Recognition Evaluation Plan. https://www.nist.gov/sites/default/files/documents/2016/10/\07/sre16_eval_plan_v1.3.pdf.

[Novotný et al., 2018a] Novotný, O., Matějka, P., Plchot, O., and Glembek, O. (2018a). On the use of DNN Autoencoder for Robust Speaker Recognition. Technical report. http://www.fit.vutbr.cz/research/view_pub.php.cs?id=11855.

[Novotný et al., 2016] Novotný, O., Matějka, P., Glembek, O., Plchot, O., Grézl, F., Burget, L., and Černocký, J. (2016). Analysis of the DNN-Based SRE Systems in Multi-language Conditions. In *Proceedings of SLT 2016*, pages 199–204.

[Novotný et al., 2019a] Novotný, O., Plchot, O., Glembek, O., and Burget, L. (2019a). Factorization of Discriminatively Trained i-Vector Extractor for Speaker Recognition. In *Proceedings of Interspeech 2019*, pages 4330–4334.

[Novotný et al., 2019b] Novotný, O., Plchot, O., Glembek, O., Burget, L., and Matějka, P. (2019b). Discriminatively Re-trained i-Vector Extractor For Speaker Recognition. In *Proceedings of ICASSP 2019*, pages 6031–6035.

[Novotný et al., 2019c] Novotný, O., Plchot, O., Glembek, O., Černocký, J., and Burget, L. (2019c). Analysis of DNN Speech Signal Enhancement for Robust Speaker Recognition. *Computer Speech and Language*, 2019(58):403–421.

[Novotný et al., 2018b] Novotný, O., Plchot, O., Matějka, P., Mošner, L., and Glembek, O. (2018b). On the use of X-vectors for Robust Speaker Recognition. In *Proceedings of Odyssey 2018*, number 6, pages 168–175.

[Openshaw and Masan, 1994] Openshaw, J. and Masan, J. (1994). On the limitations of cepstral features in noise. In *Proceedings of ICASSP 1994*, Adelaide, SA, Australia.

[Peddinti et al., 2015] Peddinti, V., Povey, D., and Khudanpur, S. (2015). A time delay neural network architecture for efficient modeling of long temporal contexts. In *Proceedings of Interspeech 2015*.

[Pelecanos and Sridharan, 2006] Pelecanos, J. and Sridharan, S. (2006). Feature Warping for Robust Speaker Verification. In *Proceedings of Odyssey 2006*, Crete, Greece.

[Pešán et al., 2016] Pešán, J., Burget, L., and Černocký, J. (2016). Sequence Summarizing Neural Networks for Spoken Language Recognition. In *Proceedings of Interspeech 2016*, pages 3285–3289.

[Plchot, 2014] Plchot, O. (2014). *Extensions to Probabilistic Linear Discriminant Analysis for Speaker Recognition*. Ph.D. thesis, Brno University of Technology.

[Plchot et al., 2016a] Plchot, O., Burget, L., Aronowitz, H., and Matějka, P. (2016a). Audio Enhancing With DNN Autoencoder For Speaker Recognition. In *Proceedings of ICASSP 2016*, pages 5090–5094.

[Plchot et al., 2016b] Plchot, O., Matějka, P., Fér, R., Glembek, O., Novotný, O., Pešán, J., Veselý, K., Ondel, L., Karafiát, M., Grézl, F., Kesiraju, S., Burget, L., Brummer, N., du Albert Swart, P., Cumani, S., Mallidi, H. S., and Li, R. (2016b). BAT System Description for NIST LRE 2015. In *Proceedings of Odyssey 2016*, volume 2016, pages 166–173.

[Plchot et al., 2013] Plchot, O., Matsoukas, S., Matějka, P., Dehak, N., Ma, J., Cumani, S., Glembek, O., Heřmanský, H., Mesgarani, N., Soufifar, M. M., Thomas, S., Zhang, B., and Zhou, X. (2013). Developing a Speaker Identification System for the DARPA RATS Project. In *Proceedings of ICASSP 2013*, Vancouver, CA.

[Povey, 2006] Povey, D. (2006). SPAM and full covariance for speech recognition. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*.

[Povey et al., 2018] Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohammadi, M., and Khudanpur, S. (2018). Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks. In *Proceedings of Interspeech 2018*, pages 3743–3747.

[Prince and Elder, 2007] Prince, S. and Elder, J. (2007). Probabilistic Linear Discriminant Analysis for Inferences About Identity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

[Rabiner and Juang, 1993] Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Rajan et al., 2014] Rajan, P., Afanasyev, A., Hautamäki, V., and Kinnunen, T. (2014). From single to multiple enrollment i-vectors: Practical PLDA scoring variants for speaker verification. *Digital Signal Processing*, 31:93 – 101.

