PUBLICATIONS OF
THE UNIVERSITY OF EASTERN FINLAND

Dissertations in Forestry and Natural Sciences

UNIVERSITY OF EASTERN FINLAND

VILLE VESTMAN

Methods for fast, robust, and secure speaker recognition

PUBLICATIONS OF THE UNIVERSITY OF EASTERN FINLAND
DISSERTATIONS IN FORESTRY AND NATURAL SCIENCES

N:o 389

Ville Vestman

METHODS FOR FAST, ROBUST, AND SECURE SPEAKER RECOGNITION

ACADEMIC DISSERTATION

To be presented by the permission of the Faculty of Science and Forestry for public online examination on November 10th, 2020, at 10 a.m.

University of Eastern Finland
School of Computing
Joensuu 2020

Grano Oy
Jyväskylä, 2020
Editors: Pertti Pasanen, Matti Tedre,
Jukka Tuomela, and Matti Vornanen

Distribution:
University of Eastern Finland Library / Sales of publications
`julkaisumyynti@uef.fi`
`http://www.uef.fi/kirjasto`

ISBN: 978-952-61-3483-3 (print)
ISSNL: 1798-5668
ISSN: 1798-5668
ISBN: 978-952-61-3484-0 (pdf)
ISSNL: 1798-5668
ISSN: 1798-5676

Author's address:	University of Eastern Finland School of Computing P.O.Box 111 80101 Joensuu Finland email: ville.vestman@uef.fi
Supervisors:	Associate Professor Tomi H. Kinnunen, Ph.D. University of Eastern Finland School of Computing P.O.Box 111 80101 Joensuu Finland email: tomi.kinnunen@uef.fi
	Senior Principal Researcher Kong Aik Lee, Ph.D. NEC Corporation Central Research Laboratories 1753, Shimonumabe, Nakahara, Kawasaki City Kanagawa Prefecture, 211-866 Japan email: kongaik.lee@nec.com
Reviewers:	Associate Professor Jan "Honza" Černocký, Ph.D. Brno University of Technology Department of Computer Graphics and Multimedia Božetěchova 2 612 00 Brno Czech Republic email: cernocky@fit.vut.cz
	Professor Javier Hernando, Ph.D. Polytechnic University of Catalonia Center for Language and Speech Technologies and Applications C/Jordi Girona 1-3 08034 Barcelona Spain email: javier.hernando@upc.edu
Opponents:	Associate Professor Tom Bäckström, Ph.D. Aalto University Department of Signal Processing and Acoustics P.O.Box 12200 00076 Aalto Finland email: tom.backstrom@aalto.fi
	Associate Professor Brian Kan-Wing Mak, Ph.D. The Hong Kong University of Science and Technology Department of Computer Science and Engineering Clear Water Bay, Kowloon Hong Kong email: mak@cse.ust.hk

Ville Vestman
Methods for Fast, Robust, and Secure Speaker Recognition
Joensuu: University of Eastern Finland, 2020
Publications of the University of Eastern Finland
Dissertations in Forestry and Natural Sciences

ABSTRACT

Automatic speaker recognition is used in authentication, surveillance, and forensic applications. Authentication applications use voice as a convenient method to access physical locations or log in to devices. For surveillance purposes, recognition technology can be used, for example, by security agencies to find a criminal by monitoring speech data on telephone networks. In forensic cases, the voices recorded from a crime scene may be analyzed and identified automatically to find clues and evidence relating to the crime. All the above applications benefit from the improvements to speaker recognition technology. The improvements can include increased *speed*, *robustness*, and *security*. Faster recognition systems help create a better user experience in authentication applications and facilitate surveillance by allowing more speech data to be processed in the same amount of time. Robustness helps in all applications by mitigating the detrimental effects caused by variabilities in speech, such as variation caused by recording devices, acoustic environments, and the speaker's health. Finally, the security of speaker recognition must be considered if the technology is used for authentication to minimize the risks associated with malicious use.

This doctoral dissertation presents a versatile selection of studies on the above three topics: speed, robustness, and the security of automatic speaker recognition. The studies include both theoretical and experimental research as well as tutorial-like discussions, challenge organization work (ASVspoof 2019), and science popularization elements. In addition to the studies included in the dissertation, this dissertation offers a technical overview of the selected techniques commonly used in modern speaker recognition systems.

The speed of speaker recognition systems is considered in three studies. The first study compares multiple fast-to-train speaker recognition systems based on dimensionality reduction of *Gaussian mixture model* (GMM) supervectors. The second study deploys one of the compared systems in a web application used for demonstrating speaker recognition technology to the public. This study focuses on *recognition speed* rather than the speed of training. In the last study, the training speed of the well-known i-vector model is improved by performing computations using a *graphics processing unit*.

Likewise, robustness is considered in three studies. The first two studies address the issue of mismatch between speaker enrollment and testing utterances by proposing robust acoustic features. The proposed features, based on *time-varying linear prediction*, reveal promising results in mismatch conditions caused by reverberation and changing the speaking style to whispering. The third study takes a different approach by focusing on the utterance-level features (embeddings) rather than on acoustic features. This study combines the ideas of GMMs, generative i-vector models, and deep neural network-based feature extractors into a so-called *neural i-vector* model.

Finally, the topic of security is also discussed in three studies. These studies consider various kinds of spoofing attacks against automatic speaker verification (ASV) systems. The first one presents the *ASVspoof 2019 challenge* and its results. This challenge evaluated anti-spoofing methods against replayed, converted, and synthesized speech. The second study evaluates the effectiveness of mimicry attacks enhanced using technology-assisted target-speaker selection. The last study proposes so called *worst-case false alarm rate* metric, which can be used to evaluate the potential of technology-assisted target-speaker selection attacks. Additionally, the study proposes a generative model of ASV scores, which allows the estimation of the proposed metric for arbitrary speaker population sizes.

In summary, this dissertation advances and supports speaker recognition research on multiple fronts. It provides some future directions for improving the core technology, and it supports further research on ASV security. It explains the peculiarities of speaker recognition for whispered speech, and it offers ideas on how to design engaging speaker recognition technology demonstrations.

Universal Decimal Classification: *004.8, 004.85, 004.934*
Library of Congress Subject Headings: *Pattern recognition systems; Voice; Speech; Speech processing systems; Identification; Authentication; Machine learning; Neural networks (Computer science)*
Yleinen suomalainen ontologia: *identifiointi; tunnistaminen; todentaminen; verifiointi; puhujantunnistus; hahmontunnistus; tekoäly; koneoppiminen; neuroverkot*

ACKNOWLEDGMENTS

I carried out the research work for this dissertation at the University of Eastern Finland (2016 – 2020) and at the NEC Corporation, Japan (Spring 2019). My work was mostly funded by the Doctoral Programme in Science, Technology, and Computing (SCITECO) of the University of Eastern Finland (UEF). The NEC corporation provided me a generous daily allowance during my 4 month stay at Japan and paid most of the related travel costs. The Academy of Finland (project #309629) funded some of my conference attendances. In addition, I received travel grants from the International Speech Communication Association (ISCA) and IEEE Signal Processing Society (SPS). NVIDIA Corporation donated Titan V and Titan Xp GPUs to our research group, which I used in addition to the computation resources provided by the UEF and NEC.

I was first introduced to speech technology by Associate Professor Tomi Kinnunen in Spring 2015, when I took his Digital Speech Processing course. After that, he encouraged me to first work on a speech-related master's thesis and then continue with the doctoral studies, both of which he supervised. This collaboration has so far resulted in nearly twenty jointly authored publications and it has given me opportunities to work in many national and international collaborations. For making all this possible, I would like to express my sincere gratitude to Associate Professor Tomi Kinnunen.

I would also like to express my gratitude to Dr. Kong Aik Lee, who supervised me during my stay in Japan. His guidance has been extremely valuable during the last two years of the doctoral studies. I am also grateful to all the seventeen co-authors of the publications of this dissertation. Especially, I would like to thank Professor Paavo Alku, Dr. Md Sahidullah, and Dr. Rosa González Hautamäki, who all guided me at the early stages of my studies.

I am thankful to the pre-examiners of the dissertation, Associate Professor Jan "Honza" Černocký and Professor Javier Hernando for their valuable comments to improve the work. I would also like to thank Associate Professors Tom Bäckström and Brian Kan-Wing Mak for agreeing to act as opponents in the public examination of the dissertation.

Next, I would like to express my gratitude to everyone in the computational speech research group of the UEF for creating a positive and encouraging working environment. Also, I am grateful for the support I have received from the other staff members of the department. Special thanks to Juha Hakkarainen and Anssi Kanervisto for keeping our servers running at the UEF.

During these years, I have been lucky to have many friends and fellow sports people around me, who have kept me sane, happy, and fit. Especially, I would like to thank (in alphabetical order) Daniel M., Dinara, Mika K., Mikko H., Mingyue D., Mira Kuura, Prof. Pasi F., Radu, Timo I., and Trung. Thanks also to Sykettä Sports, PörrPörr Symposium, and Louhelan Woima.

Finally, the greatest thanks goes to my parents and siblings, on whose support I could always rely.

Joensuu, August 8, 2020

Ville Vestman

LIST OF PUBLICATIONS

This thesis comprises the present review of the author's work in the field of speaker recognition and the following selection of the author's peer-reviewed publications:

I **V. Vestman**, D. Gowda, M. Sahidullah, P. Alku, and T. Kinnunen, "Time-varying autoregressions for speaker verification in reverberant conditions" *Proc. INTERSPEECH*, Stockholm, Sweden, 1512–1516 (2017).

II **V. Vestman**, D. Gowda, M. Sahidullah, P. Alku, and T. Kinnunen, "Speaker recognition from whispered speech: A tutorial survey and an application of time-varying linear prediction," *Speech Communication*, **99**, 62–79 (2018).

III **V. Vestman** and T. Kinnunen, "Supervector compression strategies to speed up i-vector system development", *Proc. Odyssey: The Speaker and Language Recognition Workshop*, Les Sables d'Olonne, France, 357–364 (2018).

IV **V. Vestman**, B. Soomro, A. Kanervisto, V. Hautamäki, and T. Kinnunen, "Who do I sound like? Showcasing speaker recognition technology by YouTube voice search," *Proc. ICASSP*, Brighton, UK, 5781–5785 (2019).

V **V. Vestman**, K. A. Lee, T. H. Kinnunen, and T. Koshinaka, "Unleashing the unused potential of i-vectors enabled by GPU acceleration," *Proc. INTERSPEECH*, Graz, Austria, 351–355 (2019).

VI M. Todisco, X. Wang, **V. Vestman**, M. Sahidullah, H. Delgado, A. Nautsch, J. Yamagishi, N. Evans, T. Kinnunen, and K. A. Lee, "ASVspoof 2019: future horizons in spoofed and fake audio detection," *Proc. INTERSPEECH*, Graz, Austria, 1008–1012 (2019).

VII **V. Vestman**, T. Kinnunen, R. González Hautamäki, and M. Sahidullah, "Voice mimicry attacks assisted by automatic speaker verification," *Computer Speech & Language*, **59**, 36–54 (2020).

VIII A. Sholokhov, T. Kinnunen, **V. Vestman**, and K. A. Lee, "Voice biometrics security: extrapolating false alarm rate via hierarchical Bayesian modeling of speaker verification scores," *Computer Speech & Language*, **60**, 101024 (2020).

IX **V. Vestman**, K.A. Lee, and T. H. Kinnunen, "Neural i-vectors," *Proc. Odyssey: The Speaker and Language Recognition Workshop*, Tokyo, Japan, 67–74 (2020).

Throughout the overview, these papers are referred to by Roman numerals. During the Ph.D. work, the author was also a contributing author in seven other peer-reviewed publications [1–7] that were not included in this dissertation.

AUTHOR'S CONTRIBUTIONS

The publications I selected for this dissertation are original research papers on speaker recognition. My contributions to each of the papers are explained below.

In Publications **I** and **II**, I developed a new feature extraction method with my co-author D. Gowda. Gowda developed the core part of the new method, whereas I was responsible for deploying the method to the feature extraction pipeline for speaker verification. I conducted most of the experiments and wrote the majority of these two publications. In addition, in Publication **II**, I developed a method for the automatic alignment of normal and whispered speech. All co-authors provided helpful comments during the experimental phase and helped to author the papers.

In Publication **III**, I developed simplifications to speaker recognition systems to speed up system development. I conducted speaker verification experiments and wrote most of the paper. Co-author T. Kinnunen gave helpful insights during the experimentation phase and helped finalize the paper.

For Publication **IV**, I created a browser-based speaker recognition demonstration application built on top of a web platform developed by my co-author B. Soomro with the assistance of A. Kanervisto. For this work, I developed the speaker recognition system, integrated it with the web platform, designed an interface for the web application, conducted most of the experiments, including gathering subjective evaluation data from test users, and wrote the majority of the paper. Co-authors T. Kinnunen and V. Hautamäki helped recruit test users for the subjective experiments. All co-authors helped write the paper.

In Publication **V**, I implemented a fast speaker recognition system using graphics processing unit acceleration to conduct experiments that were previously impractical due to the substantial computational requirements. Co-author K.A. Lee provided helpful insights during the implementation and experimentation, and all co-authors helped finalize the paper, which was mostly written by the main author.

Publication **VI** is a collaborative work done by the organizing committee of the ASVspoof 2019 challenge. My primary contributions were developing the speaker recognition system for the challenge, implementing parts of the evaluation scripts, and helping to the process score files submitted by the challenge participants.

For Publication **VII**, all four authors had nearly equal contributions in studying the threat of voice mimicry to the security of speaker recognition systems. I conducted automatic speaker recognition experiments for the "attacked side" and wrote the majority of the experimental part. Co-author M. Sahidullah conducted experiments for the "attacker side", and R. González Hautamäki collected the speech data, prepared the data for prosody and formant analysis, and performed the perceptual evaluation. The study was initiated and organized by T. Kinnunen, who wrote major parts of the paper.

The idea in Publication **VIII** was developed by the first two authors A. Sholokhov and T. Kinnunen. Overall, the contributions were almost equally split between the first three authors including the undersigned, who conducted all experiments and wrote the experimental part of the article. Co-author K.A. Lee participated in designing the experiments and helped finalize the paper.

For the last publication, Publication **IX**, co-author T. Kinnunen wrote the introduction and conclusions and co-author K.A. Lee wrote minor parts of the paper and created four system design diagrams. In addition, K.A. Lee helped implement the squeeze-and-excite and residual modules used in the study. I implemented the deep embedding extractors and i-vector systems for the study, conducted all experiments, and wrote most of the paper. The idea was jointly developed and refined during the research by all co-authors.

TABLE OF CONTENTS

1 INTRODUCTION

A widespread transition in the field of machine learning is currently occurring — a transition from rule-based methods and shallow statistical models to deep learning with *deep neural networks* (DNNs). The transition has been particularly prevalent in the fields of image [8, 9] and speech processing [10]. Recent advancements in these fields, for example in face [11], speech [12], and speaker recognition [13], are largely a result of developing more powerful deep learning models. The products of this progress have been deployed in consumer applications at a rapid rate [14]. For example, many new smartphones use face recognition to unlock the phone. In addition, translator applications that can recognize spoken words and language automatically, translate speech to another language, and synthesize the translation results to natural-sounding speech are commonly found in phones.

While the enhanced accuracy of modern technologies has enabled a wide variety of new applications, the need for further research continues into the foreseeable future. The problems in machine learning are often not fully solved, but the existing solutions can always be improved. As the techniques improve over time, they can be applied to increasingly more difficult tasks.

An example of a difficult application for machine learning is the use of *biometric identifiers* to identify people. Biometric identifiers are measurable characteristics of humans, such as fingerprints, facial characteristics, heartbeat, or other physiological or behavioral traits [15]. The difficulty of biometric identification is highlighted by the fact that even a well-established fingerprint-matching task can be challenging when encountering partial fingerprints or fingerprints from wet or damaged fingers [15].

The challenges are even more profound with biometric identifiers of voice (*voice biometrics*) because human speech is subject to many nuisance variations [16]. The variability can be caused by background noises, varying acoustic environments, and varying recording devices. While all of these sources of variability (*extrinsic factors*) are independent of the speaker, another set of variabilities originates from the speaker (*intrinsic factors*): The voice can be different when people are ill, tired, or emotional. In addition, voice changes as a result of aging. Furthermore, the voice can be altered *intentionally* [17], for example, when a voice actor plays a character.

As voice is one of the most natural forms of communication between humans, it is also a very convenient method of providing biometric identifiers. The applications of voice biometrics include its use in forensics [18], surveillance [19], authentication [20], and human-to-machine communications [21]. An example of forensic application is the use of technology to identify people at a crime scene based on a voice recording. Voice biometrics can be used for surveillance, for example, by listening to voice communications over the Internet. The potential authentication applications vary from banking scenarios to unlocking a door or phone. Finally, voice biometrics can be used to enrich human-to-machine communications by providing electronic appliances a means to know who is communicating with the device (e.g., Google Home [21]).

1.1 RESEARCH THEMES IN THIS DISSERTATION

The broad focus of this dissertation is on *automatic speaker recognition* [13,16]. Speaker recognition refers to identifying or verifying people's identities from their voices, and the word *automatic* signifies that this is done by a computer rather than a human. The task can further be divided into automatic *speaker identification* (SID) and *automatic speaker verification* (ASV) tasks. The former answers the question 'Who is the speaker?', whereas the latter considers the question 'Is the speaker whom he or she claims to be?'. The practical differences in implementing SID and ASV systems are elaborated in Section 2.1. Another closely related task is *speaker diarization* [22,23]. Speaker diarization is used to determine *who spoke when*, when multiple speakers are present in a speech recording.