[Rajan et al., 2013] Rajan, P., Kinnunen, T., and Hautamäki, V. (2013). Effect of multicondition training on i-vector PLDA configurations for speaker recognition. In *Proceedings of Interspeech 2013*, pages 3694–3697.

[Ramos-Castro et al., 2007] Ramos-Castro, D., Fierrez-Aguilar, J., Gonzalez-Rodriguez, J., and Ortega-Garcia, J. (2007). Speaker verification using speaker- and test-dependent fast score normalization. *Pattern Recognition Letters*, 28:90 – 98.

[Ravanelli et al., 2016] Ravanelli, M., Svaizer, P., and Omologo, M. (2016). Realistic Multi-Microphone Data Simulation for Distant Speech Recognition. In *Proceedings of Interspeech 2016*, pages 2786–2790.

[Reynolds, 2002] Reynolds, D. (2002). Automatic Speaker Recognition, Acoustics and Beyond. In *SuperSID project at JHU Summer Workshop*.

[Reynolds, 1997] Reynolds, D. A. (1997). Comparison of background normalization methods for text-independent speaker verification. In *Proceedings of Eurospeech*.

[Reynolds et al., 2000] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, 10(1):19 – 41.

[Ribas et al., 2015] Ribas, D., Vincent, E., and Calvo, J. R. (2015). Full multicondition training for robust i-vector based speaker recognition. In *Proceedings of Interspeech 2015*, Dresden, Germany.

[Rohdin et al., 2018] Rohdin, J., Silnova, A., Diez, M., Plchot, O., Matějka, P., and Burget, L. (2018). End-to-end DNN based speaker recognition inspired by i-vector and PLDA. In *Proceedings of ICASSP 2018*.

[Saon et al., 2013] Saon, G., Soltau, H., Nahamoo, D., and Picheny, M. (2013). Speaker adaptation of neural network acoustic models using i-vectors. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 55–59.

[Shum et al., 2010] Shum, S., Dehak, N., Dehak, R., and Glass, J. R. (2010). Unsupervised Speaker Adaptation based on the Cosine Similarity for Text-Independent Speaker Verification. In *Proceedings of Odyssey 2010*, Brno, Czech Republic.

[Shum et al., 2014] Shum, S., Reynolds, D., Garcia-Romero, D., and McCree, A. (2014). Unsupervised clustering approaches for domain adaptation in speaker recognition systems. In *Proceedings of Odyssey 2014*, Joensuu, Finlan.

[Silnova et al., 2020] Silnova, A., Brummer, N., Rohdin, A. J., Stafylakis, T., and Burget, L. (2020). Probabilistic embeddings for speaker diarization. In *Proceedings of Odyssey 2020*, volume 2020, pages 24–31.

[Silnova et al., 2015] Silnova, A., Glembek, O., Kinnunen, T., and Matějka, P. (2015). Exploring ANN Back-Ends for i-Vector Based Speaker Age Estimation. In *Proceedings of Interspeech 2015*, pages 3036–3040.

[Silovský, 2011] Silovský, J. (2011). *Generativní a diskriminativní klasifikátory v úlohách textově nezávislého rozpoznávání a diarizace mluvčích*. PhD thesis, Technical University of Liberec.

[Snyder, 2017] Snyder, D. (2017). NIST SRE 2016 Xvector Recipe. https://david-ryan-snyder.github.io/2017/10/04/model_sre16_v2.html.

[Snyder et al., 2015] Snyder, D., Chen, G., and Povey, D. (2015). MUSAN: A Music, Speech, and Noise Corpus. arXiv:1510.08484v1.

[Snyder et al., 2017] Snyder, D., Garcia-Romero, D., Povey, D., and Khudanpur, S. (2017). Deep Neural Network Embeddings for Text-Independent Speaker Verification. *Proceedings of Interspeech 2017*, pages 999–1003.

[Snyder et al., 2019] Snyder, D., Garcia-Romero, D., Sell, G., McCree, A., Povey, D., and Khudanpur, S. (2019). Speaker Recognition for Multi-speaker Conversations Using X-vectors. In *Proceedings of ICASSP 2019*, pages 5796–5800.

[Snyder et al., 2018] Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., and Khudanpur, S. (2018). X-Vectors: Robust DNN Embeddings for Speaker Recognition. In *Proceedings of ICASSP 2018*.

[Snyder et al., 2016] Snyder, D., Ghahremani, P., Povey, D., Garcia-Romero, D., Carmiel, Y., and Khudanpur, S. (2016). Deep neural network-based speaker embeddings for end-to-end speaker verification. *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 165–170.

[Song et al., 2013] Song, Y., Jiang, B., Bao, Y., Wei, S., and Dai, L.-R. (2013). I-vector representation based on bottleneck features for language identification. *Electronics Letters*, 49(24):1569–1570.