The specific research themes of this dissertation are *speed*, *robustness*, and the *security* of the speaker recognition system. One of these themes is considered in each of the publications, **I** through **IX**. The following paragraphs briefly discuss each of the themes in the context of automatic speaker recognition.

The **speed** of speaker recognition systems can be considered from two viewpoints. We can either consider the time required for *training* these systems or the speed of the *deployed* systems in actual use. The two cases are different from each other. During system development, available computing resources are usually ample, but so is the amount of data needed to train the system. However, when the trained system has been deployed, it usually only performs the recognition for one speech recording at a time, but possibly with severely limited computational resources available in the end-user devices. The lack of speed at training time can exhaust the computing resources needed for system optimization, resulting in suboptimal recognition accuracy. The lack of speed in the deployed systems results in compromised user experience.

In general, different approaches exist to accelerate computation. For example, the computational complexity can be reduced by *simplifying* or *approximating* to the machine learning models and algorithms [24,25, **III**]. With this approach, the goal is to have a minimal detrimental effect on recognition accuracy while accelerating computation. Another method to speed up computation is to use more suitable or powerful hardware. A prime example of this is using *graphics processing units* (GPUs) instead of *central processing units* (CPUs) to train DNNs. By taking advantage of the massive parallelism provided by hundreds or thousands of cores in GPUs, networks can potentially be trained up to 50 times faster than with CPUs [26].

As mentioned, the second theme, **robustness**, requires special attention in speech-related applications due to the high level of variability in speech. Diverse sources of variability are specified in Table 1.1. Some of the intrinsic variations in speech (Table 1.1a) can be induced by the conscious effort of the speaker, whereas some variations are more subconscious or inherent in nature. An example of a subconscious voice alteration is the *Lombard effect* [27], in which the speaker subconsciously changes his or her voice to counteract the lack of audibility caused by a noisy environment. Similarly, the speaker's voice can change depending on with whom the speaker is speaking. Further, the speech may not always be regular conversational speech but could be read speech or acted speech. Each of the different situations has manifestations in the characteristics of speech. Furthermore, the voice can be affected by health conditions, emotional states, or levels of mental alertness. Finally, not all variability in voices is detrimental for speaker recognition. While the *within-speaker* sources of variability presented in Table 1.1a often cause challenges,

Table 1.1: Sources of variability in speech. Adapted from [16].

(a) Variability induced by the speaker (intrinsic factors).

• Changes in speaking style (e.g., normal, shouting, or whispering) • Acting • Lexical content of speech • Changing language or accent • Style variation in conversational versus read speech	• Health condition (e.g., cold, Parkinson's disease) • Emotional state (e.g., calm, angry, frightened, or delighted) • Mental alertness • Lombard effect • Style variation based on discussion partner (e.g., adult, child, pet, or machine)

(b) Variability due to extrinsic factors.

Variability in technology	Variability in environment
• Properties of the microphone • Sampling rate and bit rate • Lossy audio formats (e.g., `.mp3`, `.m4a`, and `.ogg`) • Voice enhancement methods (e.g., equalizer or compressor) • Transmission channels (cord, cordless, landline, mobile, and voice over Internet protocol (VoIP))	• Background noise (e.g., traffic noise, babble noise, noise from air conditioning, and wind noise) • Acoustic conditions • Distance to microphone

the *between-speaker* variability of voices, such as the characteristics of vocal tract and articulation, enables speaker recognition in the first place.

The variability that is unrelated to the speaker relates to the environment where the speech was spoken and how the audio was captured, transmitted, processed, and stored (Table 1.1b). Given two distinct environments, sound waves propagate and reflect differently based on the surrounding objects and their materials. Although we, as humans, can still perceive that the sound comes from the same source in both environments, the resulting differences at the signal level can be considerable and cause challenges for machines. Further, different environments can have different background noises. For example, in an office room, the sound of air conditioning is often present, whereas on the street, we can encounter traffic noise. The former of these noises is *stationary* (i.e., it does not change considerably over a period), whereas the latter is *nonstationary*.

Intrinsic and extrinsic variability can be detrimental for automatic speaker recognition in different ways, depending on where in the speech recordings the variability occurs. First, if the speaker recognition system is trained on different types of speech data than what is expected in actual use, the system typically performs poorly because it has not previously encountered the new type of data. This problem is known as *domain mismatch* [28,29]. Second, even if the system is trained with matched data conditions, the task of comparing speakers from two recordings can still be challenging due to the variations between the two recordings.

The nuisance variability in speech strongly influences the accuracy of speaker recognition systems. While studies have demonstrated almost perfect speaker recognition results using laboratory quality data with a small amount of variability [30, 31], the results using more realistic data are considerably worse [4]. Thus, improving the robustness to nuisance variability is essential for making automatic speaker recognition viable for practical application. A typical approach to improving the robustness is to train recognition systems using larger datasets with more variability. Larger datasets with more variable audio allow the statistical models to learn speaker representations better. However, collecting such datasets is often laborious; thus researchers have developed approaches to automatically collect large datasets of speech, including speaker labels, from the Internet [32,33]. Another frequently used strategy to increase the size of training datasets is to *augment data* with altered copies of the original recordings [34,35]. These alterations can be done by adding background noise or by reverberating speech signals to simulate different environments. A complementary strategy is the use of various signal processing approaches to improve the robustness of the systems [36,37, **I**, **II**].

In recent years, automatic speaker recognition has been advancing rapidly, allowing the technology to be adopted in a growing number of applications. However, the adoption rate of innovative technology might be also hindered by **security** issues, especially in high-security applications such as banking [38]. Thus, the third theme of this dissertation, security of speaker recognition, has been gaining momentum recently. The security of speech-related technologies can be compromised in various ways. Figure 1.1 provides selected examples of potential security issues and malicious uses of technology. The first example (a) illustrates a case of a *replay attack* [39]. In principle, it does not require highly technical skills because it only requires the ability to record the target's (victim's) speech followed by playing the recording to the ASV system using a loudspeaker. Figure 1.1b depicts a case of voice conversion attack [40], in which the attacker uses technology to modify his or her voice to sound like the target speaker. By doing so, the attacker could then try to deceive the target's friend (or an ASV system) into thinking that he or she is speaking with the target. In the last example (Figure 1.1c), the attacker uses an ASV system to facilitate deceiving another ASV system [**VII**].

The development of the security of ASV technology has been fostered by biennial ASVspoof challenges (2015, 2017, and 2019) [41,42, **VI**]. These challenges focused on improving the detection of spoofing attacks against ASV systems. The challenges consider two kinds of spoofing attack scenarios known as *logical access* and *physical access*. In the former, the attack audio is injected into the ASV system directly without passing through a microphone. This kind of attack is feasible when attacking automatized phone services because the attacker can redirect the playback output directly to the microphone input, bypassing the need to use a microphone. In contrast, in the physical access scenario, the attacker uses the microphone of the ASV system similarly as depicted in Figure 1.1a. In addition to different access scenarios, ASVspoof challenges consider multiple types of audio for spoofing, including replayed audio and audio generated via voice conversion or speech synthesis technologies. All of these types of spoofed audio can, in principle, be used in both

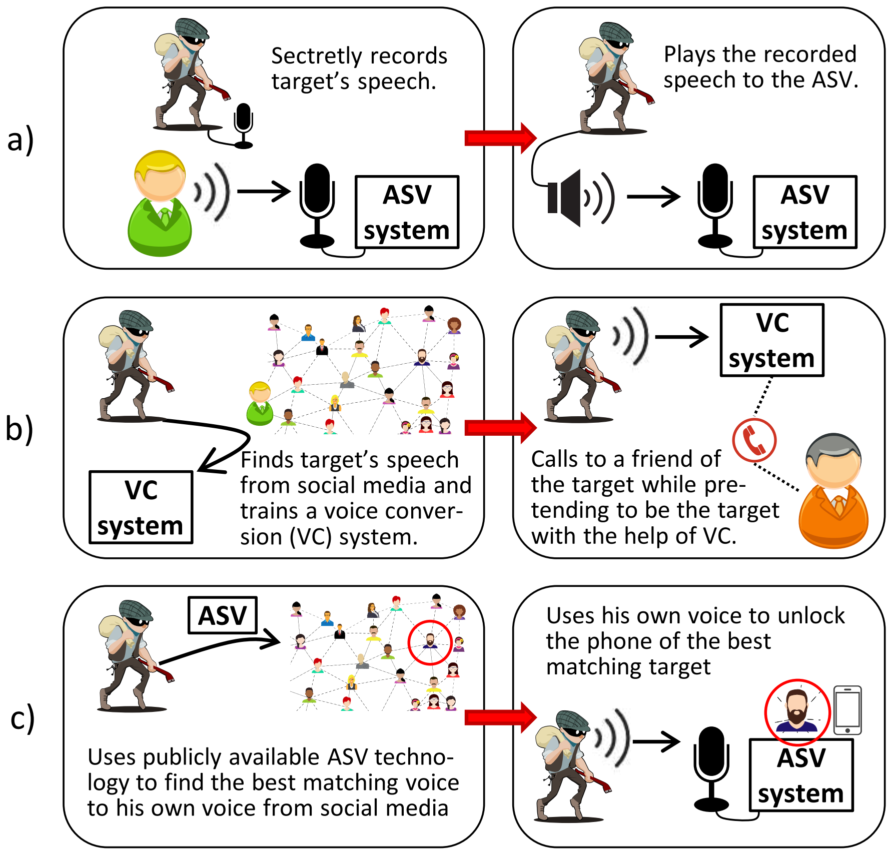

a)
Sectretly records target's speech.
ASV system
Plays the recorded speech to the ASV.
ASV system
b)
VC system
Finds target's speech from social media and trains a voice conversion (VC) system.
VC system
Calls to a friend of the target while pretending to be the target with the help of VC.
c)
ASV
Uses publicly available ASV technology to find the best matching voice to his own voice from social media
Uses his own voice to unlock the phone of the best matching target
ASV system

Figure 1.1: Example scenarios of spoofing attempts to fool speaker verification systems or humans.

logical and physical access attacks. Thus far, the ASVspoof series has considered replayed audio with physical access and voice conversion and speech synthesis with logical access. As a result of the ASVspoof challenges, these attack types are currently the most extensively studied, and several spoofing attack detectors can detect such attacks [43,44].

1.2 LINKING PUBLICATIONS TO RESEARCH THEMES

Table 1.2 connects each of the publications to the selected research themes. The theme of Publications **I** and **II** is robustness, more specifically, robustness to reverberant and whispered speech. Publication **IX** also considers robustness with a focus on developing the core ASV technology. In Publications **III**, **IV**, and **V**, the focus is largely on accelerating either the i-vector system development or online recognition phase. Publication **IV** has the special function of *science popularization*. In this work, a speaker recognition system was deployed into a web app, which was then presented to the public. The remaining theme, the security of ASV technology, is considered in Publications **VI**, **VII**, and **VIII**.

Table 1.2 also illustrates the detailed focus areas and methods used in the publications. These are intended as an advanced overview to offer the overall idea on the topics discussed in the publications. The detailed explanations of the terms and methods are reserved for the upcoming chapters, which are briefly described below.

Chapter 2 explains the most common variants of speaker recognition tasks and provides an overview of speaker recognition system designs. In addition, it presents the discussion of the topics of acoustic feature extraction and system evaluation. Then, Chapter 3 focuses on the probabilistic generative models used in speaker recognition. Next, Chapter 4 presents the DNN models. Finally, Chapter 5 summarizes the publications in this dissertation, and Chapter 6 concludes the work.

Table 1.2: An overview of the publications included in this dissertation.

Publication	Research themes			Focus area	Methods used
	Speed	Robustness	Security		
I		✓		Speaker verification in reverberant conditions	Feature extraction using time-varying linear prediction
II		✓		Speaker recognition from whispered speech, analysis of whispered speech	Time-varying linear prediction, alignment of normal and whispered speech using dynamic time warping (DTW)
III	✓			Use of Gaussian mixture model (GMM) supervectors to speed up ASV development	Variants of probabilistic principal component analysis (PPCA)
IV	✓			Popularization of ASV technology, fast online speaker recognition	Methods of **III** deployed into a web application
V	✓			Optimization of speed and accuracy of i-vector technology	GPU acceleration, minimum divergence re-estimation and other training phase improvements
VI			✓	Anti-spoofing	Evaluation of ASVspoof challenge results
VII			✓	ASV assisted voice impersonation attacks against another ASV	i-vector, x-vector, data collection, perceptual evaluation
VIII			✓	Evaluation of speaker verification performance with large speaker populations	Development of new evaluation metric and model for ASV scores, i-vector, x-vector
IX		✓		Speaker verification by combining discriminative and generative approaches of speaker modeling	Data augmentation, discriminatively trained features, i-vectors

2 FUNDAMENTALS OF SPEAKER RECOGNITION

Speaker recognition tasks come in various forms, and recognition systems can be designed in multiple ways. This chapter describes the most common forms of speaker recognition tasks and presents the overall structure of typical speaker recognition systems. In addition, this chapter discusses *acoustic feature extraction*, a process that transforms input speech waveforms into features that are more suitable for further modeling. Finally, the last section discusses how the performance of speaker recognition systems is evaluated.

2.1 MODES OF SPEAKER RECOGNITION

In both speaker verification and identification, the speakers to be recognized (*target speakers*) must be *enrolled* in the system database before recognizing them. During the enrollment (or registration) phase, the recognition system is provided with a speech sample (*enrollment utterance*) from a target speaker. It is used to create a model (template) for the target speaker. To increase reliability and robustness, it is helpful to use an ample amount of speech in the enrollment (possibly up to a few minutes) [45]. In addition, using multiple enrollment recordings recorded at different times (sessions) can improve performance by increasing robustness toward nuisance variability [46, 47] because different sessions can often contain different sources of variabilities (see Table 1.1).

The key difference between **verification** and **identification** tasks is in how the enrolled speaker models are used during the recognition (testing) phase. In the verification task, one verifies that the speaker is whom he or she claims to be, so the test utterance is only compared against the model of the claimed identity. If the similarity score between the speech sample and the model is high enough, the speaker passes the verification test. In the identification task, the test utterance is compared against all models of the enrolled speakers. The speaker with the highest similarity score is returned as the result (i.e., the system answers the question 'Who is the speaker?'). This is known as *closed-set* speaker identification. In the *open-set* task, the system has the option to declare that no enrolled speakers match the test segment [48].

If a speaker recognition system expects the lexical content of the enrollment and test utterances to match, the system is *text-dependent*. For example, in the enrollment phase, the speaker could be asked to utter a passphrase that he or she must use later on in the testing phase. In the *text-independent* scenario, the lexical content of enrollment and test utterances is not required to match. Out of these two modes, the text-independent mode is more general in its application areas because it does not restrict the lexical content. For the same reason, more data are available for developing of text-independent systems than for text-dependent systems. The benefit of text-dependent speaker verification is that it removes one major source of variability (lexical content), which can lead to higher accuracy. This could be beneficial in applications where high accuracy is preferred over user convenience.

2.2 OVERVIEW OF SPEAKER RECOGNITION SYSTEM DESIGNS

This section provides an overview of common system pipelines for speaker recognition focusing on those used in the publications in this dissertation. These include speaker classifier based on the *Gaussian mixture model – universal background model* (GMM-UBM) [49], the pipeline based on *i-vectors* [50] and probabilistic linear discriminant analysis (PLDA) [51], and the deep learning approach to extract *x-vectors* [35]. The GMM-UBM, i-vector, and PLDA models are discussed in more detail in Chapter 3, and Chapter 4 reviews the deep learning models.

The above system pipelines start with an *acoustic feature extraction* step, as depicted in Figure 2.1. Feature extraction converts time-domain audio waveforms into sequences of *acoustic feature vectors*. Typically, each of the feature vectors contains information about a short segment of the original waveform. The extraction process of these short-time features often involves transforming short-time segments through *Fourier transform*. This and other common feature extraction approaches are described in more detail in the next section.

In the i-vector and x-vector pipelines, the basic idea is to transform variable-length sequences of acoustic feature vectors into fixed and compact sized vectors so that these vectors contain an ample amount of speaker-related information while minimizing statistical redundancies. The common term to describe such a vector is *speaker embedding*. Being vectors, the comparison of the enrollment and test embeddings can be as simple as computing the angle between the two vectors (the *cosine similarity measure*) [52]. However, the most successful embedding comparator (or *classifier*) in recent years has been PLDA, a generative probabilistic model discussed later in this dissertation.

In Figure 2.1, the i-vector and x-vector pipelines fall under Design 1, whereas the GMM-UBM classifier follows Design 2. In Design 2, a statistical model is fitted to the feature vectors extracted from the enrollment utterance to create a speaker model. This model is often not trained from scratch but instead is adapted from the (UBM) [49]. The UBM is trained using a large pool of speakers and utterances, which are not part of enrollment or test data. In the GMM-UBM design, the UBM serves as a common anchor to all speaker models and acts as an alternative hypothesis model in *likelihood ratio testing*, in which the following hypotheses are considered:

$$\begin{cases} H_0 : \text{Test utterance } u \text{ is from speaker } s, \\ H_1 : \text{Test utterance } u \text{ is not from speaker } s. \end{cases}$$

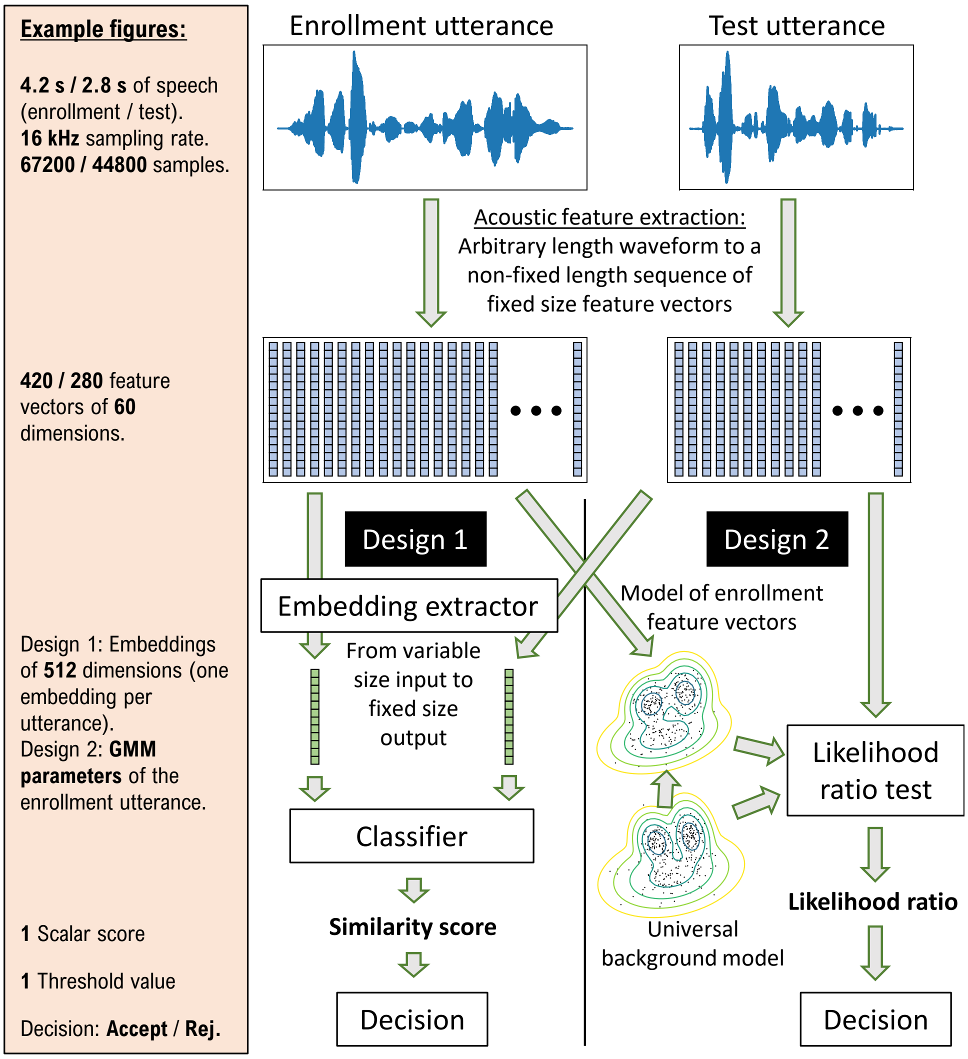
The likelihood ratio is computed as follows:

$$\mathrm{LR} = \frac{p(u|H_0)}{p(u|H_1)} = \frac{p(u|\theta_s)}{p(u|\theta_{\mathrm{UBM}})}, \tag{2.1}$$

where the null hypothesis H_0 is represented by the enrollment model defined by the parameters θ_s and the alternative hypothesis H_1 is represented by the UBM defined by the parameters θ_{UBM}. The quantities $p(u|\theta_s)$ and $p(u|\theta_{\mathrm{UBM}})$ are the probability density functions for the given models evaluated for the test utterance u [49].

The raw output from speaker recognition systems is a similarity score $s \in \mathbb{R}$. The similarity score is commonly, but not necessarily, a logarithm of the likelihood ratio (both GMM-UBM and PLDA provide it as an output). The reason for taking the logarithm of the likelihood ratio is that it makes the computation of joint likelihoods and Gaussian densities more convenient and numerically stable [53, p. 26]. Applying the logarithm does not affect the order of scores as the logarithm is a strictly increasing function and hence it does not affect the recognition performance.

In the verification setting, the output score is compared against a decision threshold $\lambda \in \mathbb{R}$. If $s > \lambda$, then the verification trial is accepted; and otherwise, it is rejected. In closed-set identification, the threshold is not needed because the scores of all enrolled speakers are compared with each other and the speaker with the maximum score is selected.



Example figures:
4.2 s / 2.8 s of speech (enrollment / test).
16 kHz sampling rate.
67200 / 44800 samples.
420 / 280 feature vectors of 60 dimensions.
Design 1: Embeddings of 512 dimensions (one embedding per utterance).
Design 2: GMM parameters of the enrollment utterance.
1 Scalar score
1 Threshold value
Decision: Accept / Rej.
Enrollment utterance
Test utterance
Acoustic feature extraction:
Arbitrary length waveform to a non-fixed length sequence of fixed size feature vectors
Design 1
Design 2
Embedding extractor
From variable size input to fixed size output
Classifier
Similarity score
Decision
Model of enrollment feature vectors
Universal background model
Likelihood ratio test
Likelihood ratio
Decision

Figure 2.1: Examples of common speaker verification system designs. Design 1 is seen in the i-vector and x-vector systems, whereas Design 2 represents the GMM-UBM approach.

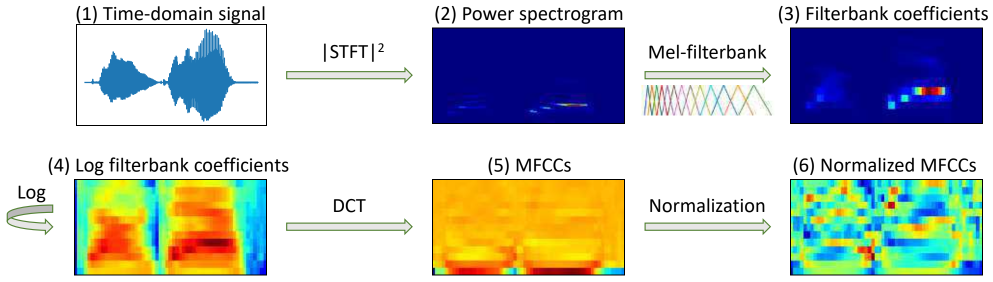


Figure 2.2: The process of computing *mel frequency cepstral coefficients* (MFCCs): First, a power spectrogram is obtained via the *short-time Fourier transform* (STFT). Second, a mel-scaled filterbank is applied. Third, the resulting filterbank coefficients are log-compressed. Fourth, the discrete cosine transform (DCT) is applied to obtain MFCCs. Finally, the MFCCs are normalized to have zero mean and unit variance over time.

2.3 ACOUSTIC FEATURE EXTRACTION FROM SPEECH SIGNALS

Figure 2.1 illustrates that two types of features are often present in speaker verification systems: (1) short-time features (also referred to as *acoustic features*), which are typically extracted from 20 or 25 ms long speech segments (*frames*); and (2) embeddings, which represent information extracted from the whole utterance in fixed-size format. The short-time features are often obtained as a result of rule-based (or handcrafted) processes, whereas the computation of the latter usually relies on statistical models and machine learning. This section focuses on the short-time features by presenting the most used techniques for acoustic feature extraction.

Mel frequency cepstral coefficients (MFCCs) [54] have been the most frequently used acoustic features for speaker recognition tasks. They have served as the standard baseline in most speech feature-oriented research papers in the past several decades. The success of MFCC features stems from their ability to perform well in various applications, while being relatively computationally inexpensive and easy to implement. The computation scheme of MFCCs is presented in Figure 2.2. In addition to MFCCs, many other feature extraction schemes have a similar computational pipeline. Various parts of this pipeline are explained in the following subsections.

2.3.1 Speech preprocessing and speech activity detection

Before feature extraction, the time-domain input signal is usually preprocessed. Some of the commonly used preprocessing steps include removing *direct current* (DC) offset, normalizing the signal's maximum amplitude to a fixed value, and *pre-emphasis filtering* [55]. Pre-emphasis flattens the typically low-frequency dominated speech spectra by emphasizing high frequencies. Whether this is beneficial depends on the data, feature extraction method, and statistical model.

Then, as the input signal may contain portions that do not contain speech, such as silence or noise, it is beneficial to detect and discard these portions of the input signal. Therefore, most speaker recognition systems incorporate a *speech activity detector* (SAD) [56, 57] that aims at removing nonspeech frames. The removal of nonspeech frames in ideal, noise-free conditions is relatively easy because one can simply use the signal energy computed from a speech frame as an indicator of silence. If the energy is less than a specified threshold, the frame is considered to be silence; otherwise, it is considered speech. As the overall energy levels of different recordings can vary, it is beneficial to incorporate an adaptive threshold setting strategy. An example is shown in [13], where SAD sets the threshold based on the energy of the highest energy frame. The detection of frames without speech or silence, such as frames corrupted with a considerable amount of background noise, is much more challenging because energy is no longer a reliable indicator for speech activity. Therefore, many systems rely on energy-based SAD and instead mitigate the detrimental effects of non-speech (e.g., noise) frames in the other parts of the system, such as in embedding extractor or in PLDA. Nevertheless, many sophisticated SADs exist, such as the ones that use GMM [57] or DNN models [58] to address the issue at the SAD stage.

2.3.2 Speech spectrum estimation

After preprocessing, the first step in the feature extraction process is to compute a specific time-frequency representation, the *spectrogram*. In the case of MFCCs, the spectrogram is obtained as follows. First, the signal is split into overlapping frames that typically have a duration of 20 or 25 ms with an overlap of 10 or 15 ms between consecutive frames. Second, the frames are processed using a *window function* that tapers the values near the endpoints of the frames toward zero. The windowing benefits in the third step, which applies the *discrete Fourier transform* (DFT) [59, p. 99] to each frame by reducing *spectral leakage* [60]. Spectral leakage is an effect that causes the energy of a frequency to 'leak' into other frequency bins in DFT presentation. The effect is due to the frequency components not being periodic in the observation window. The effect is mitigated by windowing, which lessens the abrupt discontinuities at the end points of the frames.

When combined, the above three steps (framing, windowing, and DFT), are known as the *short-time Fourier transform* (STFT) [61, p. 81]. The fourth step is to take a magnitude of the complex-valued DFT outputs to obtain a *magnitude spectrum* of each frame. These spectra are then squared to obtain the *power spectra*, or a *power spectrogram*. An example of a power spectrogram is shown in the second panel of Figure 2.2. Using a linear scale, the visualization of a power spectrogram appears quite uninformative. The visualization quality can be improved by adding a logarithmic transform, as observed in the fourth panel of the figure.

Although STFT is commonly used to produce the time-frequency representation of a signal, several alternative methods are available [55, 62, 63]. One of the most prominent approaches for obtaining speech spectra (and spectrograms) is *linear prediction* (LP) [55]. The LP technique is adopted in features, such as *linear prediction cepstral coefficients* (LPCC) [54], *frequency domain linear prediction* (FDLP) features [64], *2-D autoregressive* (2-D AR) features [65], and the new features proposed in Publications **I** and **II**.

In LP modeling [55], the current sample of a speech frame $x[n]$, $n = 1, \ldots, L$, is predicted as a linear combination of the past p samples [55]. That is,

$$\hat{x}[n] = \sum_{k=1}^{p} a_k x[n-k], \tag{2.2}$$

where the real-valued coefficients a_k are known as *predictor coefficients*. Predictor coefficients are typically found by minimizing the mean squared error of the error signal $e[n] = x[n] - \hat{x}[n]$. Following the *autocorrelation method* of LP [30,55] this leads to solving a set of *normal equations* given as follows:

$$\sum_{k=1}^{p} a_k r_{|i-k|} = r_i, \quad i = 1, \dots, p, \tag{2.3}$$

where

$$r_i = \sum_{n=1}^{L} x[n]x[n-i] \tag{2.4}$$

are known as *autocorrelation coefficients*. Here, $x[n]$ is assumed to be zero when $n < 1$. The solved predictor coefficients can be used to estimate the envelope of the speech spectrum $X[m]$ as follows [30,55]:

$$X[m] = \frac{G}{\left|1 - \sum_{k=1}^{p} a_k e^{-\mathrm{i}2\pi mk/L}\right|}, \quad m = 0, \dots, L-1,$$

where G is the *gain coefficient*, which can be computed as follows:

$$G = r_0 - \sum_{k=1}^{p} a_k r_k.$$

The above spectrum estimation is performed independently for each frame (L denotes the number of samples per frame). The model order p can be used to control the amount of detail in the spectral estimate. By increasing model order p, the LP spectrum can be made arbitrarily close to the corresponding DFT spectrum [55].

2.3.3 Filterbanks

Once the spectral representation is obtained, the next step is to apply a *filterbank*. A filterbank defines a set of filters used to compute signal energies in the frequency bands defined by the filters. The application of a filterbank serves two primary purposes. First, it allows *frequency warping* by placing more filters in certain frequency bands than in the others. Second, it reduces the dimensionality of spectral data obtained from DFT or LP analysis. The filterbank used in the computation of MFCCs consists of triangular bandpass filters spaced according to the mel-scale [66]. The mel-scale is a logarithmic scale based on perceptual studies that measured equal pitch differences in different frequencies. As humans are more sensitive to pitch differences at low frequencies, the mel-scaled filterbank places filters more densely in the low-frequency bands. The mel-scaled filterbank may be replaced by filters with different shapes and scales in other feature extraction approaches. For example, by spacing filters linearly, one obtains *linear frequency cepstral coefficients* (LFCCs) [54]. It is also possible to use a linear filterbank to compute MFCCs if the warping of the frequency axis is already included in the DFT, as done in [67]. Finally, there are cases when the filterbank is not applied at all. An example of this is the extraction of LPCC features, where the *linear prediction coefficients* (LPCs) are converted directly to LPCCs, as described in [68].

2.3.4 Cepstral features and feature normalization

The filterbank outputs are converted to cepstral coefficients by applying logarithmic compression followed by the *discrete cosine transform* (DCT). The logarithmic nonlinearity can be replaced with an other suitable nonlinear function. For example, *power-normalized cepstral coefficients* (PNCCs) [69] use the 15th root compression, and *perceptual linear prediction* (PLP) features [70] use cubic-root compression instead of log compression. The DCT operation following the logarithmic compression decorrelates the log filterbank signal [71]. The decorrelation process removes redundancies, allowing more a compact presentation. Thus, the last DCT coefficients are often discarded from further processing.

As the final step of acoustic feature extraction, cepstral features are usually normalized to suppress the influence of convolutional noise (such as reverberation or the variation induced by differences in microphones [72]). A convolution in the time domain corresponds to multiplication in the spectral domain and addition in the cepstral (i.e., log spectral) domain. To this end, the MFCCs are normalized by subtracting the mean MFCC vector (computed over time) from all MFCC feature vectors. This operation is known as *cepstral mean subtraction* (CMS). If the MFCCs are further divided by their standard deviations, the operation is known as *cepstral mean and variance normalization* (CMVN). Finally, instead of normalizing the features using the statistics of the full utterance, the means and standard deviations are often computed using a sliding window centered on the processed frame [73,74]. This makes the normalization more adaptive to varying conditions within an utterance.

2.3.5 Delta features

Not shown in Figure 2.2 is another commonly performed step: The MFCCs are often appended with their *delta* (**Δ**) and *delta-delta* (**ΔΔ**) features. While the MFCC base coefficients described above provide a 'snapshot' of speech properties in a given frame, the delta features capture information about the dynamics of speech (i.e., how the speech changes from frame to frame). Delta features have been effectively used in speaker recognition systems of the past few decades with similar constructs dating back at least to the early 1980s [75].

A simple way (one method among many [76, p. 98]) to compute delta features for a frame at time index t is as follows:

$$\boldsymbol{\Delta}_t = \boldsymbol{m}_{t+1} - \boldsymbol{m}_{t-1},$$

where $\boldsymbol{m}_t$ is a vector containing MFCCs for the frame at time t. Similarly, delta-delta features describing the dynamics of the delta features can be obtained as follows:

$$\boldsymbol{\Delta\Delta}_t = \boldsymbol{\Delta}_{t+1} - \boldsymbol{\Delta}_{t-1}.$$

If calculated as above, the feature vector formed from MFCCs appended with deltas and delta-deltas contain information from not just one frame but five consecutive frames. Note that the above delta features can be computed with a convolution filter $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$. Due to the convolutional nature of delta features, their utility in deep learning-based speaker recognition (e.g., x-vector [35]) is questionable because *convolutional neural networks* are readily designed to model the changes between consecutive frames. This lessened importance of delta features is reflected in the recent neural network based state-of-the-art systems [77, 78], where delta features may no longer be used. In contrast, the delta features are used by default in GMM-UBM and i-vector-based speaker recognition systems [2].

2.4 DATASETS, EVALUATIONS, AND METRICS

Most speaker recognition systems are designed either for 8 kHz (*narrowband*) or 16 kHz (*wideband*) speech data. From the Nyquist-Shannon sampling theorem [79], it follows that these sampling rates can be used to reconstruct signals with frequencies of up to 4 kHz and 8 kHz, respectively. Frequencies of up to 4 kHz are enough to convey the most energy content in speech. However, the clarity of some high-frequency consonants can be impaired with the 4 kHz bandwidth limit [80, p. 63]. The narrowband 8 kHz sampling rate is often used in telephone speech transmission. The 16 kHz wideband speech can convey very high-quality speech and is often used with the *voice over Internet protocol* (VoIP). Speaker recognition datasets commonly contain either of the above two types of speech data.