[Stevens et al., 1937] Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

[Stewart and Sandler, 2010] Stewart, R. and Sandler, M. (2010). Database of omnidirectional and b-format room impulse responses. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 165–168.

[Sturim and Reynolds, 2005] Sturim, D. E. and Reynolds, D. A. (2005). Speaker adaptive cohort selection for Tnorm in text-independent speaker verification. In *Proceedings of ICASSP 2005*, volume 1, pages I/741–I/744 Vol. 1.

[Swart and Brümmer, 2017] Swart, A. and Brümmer, N. (2017). A Generative Model for Score Normalization in Speaker Recognition. In *Proceedings of Interspeech 2017*, pages 1477–1481.

[Talkin, 1995] Talkin, D. (1995). A Robust Algorithm for Pitch Tracking (RAPT). In Kleijn, W. B. and Paliwal, K., editors, *Speech Coding and Synthesis*, New York. Elsevier.

[Tatarinov and Pollák, 2004] Tatarinov, J. and Pollák, P. (2004). Hidden Markov Models in voice activity detection. In *COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction.*

[Thyes et al., 2000] Thyes, O., Kuhn, R., Nguyen, P., and Junqua, J.-C. (2000). Speaker identification and verification using eigenvoices. In *Proceedings of Interspeech 2000*, pages 242–245.

[Tolosana et al., 2015] Tolosana, R., Vera-Rodriguez, R., Ortega-Garcia, J., and Fierrez, J. (2015). Update Strategies for HMM-Based Dynamic Signature Biometric Systems. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS).*

[Torres-Carrasquillo et al., 2002] Torres-Carrasquillo, P. A., Singer, E., Kohler, M. A., and Deller, J. R. (2002). Approaches to language identification using Gaussian mixture models and shifted delta cepstral features. In *Proc. ICSLP 2002*, pages 89–92.

[Veselý et al., 2012] Veselý, K., Karafiát, M., Grézl, F., Janda, M., and Egorova, E. (2012). The Language-Independent Bottleneck Features. In *Proceedings of IEEE 2012 Workshop on Spoken Language Technology*, pages 336–341.

[Veselý et al., 2016] Veselý, K., Watanabe, S., Žmolíková, K., Karafiát, M., Burget, L., and Černocký, J. H. (2016). Sequence summarizing neural network for speaker adaptation. In *Proceedings of ICASSP 2016*, pages 5315–5319.

[Villalba and Brummer, 2011] Villalba, J. and Brummer, N. (2011). Towards Fully Bayesian Speaker Recognition: Integrating Out the Between-speaker Covariance. In *Proceedings of Interspeech 2011*, pages 505–508.

[Waibel et al., 1989] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

[Wiener, 1964] Wiener, N. (1964). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press.

[Xu et al., 2014a] Xu, Y., Du, J., Dai, L.-R., and Lee, C.-H. (2014a). An Experimental Study on Speech Enhancement Based on Deep Neural Networks. *IEEE Signal processing letters*, 21(1).

[Xu et al., 2014b] Xu, Y., Du, J., Dai, L.-R., and Lee, C.-H. (2014b). Global variance equalization for improving deep neural network based speech enhancement. In *Proc. IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP)*, pages 71 – 75.

[Yaman et al., 2012] Yaman, S., Pelecanos, J., and Sarikaya, R. (2012). Bottleneck Features for Speaker Recognition. In *Proceedings of Odyssey 2012*.

[Yanhui et al., 2014] Yanhui, T., Jun, D., Yong, X., Lirong, D., and Chin-Hui, L. (2014). Deep Neural Network Based Speech Separation for Robust Speech Recognition. In *Proceedings of ICSP2014*, pages 532–536.

[Ying et al., 2011] Ying, D., Li, J., Fu, Q., Yan, Y., and Dang, J. (2011). Voice activity detection based on a sequential Gaussian Mixture Model. *APSIPA ASC 2011 - Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2011*, pages 861–866.

[Young et al., 2006] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X. A., Moore, G., ulian Odell, Ollason, D., Povey, D., Valtchev, V., and Woodland, P. (2006). *The HTK Book*. Cambridge University Engineering Department.

[Zigel and Cohen, 2003] Zigel, Y. and Cohen, A. (2003). On cohort selection for speaker verification. In *Proceedings of Eurospeech*.

[Zigel and Wasserblat, 2006] Zigel, Y. and Wasserblat, M. (2006). How to deal with multiple-targets in speaker identification systems? In *Proceedings of Odyssey 2006*, San Juan, Puerto Rico.

[Zwicker, 1961] Zwicker, E. (1961). Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen). *The Journal of the Acoustical Society of America*, 33(2):248–248.