Today, speaker recognition is largely based on using data-hungry machine learning methods. Thus, the availability of speech data is crucial both for training and evaluating speaker recognition systems. A speech dataset is well suited for speaker recognition research if it contains speaker labels, has a numerous speakers, and contains multiple utterances and recording sessions per speaker. In the past, such datasets were not readily available; thus, researchers often used small self-collected datasets. In recent years, the situation has improved, and many large publicly available speaker recognition datasets are available. A few examples of these are VoxCeleb [81, 82], *Speakers in the Wild* (SITW) [83], RedDots [84], and RSR2015 [85] datasets.

In addition to the better availability of the datasets, speaker recognition research has been pushed forward by numerous open speaker recognition evaluations (or challenges) [86–90]. In these evaluations, research teams from different countries and organizations submit their speaker recognition scores for a task specified by the challenge organizers. This facilitates the meaningful comparison of different speaker recognition technologies because every participant must obey common challenge rules and use a common dataset. The most prominent challenge organizer over the years has been the *National Institute of Standards and Technology* (NIST), which has been organizing speaker recognition challenges almost yearly since 1996 [88, 91]. Recently, many other community-driven challenges have taken place as well. Some examples of these are the VoxCeleb Speaker Recognition Challenge (VoxSRC) 2019 [89], SITW evaluation [90], Voices from a Distance Challenge 2019 [92], Short-duration Speaker Verification Challenge 2020 [93], and the ASVspoof 2019 Challenge (Publication **VI**). All of these challenges have motivated researchers to push the limits of their systems, which has driven the performance of speaker recognition systems forward.

Speaker verification systems are evaluated using a set of *evaluation trials*. Each trial consists of the enrollment identifier and test segment identifier. The enrollment identifier specifies the speaker model created at the enrollment stage. The test segment identifier can point to a recording from the same speaker as the enrolled speaker or from a different speaker. These two types of trials are called *target* and *non-target trials*, respectively. In the system evaluation phase, each trial is independently processed (*scored*) by the speaker verification system. A high score value indicates that the trial is likely a target trial, while a low score is likely a non-target trial. In speaker recognition challenges, the ground-truth labels are not given to participants beforehand. Instead, the participants are asked to send their scores to the organizers, who use the ground-truth labels to compute the performance metrics. This prevents participants from overfitting their systems to the evaluation trial list.

Besides defining common audio data and common evaluation trials, the third and equally important design aspect concerns the choice of performance metrics. In the field of ASV, several established evaluation metrics have been adopted by the research community. Perhaps the most common performance metrics are the *equal error rate* (EER) and the *detection cost* obtained from the *detection cost function* (DCF). The former is a non-parametric metric that does not require setting a decision threshold or any other parameters. The latter involves multiple parameter settings, as is discussed below.

For any given decision threshold λ, one can compute the corresponding rates of *false alarm* (or *false acceptance*) and *false rejection* (or *miss*). The rates of false acceptance (P_{fa}) and miss (P_{miss}) are given as follows:

$$P_{\text{fa}}(\lambda) = \frac{1}{|S_{\text{non}}|} \sum_{s \in S_{\text{non}}} \mathbb{I}(s > \lambda) \qquad \text{and} \qquad P_{\text{miss}}(\lambda) = \frac{1}{|S_{\text{tar}}|} \sum_{s \in S_{\text{tar}}} \mathbb{I}(s < \lambda),$$

where $\mathbb{I}(\cdot)$ is the indicator function that outputs 1 if the comparison in brackets is true, and 0 otherwise. The sets S_{tar} and S_{non} contain scores for target trials and non-target trials, respectively, and $|\cdot|$ denotes the total number of trials in a given set. By increasing the detection threshold λ, the false acceptance rate P_{fa} decreases, and the miss rate P_{miss} increases. The EER is defined as the rate at which $P_{\text{fa}}(\lambda) = P_{\text{miss}}(\lambda)$. In practice, the score sets are often such that the above equation does not exactly hold with any detection threshold. In such cases, one can search for a threshold that provides the smallest difference between $P_{\text{fa}}(\lambda)$ and $P_{\text{miss}}(\lambda)$ and compute the average of these two values.

The performance of a system can be visualized by drawing the *detection error tradeoff* (DET) curve [94], which is obtained by plotting the miss rate against the false alarm rate at different thresholds. The axes of the DET curve are scaled using a normal deviate scale. Figure 2.3 presents examples of the DET curves.

Unlike the nonparametric EER metric, the DCF can be manually adjusted for a specific application. For some applications, the user-convenience (low miss rates) can be more important than the the security (low false alarm rates), and vice versa. The adjustability is achieved using three control parameters. These are the prior probability of the target speaker (P_{tar}) and the costs of falsely accepting a nontarget speaker (C_{fa}) and missing a target speaker (C_{miss}). Table 2.1 lists examples of DCF settings in two different scenarios. The first scenario is representative of a surveillance-type application, in which the prior probability of the target speaker among a larger population is low. Thus, P_{tar} is set to 0.01. The second scenario considers an access control system, whose users are assumed to be well-intentioned. This assumption favors the use of a high P_{tar} value of 0.99. The associated risks of falsely accepting a malicious user are high, which supports the use of the high cost value of $C_{\text{fa}} = 10$.

The detection cost for a specific threshold value λ is computed as follows [86]:

$$C_{\text{det}}(\lambda) = P_{\text{tar}} C_{\text{miss}} P_{\text{miss}}(\lambda) + (1 - P_{\text{tar}}) C_{\text{fa}} P_{\text{fa}}(\lambda).$$

As the costs of the DCF can be arbitrary positive values, the resulting detection cost values can be difficult to interpret. Thus, the detection cost is normalized with the *default detection cost* defined as follows:

$$C_{\text{default}} = \min \begin{cases} P_{\text{tar}} C_{\text{miss}} \\ (1 - P_{\text{tar}}) C_{\text{fa}}. \end{cases}$$

The default cost represents a "dummy" system, which either accepts or rejects all trials (whichever leads to a lower cost). The following normalized cost indicates that the evaluated system is better than the dummy system if the cost is less than 1:

$$C_{\text{norm}}(\lambda) = \frac{C_{\text{det}}(\lambda)}{C_{\text{default}}}.$$

Finally, to evaluate the system without fixing the threshold, the minimum of the normalized detection cost (minDCF) can be computed as follows:

$$C_{\text{min}} = \min_{\lambda}(C_{\text{norm}}(\lambda)).$$

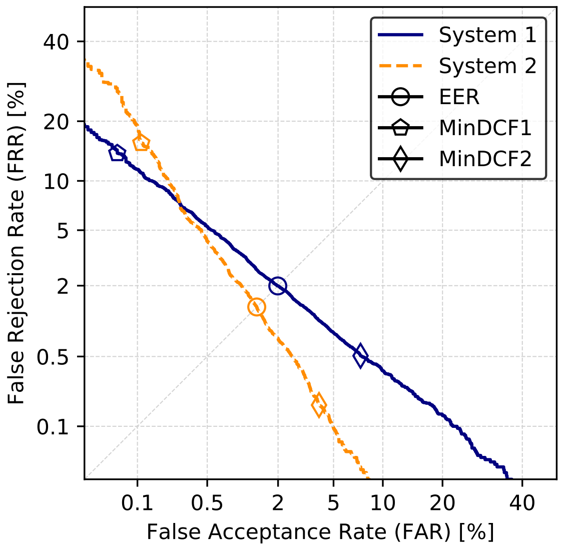


Figure 2.3: Examples of detection error tradeoff (DET) curves. System 1 has better performance at low false acceptance (false alarm) rates, while System 2 performs better at low false rejection (miss) rates. The figure displays EER points and the points determined by the minDCF metric using two different parameter settings. MinDCF1 and minDCF2 correspond to the surveillance and access control scenarios in Table 2.1, respectively.

Table 2.1: Examples of detection cost function (DCF) control parameters for surveillance and access control applications.

	C_{miss}	C_{fa}	P_{tar}
Surveillance scenario	1	1	0.01
Access control scenario	1	10	0.99

3 SPEAKER RECOGNITION WITH PROBABILISTIC GENERATIVE MODELS

Probabilistic generative models are *probabilistic* because they involve random variables and probability distributions. They are *generative* because they describe the generation process of the observed data given the target variable [95]. This contrasts with *discriminative* models that model the target variable given the observed data.

This chapter presents a selected set of probabilistic generative models commonly used in speaker recognition systems.

3.1 GAUSSIAN MIXTURE MODELS

The *Gaussian mixture model* (GMM) has been one of the cornerstones of speaker recognition systems since the 1990s. Of the most successful speaker recognition systems, only some deep learning-based systems, such as the x-vector, do not use GMMs or GMM-inspired constructs. Even if the x-vector has been the state-of-the-art system for the last few years, the GMM ideology has not been abandoned, as DNN layers that resemble GMMs have been studied in recent work [78, 96, 97, **IX**] with good results.

The GMMs have been used in diverse ways for speaker recognition, not just as a GMM-UBM classifier or DNN layer. For example, the the GMM assumptions are built into the i-vector and joint factor analysis approaches [98], which are discussed in detail in Section 3.4, and GMMs have been also used with support vector machines (SVMs) [99] and probabilistic principal component analysis (PPCA) [100, **III**]. The following subsections cover the basics of Gaussian mixture modeling of speech.

3.1.1 Multivariate Gaussian distribution

Let $\mathbf{X}$ be a continuous d-dimensional random vector following a *multivariate Gaussian* (i.e., *normal*) distribution. The probability density function of $\boldsymbol{X}$ is then given by the following:

$$p(\mathbf{X} = \boldsymbol{x}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\theta}) \equiv \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})},$$

where the parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are the *mean vector* ($\boldsymbol{\mu}$) and *covariance matrix* ($\boldsymbol{\Sigma}$) of the multivariate Gaussian distribution [101, p. 46]. The sign '$\equiv$' means "equal by definition". If the random variable is clear from the context, the following notation may be used:

$$p(\mathbf{X} = \boldsymbol{x}|\boldsymbol{\theta}) = p(\boldsymbol{x}|\boldsymbol{\theta}).$$

In the context of speaker recognition and machine learning in general, we are often interested in fitting a multivariate Gaussian distribution to a given sequence of *independent* observations (feature vectors), $\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$. The independence assumption of observations is useful in deriving formulas for model fitting using probabilistic machinery. However, this assumption tends not to hold in practice because feature vectors extracted from different frames of the same utterance are not independent due to temporal dependencies in speech. That is, on a scale of 5 to 100 ms, the speech signal is rather stationary [61] so that the consecutive speech frames can be highly correlated. Although the independence assumption may not be completely satisfied in practice, the formulas derived using stricter assumptions can often be applied with satisfactory results in practical settings.

One way to estimate parameters $\boldsymbol{\theta}$ is through a *maximum likelihood* (ML) estimation:

$$\boldsymbol{\theta}_{\text{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N | \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \prod_{n=1}^{N} p(\boldsymbol{x}_n | \boldsymbol{\theta}) \tag{3.1}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n | \boldsymbol{\theta}) \tag{3.2}$$

The maximization can be done [101, pp. 99–100] by setting

$$\frac{\partial g}{\partial \boldsymbol{\mu}} = \mathbf{0} \quad \text{and} \quad \frac{\partial g}{\partial \boldsymbol{\Sigma}} = \mathbf{0},$$

where $g(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n | \boldsymbol{\theta})$. This results in the following:

$$\hat{\boldsymbol{\mu}}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \quad \text{and} \quad \hat{\boldsymbol{\Sigma}}_{\text{ML}} = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_{\text{ML}})(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_{\text{ML}})^{\mathsf{T}}.$$

The result reveals that the ML estimates of the mean and covariance parameters are obtained by computing the sample mean and covariance of the data.

Another commonly used strategy of fitting the parameters is the *maximum a posteriori* (MAP) estimation [101, p. 149]. The MAP estimation can be considered a more general form of ML estimation that uses prior information about the model parameters in addition to the observations. That is, the model parameters are treated as random variables, which is the basic idea in *Bayesian statistics* [101, p. 191]. To explain the idea further, it is helpful to introduce *Bayes' theorem* [53, p. 15]:

$$p(\boldsymbol{\theta} | \mathcal{D}) = \frac{p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})}.$$

Here, the terms $p(\boldsymbol{\theta}|\mathcal{D})$, $p(\mathcal{D}|\boldsymbol{\theta})$, p($\boldsymbol{\theta}$), and $p(\mathcal{D})$ are called *posterior* of $\boldsymbol{\theta}$, *likelihood* of $\boldsymbol{\theta}$, *prior* of $\boldsymbol{\theta}$, and *evidence*, respectively. The evidence is a normalization constant that is needed to ensure that the posterior distribution integrates to one.

In ML estimation, only the likelihood is maximized, whereas in MAP, the posterior is maximized:

$$\boldsymbol{\theta}_{\text{MAP}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta} | \mathcal{D}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \frac{p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta}). \tag{3.3}$$

In the above maximization, the evidence $p(\mathcal{D})$ can be neglected because it does not depend on $\boldsymbol{\theta}$. As a result, maximization (3.3) differs from (3.1) only in that (3.3) has the prior $p(\boldsymbol{\theta})$. The prior distribution is a distribution of the parameters of the data distribution and has its own parameters $\boldsymbol{\theta}_{\text{prior}}$. The parameters $\boldsymbol{\theta}_{\text{prior}}$ of the prior distribution $p(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{\theta}_{\text{prior}})$ are called *hyperparameters*. The prior distribution reflects the prior beliefs or knowledge on how the data should be distributed. In this sense, it can be regarded as a *regularizer* [101, pp. 149, 206] for the plain ML estimation, preventing unexpected fitted distributions, which could occur as a result of insufficient observations $\mathcal{D}$ in terms of quantity or quality. Moreover, if the prior distribution does not indicate any preference in the choice of parameters $\boldsymbol{\theta}$ (that is, $p(\boldsymbol{\theta})$ is a uniform distribution), the MAP estimate (3.3) reduces to the ML estimate (3.1).

As discussed in [102], an appropriate choice of prior distribution can simplify the MAP estimation process of parameters. To this end, it is common to use the *conjugate prior* of the likelihood function as a prior. The prior distribution is a conjugate prior if the prior and posterior distributions are from the same family of distributions [101, p. 74]. In the case of the multivariate Gaussian likelihood function defined by the mean and covariance, the conjugate prior is the *normal-inverse-Wishart* (NIW) distribution, which has the following density:

$$\text{NIW}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{\mu}_0, \tau, \boldsymbol{\Psi}, \nu\right) = \mathcal{N}\left(\boldsymbol{\mu}\middle|\boldsymbol{\mu}_0, \frac{1}{\tau}\boldsymbol{\Sigma}\right)\mathcal{W}^{-1}\left(\boldsymbol{\Sigma}|\boldsymbol{\Psi}, \nu\right), \tag{3.4}$$

where $\mathcal{W}^{-1}$ is the probability density function for the *inverse-Wishart* distribution [101, p. 127, Eq. (4.165)]. The hyperparameters $\boldsymbol{\mu}_0, \tau, \boldsymbol{\Psi}$, and ν define the underlying normal and inverse Wishart distributions. To sample values from NIW, one first samples the covariance matrix from the inverse-Wishart distribution, and then the sampled covariance (scaled by $1/\tau$) is used to sample the mean vector from the normal distribution.

It can be shown [102] that with NIW prior the MAP estimates of parameters are given as follows:

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{\text{MAP}} &= \frac{\tau\boldsymbol{\mu}_0 + N\hat{\boldsymbol{\mu}}_{\text{ML}}}{\tau + N}, \\ \hat{\boldsymbol{\Sigma}}_{\text{MAP}} &= \frac{\boldsymbol{\Psi} + N\hat{\boldsymbol{\Sigma}}_{\text{ML}} + \frac{\tau N}{\tau+N}(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}_{\text{ML}})(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}_{\text{ML}})^{\mathsf{T}}}{\nu - d + N}.\end{aligned} \tag{3.5}$$

It is straightforward to verify that the limits of $\hat{\boldsymbol{\mu}}_{\text{MAP}}$ and $\hat{\boldsymbol{\Sigma}}_{\text{MAP}}$ as N approaches infinity are the ML estimates $\hat{\boldsymbol{\mu}}_{\text{ML}}$ and $\hat{\boldsymbol{\Sigma}}_{\text{ML}}$, respectively. That is, as the number of observations increases, the ML and MAP estimates better agree with each other.

3.1.2 Gaussian mixture model

The previous section demonstrated how to fit a single Gaussian distribution to the observed feature vectors. However, the distribution of acoustic feature vectors tends not to be a unimodal Gaussian distribution; different phones of speech have different spectral representations, and the differences also occur in acoustic features. Therefore, it may be better to model each phone or speech sound with its own Gaussian distribution instead of using a single Gaussian distribution only. Such an approach combines multiple Gaussian distributions into a *mixture model*.

A GMM [101, pp. 339] of C components can be presented as $\boldsymbol{\theta} = \{w_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\}_{c=1}^{C}$, where w_c, $\boldsymbol{\mu}_c$, and $\boldsymbol{\Sigma}_c$ are the mixing weight, mean vector, and covariance matrix of component c, respectively. The mixing weights $w_c (c = 1, \ldots, C)$ are non-negative and sum to one so that the following density function integrates correctly to one:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{c=1}^{C} w_c \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c).$$

To sample a vector from a GMM, one first samples a component index c from a *categorical distribution* defined by the mixing weights w_c and then samples a vector from $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$.

When trying to compute an ML estimate of GMM parameters for the given data $\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, the main difficulty is that the components to which each observation belongs are not known [101, pp. 348–349]. *If* these *component assignments* were known, the ML estimation could be simply done for each component following the above-presented approach for a single Gaussian distribution. Instead, the assignments can be presented with the *latent* (hidden/unobserved) variables $z_n \in \{1, \ldots, C\}$, where z_n is the assignment for observation $\boldsymbol{x}_n$. The posterior distributions of variables z_n contain probabilities that can be regarded as "soft" assignments, which are useful in the parameter estimation process that is briefly explained below.

The ML estimates of GMM parameters are typically obtained using the *expectation-maximization* (EM) algorithm [103]. The EM algorithm is an iterative algorithm guaranteed to monotonically increase the likelihood value after every iteration. Each iteration of the algorithm consists of two steps, the E and M steps. First, in the E step, the soft component assignments are computed using the GMM parameters from previous iteration $\boldsymbol{\theta}^{(t-1)}$:

$$\gamma_{n,c} = P(z_n = c|\boldsymbol{x}_n, \boldsymbol{\theta}^{(t-1)}) = \frac{p(\boldsymbol{x}_n|z_n = c, \boldsymbol{\theta}^{(t-1)}) P(z_n = c|\boldsymbol{\theta}^{(t-1)})}{p(\boldsymbol{x}_n|\boldsymbol{\theta}^{(t-1)})} \tag{3.6}$$

$$= \frac{p(\boldsymbol{x}_n|z_n = c, \boldsymbol{\theta}^{(t-1)}) P(z_n = c|\boldsymbol{\theta}^{(t-1)})}{\sum_{i=1}^{C} p(\boldsymbol{x}_n|z_n = i, \boldsymbol{\theta}^{(t-1)}) P(z_n = i|\boldsymbol{\theta}^{(t-1)})} \tag{3.7}$$

$$= \frac{w_c^{(t-1)} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c^{(t-1)}, \boldsymbol{\Sigma}_c^{(t-1)})}{\sum_{i=1}^{C} w_i^{(t-1)} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_i^{(t-1)}, \boldsymbol{\Sigma}_i^{(t-1)})}. \tag{3.8}$$

Here, (3.6) follows from Bayes' theorem, the expression for evidence in (3.7) follows from the fact that the posterior probabilities $P(z_n = c|\boldsymbol{x}_n, \boldsymbol{\theta}^{(t-1)})$ sum up to one, and finally, (3.8) follows from the definitions of the likelihood and prior.

Then, in the M step, the GMM parameters $\boldsymbol{\theta}$ are updated. The update based on maximization of log-likelihood

$$p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{c=1}^{C} w_c \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{3.9}$$

is difficult as the logarithm cannot be moved inside the inner sum [101, p. 349]. Thus, the *complete data* log-likelihood $\log p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, z_1, z_2, \ldots, z_N|\boldsymbol{\theta})$ is considered instead. However, this cannot be directly computed because latent variables z_n are not observed. Thus, the *expectation* with respect to the old parameters $\boldsymbol{\theta}^{(t-1)}$ of the complete data log-likelihood is maximized instead. The expected complete data log-likelihood is given as follows [101, p. 351]:

$$\begin{aligned}\mathbb{E}\left[\log p(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N, z_1, z_2, \dots, z_N|\boldsymbol{\theta})\right] &= \mathbb{E}\left[\sum_{n=1}^{N} \log p(\boldsymbol{x}_n, z_n|\boldsymbol{\theta})\right] \\ &= \sum_{n=1}^{N}\sum_{c=1}^{C} \mathbb{E}[\mathbb{I}_c(z_n)\log(w_c\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)] \\ &= \sum_{n=1}^{N}\sum_{c=1}^{C} \gamma_{n,c}\log(w_c\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)],\end{aligned}$$

where $\mathbb{I}_c(z)$ is an indicator function that returns 1 if $z = c$ and 0 otherwise. The maximization is done by computing the partial derivatives with respect to the parameters $w_c, \boldsymbol{\mu}_c$, and $\boldsymbol{\Sigma}_c$ and setting them to zero. The solution must also satisfy the constraint that the weights w_c sum to one. The maximization leads to the following update equations [101, p. 351]:

$$\begin{aligned}w_c &= \frac{1}{n}N_c, \\ \boldsymbol{\mu}_c &= \frac{\boldsymbol{f}_c}{N_c}, \\ \boldsymbol{\Sigma}_c &= \frac{\boldsymbol{S}_c}{N_c} - \boldsymbol{\mu}_c\boldsymbol{\mu}_c^{\mathsf{T}},\end{aligned}$$

where N_c, $\boldsymbol{f}_c$, and $\boldsymbol{S}_c$ are known as the *Baum-Welch statistics* [104], and are defined as follows:

$$N_c = \sum_{n=1}^{N} \gamma_{n,c}, \tag{3.10}$$

$$\boldsymbol{f}_c = \sum_{n=1}^{N} \gamma_{n,c}\boldsymbol{x}_n, \tag{3.11}$$

$$\boldsymbol{S}_c = \sum_{n=1}^{N} \gamma_{n,c}\boldsymbol{x}_n\boldsymbol{x}_n^{\mathsf{T}}.$$

The EM parameter estimation process repeats the above two steps until stopped. The iteration can be terminated when the relative increase in the log-likelihood (3.9) across consecutive iterations falls below a predefined threshold, for example.

3.1.3 The universal background model approach for speaker adaptation

This section describes the GMM-UBM framework [49] for speaker recognition. The backbone of the GMM-UBM framework is the UBM, which is trained using a large volume of speech data, including numerous speakers and utterances. The UBM captures a wide range of variabilities in speech and serves as the base model for adapting speaker-specific models using the enrollment data. In the testing phase, the UBM serves as an alternative hypothesis model, whereas the speaker-specific model works as the null hypothesis model in likelihood ratio testing.

The enrollment relies on the MAP estimation of parameters, for which the prior is obtained from the UBM, and the observations are from the enrollment data. In other words, the speaker specific parameters are *adapted* from the UBM parameters. The MAP adaptation could be done for all parameters of GMM (i.e., weights, means, and covariances) [49], but the common practice is to adapt only the means using (3.5) because adapting weights and covariances has not been found to be beneficial [49].

Consider the following example of a MAP adaptation to compute a speaker model. Let $\boldsymbol{\theta}_{\text{UBM}} = \{w_c^{\text{ubm}}, \boldsymbol{\mu}_c^{\text{ubm}}, \boldsymbol{\Sigma}_c^{\text{ubm}}\}_{c=1}^{C}$ be the UBM, and let $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ be the feature vectors computed from the enrollment utterance of a speaker. Then, the speaker model is obtained as $\boldsymbol{\theta}_s = \{w_c^{\text{ubm}}, \boldsymbol{\mu}_c^{\text{adapted}}, \boldsymbol{\Sigma}_c^{\text{ubm}}\}_{c=1}^{C}$, where

$$\boldsymbol{\mu}_c^{\text{adapted}} = \frac{\tau \boldsymbol{\mu}_c^{\text{ubm}} + \sum_{n=1}^{N} \gamma_{n,c} \boldsymbol{x}_n}{\tau + N_c}.$$

Here, $\sum_{n=1}^{N} \gamma_{n,c} \boldsymbol{x}_n$ corresponds to $N\hat{\boldsymbol{\mu}}_{\text{ML}}$ in (3.5), the difference being that the GMM version of the adaptation uses the soft assignments $\gamma_{n,c}$ computed with respect to the UBM. The tunable parameter $\tau \geq 0$ is known as the *relevance factor*, which controls how much weight is given to the prior information. A larger value of τ results in more weight for the prior (i.e., UBM) in the adaptation. If $\tau = 0$, the adapted model reduces to the ML estimate, meaning that the prior does not affect the adaptation at all.

The similarity score between a test utterance represented with feature vectors $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M)$ and a speaker model $\boldsymbol{\theta}_s$ can be obtained as the ratio of the log-likelihoods of the form (3.9) between the speaker model and UBM:

$$\text{score} = \log \frac{p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M | \boldsymbol{\theta}_s)}{p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M | \boldsymbol{\theta}_{\text{UBM}})}$$

The obtained log likelihood ratio can then be compared to the decision threshold of the system to accept or reject the verification trial.

3.1.4 Gaussian mixture model supervectors

While the GMM-UBM system in Section 3.1.3 follows Design 2 of Figure 2.1, the GMM-based modeling also enables multiple ways to build systems that follow Design 1. The key characteristic in Design 1 is that both the enrollment and test utterances are presented using fixed-length embeddings. In the GMM-UBM approach, the fixed-length embeddings could be constructed, for example, by concatenating the MAP-adapted mean vectors together to obtain a high-dimensional fixed-size vector called a *supervector* [99]:

$$\boldsymbol{m} = \begin{bmatrix} \mu_1^{\text{adapted}} \\ \mu_2^{\text{adapted}} \\ \vdots \\ \mu_C^{\text{adapted}} \end{bmatrix}.$$

The dimensionality of supervectors can be very large because it is common to have GMMs with 512 to 4096 components and feature spaces with 30 to 90 dimensions. For example, a 2048-component GMM with 60-dimensional feature vectors results in 122 880-dimensional supervectors.

The classifier may be either built for high-dimensional data or lower-dimensional data obtained through a dimensionality reduction. Examples of the former are the use of the SVM classifier for the supervectors [99] and the use of PLDA in high-dimensional space [105]. The latter approaches include PPCA (for dimensionality reduction) [100, **III**] followed by PLDA and the use of i-vector embedding extractor [50] with PLDA. In the following sections, the focus is limited to the latter approaches because the i-vector like pipelines have been more successful in the 2010s. Furthermore, the former approaches have not been used in the publications of this dissertation.

3.2 PROBABILISTIC PRINCIPAL COMPONENT ANALYSIS

In GMM-UBM, the acoustic feature space is modeled using a mixture of multiple Gaussian distributions due to the complex nature of acoustic feature distributions. The features in the embedding spaces, however, are commonly modeled using a single Gaussian distribution. Restricting a model to a single Gaussian distribution can be advantageous because the embeddings are already a result of statistical modeling; thus they are likely to have a simpler distribution.

One of the single-Gaussian models is the PPCA [106], a probabilistic version of the classic PCA. The PPCA model can be used to encode high-dimensional supervectors to low-dimensional latent vectors. Let $\boldsymbol{z}_n \in \mathbb{R}^e$ $(n = 1, \ldots, N)$ be the latent vector representations of the supervectors $\boldsymbol{m}_n \in \mathbb{R}^d$ $(n = 1, \ldots, N)$ with the assumption that prior of the latent vectors is standard normal; that is,

$$p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n | \boldsymbol{0}, \boldsymbol{I}).$$

Then, the PPCA model defined by parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{V}, \sigma^2\}$ is Gaussian

$$p(\boldsymbol{m}_n | \boldsymbol{z}_n, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{m}_n | \boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{z}_n, \sigma^2 \boldsymbol{I}), \tag{3.12}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$, $\boldsymbol{V} \in \mathbb{R}^{d \times e}$, and $\sigma^2 > 0$.

The PPCA model can be trained using the EM-algorithm with the parameter update equations given in [53, p. 578]. Given the trained model with parameters $\boldsymbol{\theta}$, the posterior distributions of the latent vectors are given as follows:

$$p(\boldsymbol{z}_n | \boldsymbol{m}_n, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\phi}, \boldsymbol{\Psi}),$$

where

$$\boldsymbol{\Psi} = (\boldsymbol{I} + \frac{1}{\sigma^2} \boldsymbol{V}\boldsymbol{V}^\mathsf{T})^{-1} \qquad \text{and} \tag{3.13}$$

$$\boldsymbol{\phi}_n = \frac{1}{\sigma^2} \boldsymbol{\Psi} \boldsymbol{V}^\mathsf{T} (\boldsymbol{m}_n - \boldsymbol{\mu}). \tag{3.14}$$

The mean $\boldsymbol{\phi}_n$ of the posterior distribution is used as a lower-dimensional representation of the supervector $\boldsymbol{m}_n$. This kind of representation is used as a speaker embedding in Publication **III**. Moreover, PPCA has been also used to estimate the *total variability model* presented in the Section 3.4 [100].

A close relative to the PPCA is *factor analysis* [53, pp. 583–586], which differs from PPCA in that, instead of the isotropic covariance matrix $\sigma^2 \boldsymbol{I}$ in (3.12), the factor analysis model has a diagonal covariance matrix.

3.3 PROBABILISTIC LINEAR DISCRIMINANT ANALYSIS

A PLDA model can be used to compute the speaker similarity score for a pair of embeddings. As opposed to *unsupervised* GMM and PPCA models, the PLDA is trained in a *supervised* manner by using the speaker label of each embedding. The labels allow the same latent speaker representation to be shared between the utterances of the same speaker. Thus, the latent representation is enforced to capture information common to all utterances; that is, clues about a speaker's identity.

The PLDA model has multiple variants [107]: the *original* [108], *simplified* [109], *two-covariance* [110], and *heavy-tailed* PLDA [111]. The first three variants have the same model structure, the first one is the most constrained in terms of degrees of freedom, whereas the third one is the least constrained [107] [112, p. 35]. In the fourth variant, heavy-tailed PLDA, the latent vectors are assumed to have a *Students' t* prior instead of a Gaussian prior. In the following paragraphs, the focus is on the simplified PLDA, which is among the most commonly used variants in speaker recognition. It was also used in Publications **III**, **IV**, **V**, **VII**, and **IX** of this dissertation.

The simplified PLDA model $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{V}, \boldsymbol{\Sigma}\}$ is defined for e-dimensional embeddings $\boldsymbol{\phi}_n$, $n = 1, \ldots, N$, from the same speaker as follows:

$$p(\boldsymbol{\phi}_n | \boldsymbol{z}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{z}, \boldsymbol{\Sigma}), \tag{3.15}$$

where the f-dimensional latent vector $\boldsymbol{z}$ has a prior $\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$. The model differs from the PPCA model (3.12) in that the latent vector $\boldsymbol{z}$ is shared among all embeddings $\boldsymbol{\phi}_n$ and that the covariance matrix $\boldsymbol{\Sigma}$ is full, rather than isotropic.

Similar to other latent variable models, the PLDA model is commonly trained using the EM-algorithm [107]. The trained PLDA model can be used to compute the log-likelihood score between the enrollment ($\boldsymbol{\phi}_\mathrm{e}$) and test embeddings ($\boldsymbol{\phi}_\mathrm{t}$) as follows:

$$\text{score} = \log \frac{p(\boldsymbol{\phi}_\mathrm{e}, \boldsymbol{\phi}_\mathrm{t} | \boldsymbol{\theta}, H_0)}{p(\boldsymbol{\phi}_\mathrm{e}, \boldsymbol{\phi}_\mathrm{t} | \boldsymbol{\theta}, H_1)}, \tag{3.16}$$

where H_0 is the hypothesis, in which $\boldsymbol{\phi}_\mathrm{e}$ and $\boldsymbol{\phi}_\mathrm{t}$ share the same latent vector, whereas the latent vectors are not shared in the hypothesis H_1. To compute the above log-likelihood ratio, it is necessary to marginalize (3.15) over the latent vectors [108] to obtain the following joint likelihoods:

$$p(\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N | \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{V}'\boldsymbol{V}'^{\mathsf{T}} + \boldsymbol{\Sigma}'), \tag{3.17}$$

where

$$\boldsymbol{\mu}' = \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu} \\ \vdots \\ \boldsymbol{\mu} \end{bmatrix}_{eN \times 1}, \qquad \boldsymbol{V}' = \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{V} \\ \vdots \\ \boldsymbol{V} \end{bmatrix}_{eN \times f}, \qquad \boldsymbol{\Sigma}' = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma} & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{\Sigma} \end{bmatrix}_{eN \times eN}.$$

Note that $\boldsymbol{V}'$ differs from its counterpart in [108] because of the differences between the original and simplified forms of PLDA.

Now, the score in (3.16) can be written as follows:

$$\text{score} = \log \frac{\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\phi}_\mathrm{e}\\ \boldsymbol{\phi}_\mathrm{t}\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_t & \boldsymbol{\Sigma}_b\\ \boldsymbol{\Sigma}_b & \boldsymbol{\Sigma}_t\end{bmatrix}\right)}{\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\phi}_\mathrm{e}\\ \boldsymbol{\phi}_\mathrm{t}\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_t & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_t\end{bmatrix}\right)}, \tag{3.18}$$

where the total and between-class covariance matrices are given by $\boldsymbol{\Sigma}_t = \boldsymbol{V}\boldsymbol{V}^\mathsf{T} + \boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_b = \boldsymbol{V}\boldsymbol{V}^\mathsf{T}$, respectively. Here, the numerator is obtained directly from (3.17), and the denominator follows from the independence of two embeddings with distinct latent representations.

Finally, by setting $\boldsymbol{\mu} = 0$ (by shifting the embedding space by $-\boldsymbol{\mu}$), by expanding the Gaussian densities in (3.18), applying the logarithm, and computing the inverses of the symmetric block matrices, the score (3.18) can be written in the following form [109]:

$$\text{score} = \boldsymbol{\phi}_\mathrm{e}^\mathsf{T}\boldsymbol{Q}\boldsymbol{\phi}_\mathrm{e} + \boldsymbol{\phi}_\mathrm{t}^\mathsf{T}\boldsymbol{Q}\boldsymbol{\phi}_\mathrm{t} + 2\boldsymbol{\phi}_\mathrm{t}^\mathsf{T}\boldsymbol{P}\boldsymbol{\phi}_\mathrm{e} + \texttt{constant},$$

where

$$\boldsymbol{Q} = \boldsymbol{\Sigma}_t^{-1} - (\boldsymbol{\Sigma}_t - \boldsymbol{\Sigma}_b\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}_b)^{-1},$$
$$\boldsymbol{P} = \boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}_b(\boldsymbol{\Sigma}_t - \boldsymbol{\Sigma}_b\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\Sigma}_b)^{-1}.$$

To use the above scoring method without training the PLDA model parameters $\boldsymbol{\mu}$, $\boldsymbol{V}$, and $\boldsymbol{\Sigma}$ using the EM-algorithm, one can directly compute the total and between-class covariance matrices from the training data. This strategy was used in Publications **V** and **IX**.

3.4 MULTI-GAUSSIAN FACTOR ANALYSIS

When combined, the computation of the MAP-adapted GMM supervectors (Section 3.1.4), supervector compression using PPCA (Section 3.2), and PLDA scoring (Section 3.3) form the speaker verification pipeline used in Publication **III**. There are two drawbacks with this approach, which can be tackled using alternative approaches presented below. These drawbacks are: First, after computing the MAP-adapted supervectors, the *uncertainty* caused by short-duration recordings is not propagated to the following steps. This is evident from (3.13), where all the utterances have the same posterior covariance matrix regardless of the length of the utterance. The covariance matrix (3.13) reflects the uncertainty in the embedding (3.14). Second, the computation of MAP-adapted supervectors has one extra tunable parameter τ, which is not needed in the following approach.

One way to mitigate the above drawbacks is to create a factor analysis model that operates on acoustic features directly instead of GMM supervectors. This approach creates two additional challenges compared to the simple case presented in Section 3.2. First, instead of having just a single supervector per utterance, one utterance has a varied number of acoustic feature vectors (frames). Second, it is not sufficient to model the acoustic feature space using a single Gaussian distribution only. Addressing these challenges leads to a similar approach to that taken in the PLDA model. In the PLDA, multiple embeddings from the same speaker share the same latent variable. In this case, the requirement is the ability to present multiple frames *and* Gaussian components using the same latent variable. The idea of *tying* the same latent variable across multiple frames and Gaussian components is emphasized in [113] and [114].

The model with the tying of frames and Gaussian components was first presented in [115] and was further elaborated in [50]. To be exact, this model, known as the *total variability model*, was obtained as a modification of the *joint factor analysis* (JFA) [98]. The JFA model has separate latent vectors for the speaker and channel effects, whereas the total variability model captures both effects in a single latent vector.

Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ be feature vectors extracted from an utterance, and let $\boldsymbol{z}$ be the shared latent vector with the standard normal prior. Then, the total variability model defined by parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_c, \boldsymbol{T}_c, \boldsymbol{\Sigma}_c\}_{c=1}^{C}$ can be presented as follows:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N | \boldsymbol{z}, \boldsymbol{\theta}) = \prod_{c=1}^{C} \prod_{\boldsymbol{x} \in X_c} \mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_c + \boldsymbol{T}_c \boldsymbol{z}, \boldsymbol{\Sigma}_c),$$

where X_c contains feature vectors from $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ that are aligned to component c. Here, each Gaussian component c is represented with a mean vector $\boldsymbol{\mu}_c$, projection matrix $\boldsymbol{T}_c$, and covariance matrix $\boldsymbol{\Sigma}_c$. The same latent vector $\boldsymbol{z}$ is shared across all Gaussian components.

Given the total variability model $\boldsymbol{\theta}$, the posterior distribution of the latent vector is obtained as [116]

$$p(\boldsymbol{z} | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N, \boldsymbol{\theta}) = \mathcal{N}(z | \boldsymbol{\phi}, \boldsymbol{\Psi}),$$

where

$$\begin{aligned}
\boldsymbol{\Psi} &= \left(\boldsymbol{I} + \sum_{c=1}^{C} \widehat{N}_c \boldsymbol{T}_c^{\mathsf{T}} \boldsymbol{\Sigma}_c^{-1} \boldsymbol{T}_c \right)^{-1}, \\
\boldsymbol{\phi} &= \boldsymbol{\Psi} \sum_{c=1}^{C} \boldsymbol{T}_c^{\mathsf{T}} \boldsymbol{\Sigma}_c^{-1} (\hat{\boldsymbol{f}}_c - \widehat{N}_c \boldsymbol{\mu}_c), && (3.19) \\
\widehat{N}_c &= |X_c| \quad \text{(the number of feature vectors in } X_c\text{)}, && (3.20) \\
\hat{\boldsymbol{f}}_c &= \sum_{\boldsymbol{x} \in X_c} \boldsymbol{x}. && (3.21)
\end{aligned}$$

Similar to GMM training, it is not known to which component each frame belongs. Therefore, the computation of *sufficient statistics* (3.20) and (3.21) is impossible. Instead, we can use the Baum-Welch statistics (3.10) and (3.11) computed with respect to the UBM to replace the above sufficient statistics.

The vectors of the form in (3.19) computed using Baum-Welch statistics are known as *i-vectors*, which are most commonly used with a PLDA back-end. The i-vector-based speaker recognition systems were state-of-the-art roughly from 2010 to 2017. In the late 2010s, DNN embeddings started to outperform i-vectors [117].

4 SPEAKER RECOGNITION WITH DEEP NEURAL NETWORKS

This chapter discusses the topic of *deep neural networks* (DNNs) [26,118,119] and their applications in speaker recognition. The focus is on core concepts of DNNs, such as *computational graphs*, *loss functions*, *automatic differentiation* [120], and *gradient-based parameter optimization*. These concepts are the key to understanding how neural networks are trained regardless of the application. The last section of the chapter then focuses on the use cases in speaker recognition.

4.1 NEURAL NETWORKS AS COMPUTATIONAL GRAPHS

Neural networks are computational models comprising numerous elementary processing operations. Multiple elementary operations can be composed to form arbitrarily complicated expressions. The computation of these expressions can be represented using *computational graphs*. The following example is loosely based on the presentation in [120] and elucidates this concept. Let $f\colon \mathbb{R}^2 \to \mathbb{R}$, $g\colon \mathbb{R}^2 \to \mathbb{R}^2$, and $h\colon \mathbb{R}^2 \to \mathbb{R}^2$ be functions defined by the following equations

$$
\begin{aligned}
f(\boldsymbol{x}) &= \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + d = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + d, \\
g(\boldsymbol{x}) &= \begin{bmatrix} \tanh(x_1) \\ \tanh(x_2) \end{bmatrix}, \text{ and} \\
h(\boldsymbol{x}) &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}.
\end{aligned}
$$

Then, a composite function $f \circ g \circ h\colon \mathbb{R}^2 \to \mathbb{R}$, where $y = (f \circ g \circ h)(\boldsymbol{x}) = f(g(h(\boldsymbol{x})))$, can be presented with a computational graph, which is displayed in Figure 4.1. This graph can be considered a simple neural network with x_1 and x_2 as the *input variables*, $v_1, \ldots, v_{13}$ as the *hidden variables*, and y as the *output variable*. The remaining nine nodes (a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2, c_1, c_2, d) represent the (learnable) *parameters* of the network. The output node and each hidden node of the graph are associated with an elementary operation. In the example graph, these operations are either multiplications, additions, or evaluations of a hyperbolic tangent.

The computational graph in Figure 4.1 is an example of a *feedforward neural network*. Feedforward networks do not have *cycles*, and the computation flows only in one direction, from the input to output [119, p. 163]. Another type of network suited for *sequentially* organized input data, such as speech signals, is the *recurrent neural network* (RNN). In RNNs, the computation for the current time step may use variables from the previous time step [119, p. 368]. The temporal connectedness allows an RNN to have internal memory, so that computation at any time step benefits

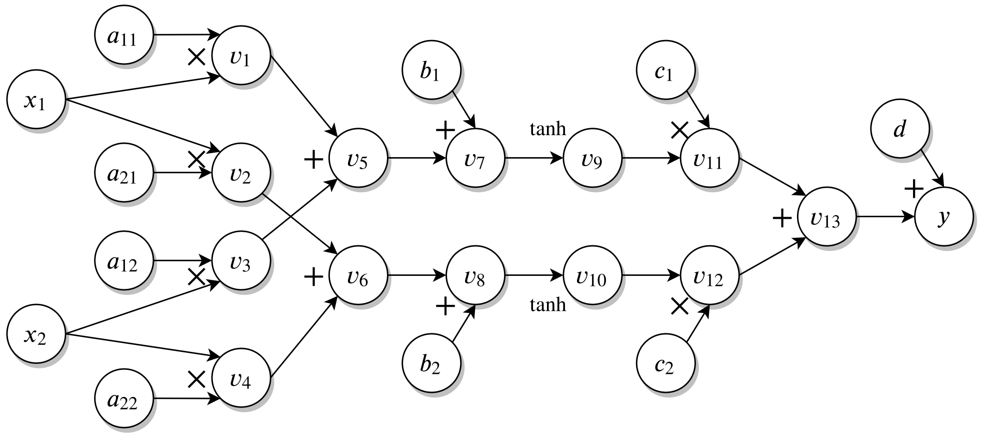


Figure 4.1: A computational graph representing the composite function $f \circ g \circ h$ defined at the beginning of Section 4.1.

from all the previous time steps. In the following sections, the focus is on feedforward networks as they have been used in Publications **VI**, **VII**, **VIII**, and **IX** of this dissertation.

Although the computational graph in Figure 4.1 consists of seven layers of operations, the associated feedforward network is considered to have only three layers. These are the *input layer* consisting of x_1 and x_2, the *output layer* consisting of y, and one *hidden layer* consisting of v_9 and v_{10}. The convention is that one layer is formed by a linear transformation followed by a nonlinearity (called an *activation function*). In the example, the linear transformation h with the activation g gives the output for the hidden layer, whereas the second linear transformation f provides the output y for the output layer.

Activation functions, such as the *hyperbolic tangent* (tanh), play a key role in increasing the modeling power of DNNs by adding nonlinearities. Indeed, without the activation function, the network in Figure 4.1 could be presented as a single linear transformation: $(f \circ h)(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) = (\boldsymbol{c}^\mathsf{T}\boldsymbol{A})\boldsymbol{x} + (\boldsymbol{c}^\mathsf{T}\boldsymbol{b} + d)$. In addition to the tanh activation used in the example, some of the other widely adopted activation functions include the *rectified linear unit* (ReLU) [121], *sigmoid* function [119, p. 65], and *softmax* function [119, p. 179].

4.2 LOSS FUNCTIONS

Loss functions are used to measure the error between the network outputs and target output values. For example, consider that the network in Figure 4.1 is used to estimate the angle of the input vector $\boldsymbol{x}$, which could be computed using the two-argument arcus tangent (arctan2). Instead of using the arcus tangent directly, the task is to obtain a good estimate of the angle using only the operations defined in the computational graph. To train the network parameters for this task, the output of the network must be compared to the correct value of the angle with a suitable loss function. Here, such a loss function could be the squared error $e(y, y_{\text{tar}}) = (y - y_{\text{tar}})^2$ between the output value y and the correct target value $y_{\text{tar}} = \text{arctan2}(\boldsymbol{x})$.

Figure 4.2 illustrates a version of the computational graph presented in Figure 4.1

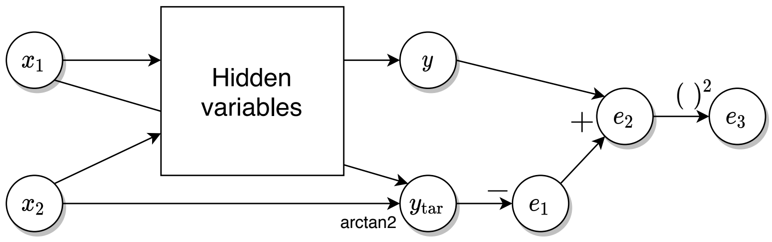


Figure 4.2: A simplified presentation of a computational graph presented in Figure 4.1 with squared error (loss) computation.

with loss function operations added to the graph. Nodes e_1, e_2, and e_3 represent the loss computation from the output value y and target value y_{tar}. Typically, y_{tar} is not computed from the input values but is provided as a part of the labeled testing set.

In the case of the above example, the output is scalar-valued, and the squared error was used as the loss. In the case of *vector-valued regression* tasks, it is common to use the *mean squared error* (MSE) loss given as follows:

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y}, \boldsymbol{y}_{\text{tar}}) = \frac{1}{D}\sum_{i=1}^{D}(y_i - (y_{\text{tar}})_i)^2,$$

where D is the number of dimensions in the output vector. Then, for *classification* tasks with B classes, it is common to use *cross-entropy* (CE) loss with the following strategy. First, the output layer of the network is constructed to have one node per class (node y_i for class i). Second, the outputs are activated using the *softmax* [119, p. 179] function

$$\sigma(\boldsymbol{y})_i = \frac{e^{y_i}}{\sum_{n=1}^{B} e^{y_n}}$$

to obtain a valid probability mass function representing probabilities for different classes. Finally, the CE loss is computed between the softmax outputs $\sigma(\boldsymbol{y})_i$ and *one-hot encoded* class label $\boldsymbol{y}_{\text{tar}}$ as follows:

$$\mathcal{L}_{\text{CE}}(\sigma(\boldsymbol{y}), \boldsymbol{y}_{\text{tar}}) = -\sum_{j=1}^{B}(y_{\text{tar}})_j \log(\sigma(\boldsymbol{y})_j).$$

Because $\boldsymbol{y}$ is one-hot encoded, only the element at the index of the target class is 1 and the rest of the values are 0. Therefore, the above loss simplifies to the following:

$$\mathcal{L}_{\text{CE}}(\sigma(\boldsymbol{y}), c) = -\log(\sigma(\boldsymbol{y})_c),$$

where c is the index of the target class.

As noted in [119, pp. 181–182], the name of the above softmax function is misleading. The function represents a "soft" version of the *argmax* function instead of the max function. Whereas the ordinary argmax provides an index of the maximum value, which could be then represented with a one-hot vector, the softmax (or *softargmax*) provides a soft, continuous-valued version of the one-hot vector. Another name for softmax is the *normalized exponential* function [53, p. 198] because the softmax outputs are normalized to sum to one.

4.3 AUTOMATIC DIFFERENTIATION

The process of feeding the input data to the network and executing the operations to compute the loss is called a *forward pass*. After computing the loss, the network parameters are updated so that the loss becomes smaller. This is commonly achieved using gradient-based optimization methods that require computations of partial derivatives of the loss function with respect to every network parameter. To this end, the network is traversed backward (*backward pass*) to compute the partial derivatives.

In practice, the partial derivatives are computed using *automatic differentiation* (AD) [120], which is a method to compute the exact values of partial derivatives at specific points without constructing complete analytic expressions for the partial derivatives. This differs from *symbolic differentiation*, where the expressions are constructed, and from *numerical differentiation*, which only approximates the partial derivatives at specific points. For further discussion of the differences between automatic, symbolic, and numerical differentiation, the reader is pointed to [120].

Moreover, AD has two variants: *forward mode* [122] and *reverse mode* [123, pp. 24–32]. Reverse mode is used in neural network training because it better suits cases where the number of parameters is large, and the output is a scalar value (such as the loss value in neural networks) [120]. Therefore, the following discussion considers only the reverse-mode differentiation with a scalar-valued output.

During the forward pass, the values obtained for all variables are stored for use during the backward pass of reverse-mode AD. The backward pass starts at the output layer, where the first partial derivative $\dot{e}_2$ (for the graph in Figure 4.2) is given as follows:

$$\dot{e}_2 = \frac{\partial e_3}{\partial e_2} = 2e_2.$$

Then, the process continues by traversing the network backward with the help of the *chain rule* of differentiation to obtain the following:

$$\begin{aligned}
\dot{y} &= \frac{\partial e_3}{\partial y} = \frac{\partial e_3}{\partial e_2}\frac{\partial e_2}{\partial y} = \dot{e}_2\frac{\partial e_2}{\partial y} = \dot{e}_2 \times 1,\\
\dot{d} &= \frac{\partial e_3}{\partial d} = \frac{\partial e_3}{\partial e_2}\frac{\partial e_2}{\partial y}\frac{\partial y}{\partial d} = \dot{y}\frac{\partial y}{\partial d} = \dot{y} \times 1,\\
\dot{v}_{13} &= \frac{\partial e_3}{\partial v_{13}} = \frac{\partial e_3}{\partial e_2}\frac{\partial e_2}{\partial y}\frac{\partial y}{\partial v_{13}} = \dot{y}\frac{\partial y}{\partial v_{13}} = \dot{y} \times 1,\\
\dot{v}_{12} &= \frac{\partial e_3}{\partial v_{12}} = \cdots = \dot{v}_{13}\frac{\partial v_{13}}{\partial v_{12}} = \dot{v}_{13} \times 1,\\
\dot{v}_{11} &= \frac{\partial e_3}{\partial v_{11}} = \cdots = \dot{v}_{13}\frac{\partial v_{13}}{\partial v_{11}} = \dot{v}_{13} \times 1,\\
\dot{c}_2 &= \frac{\partial e_3}{\partial c_2} = \cdots = \dot{v}_{12}\frac{\partial v_{12}}{\partial c_2} = \dot{v}_{12} \times v_{10},\\
\dot{c}_1 &= \frac{\partial e_3}{\partial c_1} = \cdots = \dot{v}_{11}\frac{\partial v_{11}}{\partial c_1} = \dot{v}_{11} \times v_9,
\end{aligned}$$

and so on. All expressions on the right side of the above equation chains can be evaluated because the parameter values and the values of the hidden variables are known (the latter of which were computed and stored during the forward pass). There is no need to formulate complicated analytic expressions of partial derivatives at any stage.

In the example graph, no nodes are connected to two or more nodes other than the input nodes. In the case of multiple connections, the contributions from different paths must be summed to obtain the correct result, for example,

$$\dot{x}_1 = \frac{\partial e_3}{\partial x_1} = \dot{v}_1 \frac{\partial v_1}{\partial x_1} + \dot{v}_2 \frac{\partial v_2}{\partial x_1} = \dot{v}_1 a_{11} + \dot{v}_2 a_{21}.$$

The above partial derivative with respect to the input variable x_1 is given for the sake of illustration only.

4.4 OPTIMIZATION OF NETWORK PARAMETERS

The goal of training a neural network is to minimize the loss function. This involves the use of a gradient. The gradient of a scalar-valued function $f(x_1, x_2, \ldots, x_n)$ is defined using partial derivatives, as follows:

$$\nabla f(\boldsymbol{y}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{y}) \\ \frac{\partial f}{\partial x_2}(\boldsymbol{y}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\boldsymbol{y}) \end{bmatrix}.$$

The gradient of the loss indicates the direction in the parameter space in which the loss function increases the most. As the goal is to minimize the loss, the parameters are updated in the opposite direction (i.e., to the direction of the negative gradient). The magnitude of the update is controlled with a *learning rate* $\eta > 0$. For example, let $\boldsymbol{\theta} = (a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2, c_1, c_2, d)$ and let $\mathcal{L}(\boldsymbol{\theta})$ be the loss function of the network in Figure 4.2. Then, the updated parameters are obtained as follows:

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}).$$

Here, the components of $\nabla \mathcal{L}(\boldsymbol{\theta})$ are the partial derivatives computed during the AD. This optimization method is known as *gradient descent* [124].

In neural network training, the network parameters are updated not only after feeding all of the training data through the network, but they can be updated after every individual training example. This training method is known as *stochastic (or online) gradient descent* [53, p. 144]. Then, even more commonly, the data are fed to the network in small batches of training examples, known as *minibatches*. The resulting training approach is consequently known as *minibatch gradient descent*. The losses of individual training examples within a minibatch are commonly averaged to obtain a single loss value for use with AD. In this case, the averaging operation of losses (which can be presented with addition and division operations) must be added to the computational graph before performing AD.

Figure 4.3 demonstrates the gradient descent algorithm applied to a function with two parameters. At first, the parameter updates are large, but as the magnitude of the gradient becomes smaller, the updates also become smaller.

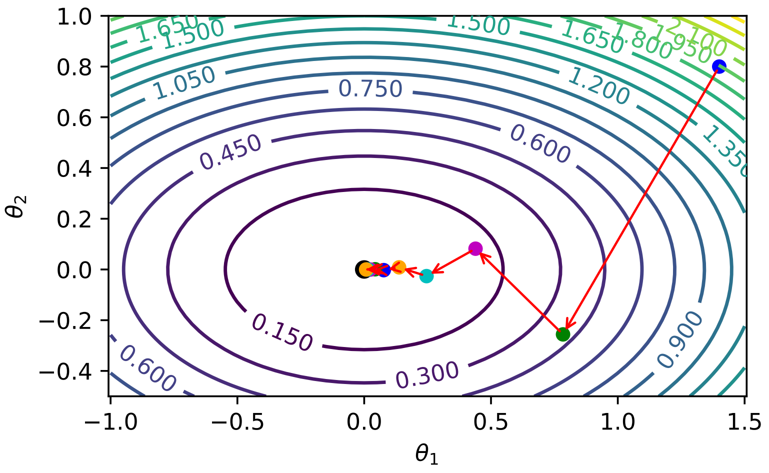


Figure 4.3: Optimizing parameters θ_1 and θ_2 to minimize the function $f(\theta_1, \theta_2) = \frac{1}{2}\theta_1^2 + \frac{3}{2}\theta_2^2$ with the gradient descent algorithm. The minimum of the function is obtained using the values $\theta_1 = \theta_2 = 0$.

In addition to the gradient descent algorithm, numerous other gradient-based optimizers exist that differ in the speed of convergence and sensitivity toward hyperparameter settings. Perhaps the most-used are gradient descent with *momentum* [125] (i.e., *Polyak's heavy ball method*), gradient descent with *Nesterov acceleration* [126, 127], *root mean square propagation* (RMSprop) [128], and *adaptive moment estimation* (Adam) [129]. In short, momentum adds memory to the parameter updates so that the current update is affected by the previous one, as follows:

$$\begin{aligned} \boldsymbol{z}_{new} &= \alpha \boldsymbol{z} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}), \\ \boldsymbol{\theta}_{new} &= \boldsymbol{\theta} + \boldsymbol{z}_{new}, \end{aligned}$$

where $\alpha \in (0, 1)$ is a parameter that controls the amount of decay in the momentum. The Nesterov acceleration is similar, but it first jumps one step forward in the parameter space in the direction of the momentum. Then, it computes gradients at that point, and adds a small correction to the parameters using this gradient. Finally, it updates the momentum. RMSprop optimizer scales the update size separately for each parameter using the square roots of the exponential moving averages of the squared partial derivatives (second-order moments). Finally, the Adam optimizer is similar to RMSprop with the addition of using first-order moments in the parameter optimization.

Parameter optimization is often regularized with *weight decay* (i.e., the L_2 *penalty*) [130]. Weight decay adds a regularization term $\frac{1}{2}\lambda\boldsymbol{\theta}^\top\boldsymbol{\theta}$ to the loss function $\mathcal{L}(\boldsymbol{\theta})$ to penalize parameter values with a high absolute value. This affects the gradient computation in AD. For example, the partial derivative with respect to the bias term $d \in \boldsymbol{\theta}$ becomes

$$\dot{d}_{\text{reg}} = \dot{d} + \lambda d$$

after adding the regularization term to the loss. This follows from the differentiation of the regularization term with respect to d (d must have a straight connection to the loss in the computational graph). Similarly, when computing the partial derivatives with respect to any other parameter, only the term 'λ times the parameter value' is added to the original partial derivative. Therefore, the loss (and the computational graph) does not need to be modified for the AD phase. It is instead enough to let the optimizer add the required terms to the original partial derivatives.

All publications in this dissertation involving DNNs (Publications **VI**, **VII**, **VIII**, and **IX**) used the minibatch gradient descent optimizer. In addition, the momentum was used in the first three publications. The last publication used weight decay.

4.5 COMMON BUILDING BLOCKS OF FEEDFORWARD NETWORKS

This section presents a selected set of commonly used constructs in feedforward networks starting with the most basic types of layers in feedforward networks, *fully connected* (FC) and *convolutional* layers [131]. The FC layers implement linear transforms similar to the function $h(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ defined earlier. The number of columns in the parameter matrix $\boldsymbol{A}$ is determined by the number of the input nodes, and the number of rows in $\boldsymbol{A}$ and $\boldsymbol{b}$ are determined by the number of nodes in the FC layer. In convolutional layers, a convolution kernel slides over the input data computing one value per one position of a kernel (Figure 4.4). The computation involves element-wise multiplication between the kernel and input data followed by the summation of the element-wise products. This allows using a shared set of weights (defined by the kernel) in all locations of input data. Typically, convolutional layers also append a learnable bias term to the outputs.

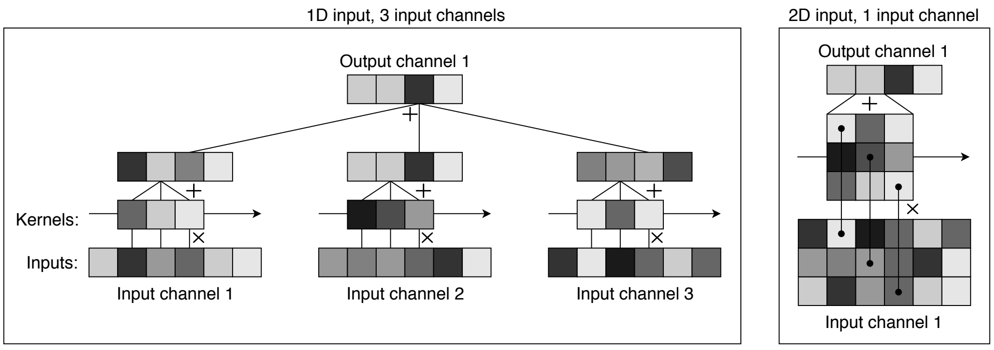


Figure 4.4: Illustrations of one-dimensional (1D) convolutions with 1D input data given in multiple channels (left) and 2D input data given in a single channel (right).

The convolutional layers come in different variations. First, the input data can be an array of one, two, three, or higher dimension.[1]. Then, the convolution kernels can move over the input data in many ways. For one-dimensional (1D) inputs, the kernels can move only in one direction. For 2D inputs, they can move in one or two dimensions. For 3D inputs, the kernel can move in one, two, or three dimensions, and so on. The number of dimensions over which the convolution kernel slides determines whether the operation is called a 1D, 2D, or 3D convolution. The output array of the convolutional layer has the same number of dimensions as the dimensionality of the convolution. The dimensionality of the convolution is different from

[1]In the context of machine learning, arrays of numerical data are typically called *tensors*. The usage of the term is not exactly the same in mathematics and physics.

the input data dimension if the kernel is the same size as the input data in some of the dimensions. An example of this is shown in the right panel of Figure 4.4.

In addition, convolutional layers can have multiple input and output *channels*. In audio processing with 1D convolutions, different dimensions of input acoustic features are represented as different input channels, and the convolution kernels span over time (this is similar to the convolution in the left panel of Figure 4.4). If a layer has many input channels, then each channel has its own kernel, and the outputs from these kernels are summed to obtain the output for one output channel. Multiple output channels are obtained by repeating the process of obtaining one output channel with different kernels for each output channel. Thus, if a layer has U input channels and Y output channels, then the total number of kernels is UY.

Figure 4.4 illustrates 1D convolutions with 1D and 2D input data. The convolution with 1D input data and multiple input channels is the same as the convolution with 2D input data with only a single input channel. Similarly, one could represent a 1D convolution of 2D inputs and multiple input channels using a 1D convolution of 3D inputs with only one input channel.

Both the convolutional layers and FC layers have been found to work well with ReLU activations [121] and *batch normalization* (BN) [132]. These two operations may be applied in either order depending on the application [133]. The ReLU activation is defined as follows:

$$f(x) = \max(0, x).$$

As observed, ReLU is a piecewise linear function that represents the identity mapping for input values greater than 0; for inputs less than 0, the output is 0. The ReLU has the following potential advantages [121]: 1) it avoids vanishing partial derivatives (i.e., values becoming too small) during AD, 2) it allows the network to have sparse representations, and 3) it is computationally inexpensive. The potential problems of ReLU include 'dying' network nodes (i.e., the partial derivative remains 0 if all inputs are smaller than 0) and an unbounded range of outputs, possibly leading to numerical problems. The former property was not found to be detrimental in the experiments conducted in [121], and the latter problem can be alleviated using regularization techniques, such as weight decay and BN.

BN uses all of the training samples within a minibatch to separately normalize each of the layer outputs to a zero sample mean and unit sample variance. More specifically, the BN operation is given as follows:

$$\mathrm{BN}(x) = \gamma \frac{x - \mu}{\sigma} + \beta,$$

where μ and σ are the mean and standard deviation computed over the minibatch dimension, respectively. The parameters γ and β allow scaling and shifting of the BN output. When BN is applied to the outputs of the convolutional layer, the normalization statistics μ and σ are computed separately for each output channel. This way, the BN parameters and statistics are shared between every position of the input — just like in the convolution operation.

Moreover, BN accelerates neural network training. The explanations for this effect include that BN helps reduce *internal covariate shift* [132], smooths the loss function [134] to derive more stable gradients, and causes *length-direction decoupling* [135, 136] of the weight parameters. In addition, BN helps avoid overfitting the training data because the computation for any given training example is not deterministic due to dependency on other (randomly selected) examples within a minibatch [132]. This, however, raises the question regarding how BN can be applied during test time when the samples are processed independently. This is achieved by maintaining a running exponential average of the normalization statistics computed over the consecutive minibatches during training. These statistics are used during testing.

4.6 DEEP LEARNING APPLIED TO SPEAKER RECOGNITION

Over the years, many different approaches have been taken to use DNNs for speaker recognition tasks. This section reviews some of the most adopted approaches. These include using DNNs as extractors of frame-level [137–139] and utterance-level features (embeddings) [35, 140, 141]. Furthermore, DNNs have been adopted in computing UBM-posterior probabilities as an alternative to using GMMs with equation (3.6) [142, 143].

The approach replacing the computation of UBM posteriors with DNN-based posteriors was proposed in 2014 [142]. In that work, a DNN (similar to Figure 4.5c without the bottleneck) with seven FC layers was used to compute the posteriors needed in Baum-Welch statistics for i-vector extraction. The input features for the DNN were 40-dimensional vectors of log mel-filterbank confficients. For each frame, a longer context was created by stacking the seven preceding and seven following feature vectors with the current feature vector. Thus, the network inputs were $40 \times (7 + 1 + 7) = 600$ dimensional. The target labels were phonetic units called *senones* computed using a separate *automatic speech recognition* (ASR) model. Senones are tied states of triphone models [142, 144]. The network had a softmax output layer consisting of 3450 nodes representing different senones. Therefore, it served as a 3450-component *DNN-UBM* for posterior probability extraction. The benefit of this approach is that the model encodes more accurate phonetic information to the posteriors. That is, the frames that are aligned with a certain component of the DNN-UBM should, in theory, always represent the corresponding triphone, no matter how it was pronounced by the speaker. In traditional GMM-UBM, different realizations of the same phoneme could become aligned with different components, and as a result, some phonetic information about differences between speakers is lost. The drawback of DNN-UBM is that it increases the complexity of the system development because it requires an ASR model for senone computation.

In [143], the DNN of DNN-UBM was replaced with a *time delay neural network* (TDNN) [146]. The TDNN is a special case of *convolutional neural network* (CNN), in which convolution kernels slide in the temporal direction. In TDNN implementations, the convolution kernels are typically *dilated* (i.e., the kernels have empty gaps). The i-vector systems based on both TDNN-UBM and DNN-UBM achieved considerably higher speaker verification accuracy compared to the corresponding i-vector systems based on GMM-UBM [142, 143].

Another commonly used approach that appeared in the early 2010s is the use of DNNs to extract *bottleneck features* [137–139]. Bottleneck features are frame-level features obtained from the traditional (usually stacked) acoustic features through transformations implemented by a neural network. The training targets for bottleneck feature extractor networks are typically some sort of phonetic units, such as senones. The bottleneck features are not obtained from the output layer but from some intermediate layer of the network that has smaller surrounding layers (hence the name 'bottleneck'). The bottleneck layer forces the network to pass the data

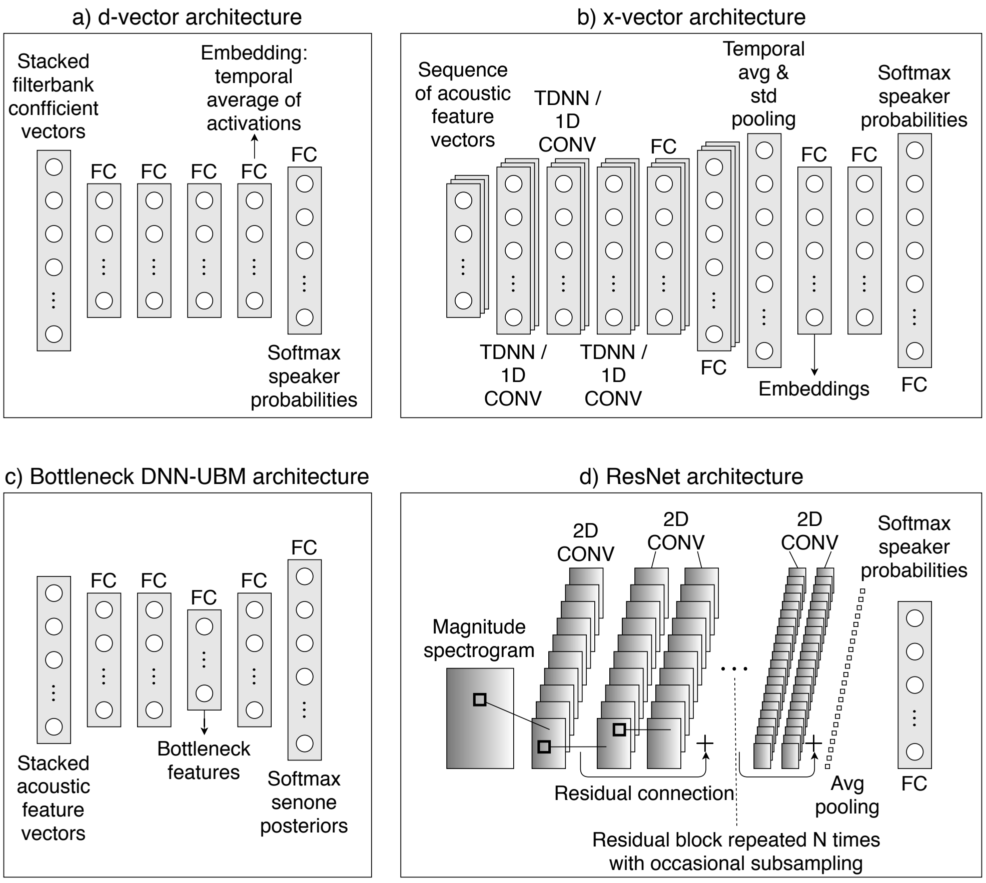

a) d-vector architecture
Stacked filterbank confficient vectors
Embedding: temporal average of activations
FC
FC
FC
FC
FC
Softmax speaker probabilities
b) x-vector architecture
Sequence of acoustic feature vectors
TDNN / 1D CONV
TDNN / 1D CONV
TDNN / 1D CONV
FC
FC
Temporal avg & std pooling
FC
FC
Softmax speaker probabilities
Embeddings
FC
c) Bottleneck DNN-UBM architecture
Stacked acoustic feature vectors
FC
FC
FC
FC
FC
Bottleneck features
Softmax senone posteriors
d) ResNet architecture
Magnitude spectrogram
2D CONV
2D CONV
2D CONV
Softmax speaker probabilities
Residual connection
Avg pooling
FC
Residual block repeated N times with occasional subsampling

Figure 4.5: Illustrations of some popular deep architectures used for speaker recognition. The d-vector [140], x-vector [35], and ResNet [145] architectures can be used to extract speaker embeddings. The bottleneck DNN-UBM architecture can be used in two ways: to extract bottleneck features or extract senone posteriors, both of which are typically used as inputs for i-vector systems.

through a relatively small-dimensional internal representation, which should contain relevant information to decode the output representation as well as possible. Thus, it can be considered a dimensionality reduction mechanism [147].

In [148], the same network is used for extracting both bottleneck features and senone posteriors. Combined with the traditional MFCCs and GMM, these allowed four combinations for the Baum-Welch statistics computation: 1) MFCC features with GMM posteriors, 2) MFCC features with DNN posteriors, 3) bottleneck features with GMM posteriors, and 4) bottleneck features with DNN posteriors. The third option gave the best results followed by the second option. Figure 4.5c displays a crudely simplified illustration of the network used in the study.

In 2014, another line of research was initiated when *deep vectors* (*d-vectors*) were proposed [140]. Deep vectors are speaker embeddings obtained through averaging the outputs of a hidden layer of a DNN trained using speaker labels as targets. Figure 4.5a (reproduced from [140]) illustrates the original d-vector architecture. A key point to note from the architecture is that unlike in the bottleneck approach, the network is trained for the actual end goal of performing speaker recognition. However, the whole network operates on the frame level even if it is used to construct utterance-level features from the outputs of a hidden layer. This can be a drawback because the network cannot fully use the information in an utterance, as it only 'sees' short segments of it. Perhaps this is one of the main reasons the d-vector approach did not exhibit impressive results when compared to the traditional i-vector systems [140].

In the *x-vector* approach presented in 2017 [117], the averaging (*pooling*) operation to switch from frame-level to utterance-level representation occurs inside the network, as illustrated in Figure 4.5b. In addition to computing averages over different frame-level representations, the pooling layer computes standard deviations and stacks them with the averages. The x-vector embeddings are extracted from the first layer after the pooling layer before applying the activation function and BN. In addition to the pooling layer, the x-vector architecture differs from the d-vector in that it uses a TDNN in the frame-level part of the network. The convolutional nature of the TDNN is useful for modeling temporal dependencies between adjacent frames while keeping the parameter count relatively low. Since their introduction, x-vectors and similar variants [78] have been considered state-of-the-art speaker embeddings.

Recently, *ResNets* [145] have also been adopted for speaker recognition with comparable results to x-vector systems [78, 141, 149]. ResNets process 2D inputs with a CNN that includes *residual connections*. A residual connection is a connection that adds the output of a layer to the output of some other subsequent layer (skipping one or more layers in between). Residual connections facilitate training deeper networks by allowing partial derivatives to propagate to the first layers of the network with fewer connections, thus alleviating vanishing gradient problems. Even with residual connections, increasing the network depth too much may not be a wise strategy, as discussed in [150]. A simplified illustration of a ResNet is presented in Figure 4.5d. Originally, ResNets were designed for image recognition; thus, the pooling layer and the layers around it may need to be modified to better suit for temporal speech data, as performed in [78] and [141].

Other than the above architectural choices, the recent research trends in speaker recognition include the study of various pooling layer constructs and loss functions. Perhaps the best known alternatives for the temporal average and standard deviation pooling are the *learnable dictionary encoder* [96] and the netVLAD (*Vector of Locally Aggregated Descriptors*) [151]. In both of these constructs, the pooling layer inside the network is replaced by a construct that resembles a GMM. That is, instead of computing statistics *globally*, they are computed *locally* around the learnable clusters (or components) of the models. Publication **IX** explains the relation of the LDE and netVLAD to the GMM in detail.

The alternatives to the commonly-used cross-entropy loss with softmax activations include *angular-softmax loss* [152], *angular additive margin loss* [153], *triplet loss* [154], and *generalized end-to-end loss* (GE2E) [155] to name but a few. The first two of these losses compute the classification loss based on the angles between the speaker embeddings and weight-vectors of the last layer. These angular loss based methods minimize intra-speaker variation and maximize inter-speaker variation by enforcing margins around the angles. Unlike the first two losses, the last two losses do not compute the loss value for each embedding independently. Instead, the general idea is to push embeddings from different speakers within a minibatch further apart while bringing embeddings from the same speaker closer to each other.

5 SUMMARY OF PUBLICATIONS

This dissertation includes nine publications. They are grouped into the three selected themes of **robustness**, **speed**, and **security**. The following presents a summary of each paper in each theme. Finally, theme-wise summaries are followed by a summary of published software implementations relating to the papers.

5.1 STUDIES ON ROBUSTNESS

The primary focus of Publications **I**, **II**, and **IX** was improving the robustness of automatic speaker recognition systems. The first two publications address the issue by proposing robust acoustic features using techniques such as *frequency domain linear prediction* (FDLP) [156–158] and *time-varying linear prediction* (TVLP) [159,160]. The first study uses proposed features in *reverberant* conditions, whereas the second study uses them to tackle the problems caused by speaking-style mismatch induced by *whispering*. The third study focuses on developing the core speaker recognition technology. The study uses modern deep embedding extractor networks with data augmentation to perform speaker recognition with audio extracted from videos. The embedding networks are also employed as Baum-Welch statistics extractors for the *neural i-vector* systems proposed in the study. The following paragraphs (see Table 5.1) summarize all of the three studies in more detail starting from the studies employing linear prediction (LP) techniques.

A well-known property of LP is its ability to convey essential information about the speech spectrum within a small number of parameters (i.e., in the predictor coefficients a_i) [161]. In Publications **I** and **II**, different types of LP are adopted for speech spectrum estimation. The baseline method for these studies is *2D autoregressive model* (2DAR) [65,162], which uses both FDLP and LP in succession. FDLP is an analogous method to LP, where the roles of the time and frequency domains are interchanged. Whereas LP is applied to windowed time-domain signals to obtain short-time frequency spectrum estimates, FDLP applies LP to windowed frequency *subbands* to obtain time-domain envelopes for the subband signals. Therefore, the first step in FDLP is to transform the time-domain signal to the frequency domain with the aid of *discrete cosine transform* (DCT). The transformed signal is then windowed and FDLP modeling is applied to each subband window. As the windowing is done in frequency rather than in the time domain, FDLP allows modeling the long-term temporal structure of speech signals from contexts longer than the typical frame contexts of 25 ms or so. The time-domain envelopes of subbands given by FDLP can be stacked (usually after subsampling by integration) to obtain a spectrograph-like representation of a speech signal.

As noted above, FDLP is followed by LP in 2DAR modeling. Because the spectrogram output of FDLP cannot be directly used as input to LP, it is converted to sequences of autocorrelation coefficients (2.4) by applying the inverse Fourier transform [55,65] to every frame of a spectrogram.

The first two studies propose a modification to 2DAR, where the LP step after FDLP is replaced by TVLP. In TVLP, the predictor coefficients are allowed to be time-varying, so that equation (2.2) becomes the following:

$$\hat{x}[n] = \sum_{k=1}^{p} a_k[n]x[n-k].$$

The predictor coefficients are restricted from being able to change arbitrarily over time by expressing them using a set of q predefined continuous basis functions $u_j[n]$ as follows:

$$a_k[n] = \sum_{j=1}^{q} b_{kj}u_j[n].$$

Here, b_{kj} are the model parameters to be solved. In the standard TVLP formulation, these parameters cannot be solved using the autocorrelation coefficients of form (2.4); therefore, applying TVLP after FDLP is problematic. Hence, Publications **I** and **II** propose a modified version of TVLP called *autocorrelation domain TVLP* that uses the autocorrelation sequences in solving parameters b_{kj}.

Both Publications **I** and **II** address robustness against a mismatch between the speaker enrollment and test conditions. In Publication **I**, the mismatch was caused by reverberation in test segments, whereas in Publication **II**, the speaking style in the enrollment and test utterances was different. More precisely, the enrollment utterances in **II** were modal speech, whereas the testing utterances were whispered speech. The proposed FDLP-TVLP method outperformed the baseline methods in both of the publications in mismatched conditions. Publication **I** also indicated promising results for non-mismatched testing conditions, whereas, in Publication **II**, the baseline methods outperformed the proposed features when speaking-style mismatch was absent.

In addition to studying the proposed features with whispered speech, Publication **II** has the following contributions. First, it includes a comprehensive literature review on speaker recognition from whispered speech. Second, it proposes a transcription-free alignment method to align utterances of normal and whispered speech that share the same lexical content. The method can detect poorly aligned speech segments so that they can be discarded when analyzing the differences in modal and whispered speech. Third, the study provides an analysis of how the first three formants (F1, F2, and F3) change when changing from normal to whispered speech. The results indicate that the formant frequency F1 changes the most by becoming, on average, 150 Hz higher in whispered speech.

The last study (Publication **IX**) under the robustness theme uses modern DNN components for speaker embedding extraction. The primary novelty of the work is the *neural i-vector*, a new construct in which the generative i-vector is extracted from Baum-Welch statistics extracted by a DNN. This DNN is a deep speaker embedding extractor network equipped with a GMM-like pooling layer. Two such pooling layers exist, namely *learnable dictionary encoder* (LDE) [96, 163] and netVLAD (VLAD is an acronym for a vector of locally aggregated descriptors) [151, 164]. The differences between our work and the existing DNN i-vector formulation [142] are the following. First, our DNN uses speaker labels as training targets instead of senone labels. Second, the posteriors used for statistics computation are not obtained from the output layer but the pooling layer. Lastly, the frame-level features for statistics computation are discriminatively trained because they are also obtained from the DNN.

The study also provides explanations on how the LDE and netVLAD pooling layers resemble GMMs. Different variants of these pooling layers are experimentally compared to each other. In addition, we experiment with residual connections [145] and *squeeze-and-excitation* (SE) modules [165,166].

This preliminary investigation of the neural i-vectors indicates that their performance is better than the other i-vector based systems reported in the literature. On the other hand, the performance of neural i-vectors is worse than their corresponding DNN embedding counterparts. Further, the benefits of residual connections and SE modules appeared to be small and dependent on the evaluation conditions. The residual connections may be more useful for deeper network architectures than those used in the study.

Table 5.1: Summary of studies on **robust** speaker recognition.

Study	Datasets	Applications	ASV systems	Contributions	Selected results
I	Male speakers from RedDots [84]	Text-dependent speaker verification in reverberant conditions	GMM-UBM	New autocorrelation domain TVLP; new FDLP-TVLP features	EERs for FFT+RASTA / 2DAR+RASTA / Proposed features (1st row: non-reverberated, 2nd row: high reverberation): 3.08% / 3.02% / 2.87% 14.03% / 8.03% / 7.51%
II	CHAINS [167]	Text-independent speaker identification and verification from whispered speech	GMM-UBM	Extended presentation of **I**; Extensive literature review on speaker recognition studies with whispered speech; Transcription-free alignment method of normal and whispered speech; Analysis of formant frequencies for whispered speech in CHAINS corpus	EERs for DFT / 2DAR / Proposed methods (1st row: normal-normal condition, 2nd row: normal-whisper mismatch condition): 2.57% / 3.10% / 3.27% 29.69% / 28.43% / 27.48% Formant analysis indicated that F1 is 150 Hz and F2 is 100 Hz higher in whispered speech than in normal speech
IX	VoxCeleb1 [81], VoxCeleb2 [82], SITW eval [83], SRE18 VAST eval [87], SRE19 VAST eval [88]	Text-independent speaker verification	DNN embeddings and neural i-vectors, PLDA, score normalization	New kind of neural i-vectors; Presentation of LDE and NetVLAD pooling layers in the context of GMM; Investigation of performance of different pooling layers, residual components, and squeeze-and-excitation modules	EERs for SITW evaluation: Our DNN embeddings (1.83%), DNN embedding literature baseline (1.70%), Our neural-ivectors (2.81%), i-vector literature baseline (3.38%)

5.2 STUDIES ON COMPUTATIONAL SPEED-UPS

Among other topics, Publications **III**, **IV**, and **V** focus on the speed of speaker recognition systems in training and testing. Publication **III** investigated multiple fast-to-train speaker embedding extractors that operate on MAP-adapted GMM-supervector inputs. Publication **IV** used one of the methods discussed in Publication **III** in an online web demonstrator of speaker recognition technology. To this end, it also considered the testing speed of the system. In the third study (Publication **V**) of this theme, the standard i-vector system is implemented to use a graphics processing unit (GPU) computation to obtain substantial training acceleration. All three studies are discussed in more detail below. In addition, a summary in tabular form is given in Table 5.2.

Publication **III** examined four different methods for reducing the dimensionality of supervectors to obtain lower-dimensional speaker embeddings. The methods were *probabilistic principal component analysis* (PPCA) [106], *factor analysis* (FA) [53, pp. 583–586], *probabilistic partial least squares* (PPLS) [168], and *supervised PPCA* (SPPCA) [169]. The first two are *unsupervised* methods that do not require speaker labels, whereas the last two are *supervised*. In the PPLS model, the labels are encoded as one-hot vectors, which are set to share the same latent representations with the corresponding supervectors. Finally, the SPPCA model uses speaker labels to create one supervector for each speaker by averaging the supervectors of different utterances of the same speaker.

The findings in Publication **III** are the following. First, the differences between most of the supervector dimensionality reduction techniques were small, supporting the use of the simplest method, that is, the PPCA. The PPCA model was more than one hundred times faster to train than the standard i-vector model (i.e., the total variability model), and it obtained similar EERs to the i-vector model. However, this result should be carefully considered because the specific i-vector baseline lacked two training procedures that would improve the performance. These two techniques, *minimum divergence re-estimation* and covariance matrix updates at M step of the EM-algorithm were later studied in Publication **V**.

Publication **V** uses the modern machine learning library PyTorch [170] to implement GPU-accelerated i-vector model training. The experiments indicate that the GPU implementation is more than an order of magnitude faster than the corresponding CPU implementation in the Kaldi speech recognition toolkit [171]. The considerable speedup allowed an extensive investigation of different training variations of the otherwise slow-to-train model. The findings of this investigation are the following: Performing minimum divergence re-estimation between training iterations results in a 7.5% to 9% lower EER (relatively), and updating the covariance matrices results in a 1.5% to 3% lower EER (relatively) with the VoxCeleb dataset. In addition, updating the frame alignments and recomputing Baum-Welch statistics after every iteration of training can lower EER by about 1% relatively.

An additional contribution of Publication **V** is documenting certain unpublished details regarding Kaldi's [171] i-vector extractor formulation. Kaldi's formulation is not the original formulation [50] but one in which the bias term of the model is augmented into the total variability matrix. Therefore, it is referred as the *augmented formulation* in Publication **V**. The study explains how this change in the model affects its training.

After **III** was published, the PPCA-based speaker identification system was deployed into a web demonstration application to be used for science popularization

Figure 5.1: A research team member demonstrating speaker recognition technology in a sub-event of *the European Researchers' Night (ERN) 2019*. The ERN events funded by the European Commission are dedicated to educating the public about the current state of the research work. The sub-event in the picture took place at *Matkus Shopping Center, Kuopio, Finland*. During the four-hour-long event, numerous people ($>$ 100) recorded their voices to discover whom they sound like.

purposes (Figure 5.1). The identification system was trained to recognize YouTube celebrities present in the VoxCeleb 1 [81] and 2 [82] datasets. In the web demonstration, the user was asked to record a short speech sample, which was compared with the VoxCeleb speakers to determine the closest sounding speakers. The results were presented as a top-five list, which included embedded YouTube video players so that the users could watch the videos of the closest matching people. To provide a smooth user experience, special attention was paid to optimizing the system speed at test time. This work is documented in Publication **IV**.

Table 5.2: Summary of studies on **fast** speaker recognition.

Study	Datasets	Applications	ASV systems	Contributions	Selected results
III	VoxCeleb1, SRE04, SRE05, SRE06, SRE10, Switchboard, Fisher	Text-independent speaker verification	GMM-supervectors + PPCA / FA / PPLS / SPPCA, PLDA	Comparison of fast-to-train alternatives to the total variability model	EERs for VoxCeleb test set: PPCA 7.04%, i-vector baseline 7.09% The PPCA model was over 100x faster to train
IV	VoxCeleb1, VoxCeleb2	Speaker identification web application: finding the closest sounding YouTube celebrity	GMM supervectors & PPCA, PLDA	A concept of a fast technology demonstrator for science popularization; Evaluation of the proposed concept	Median time for responding to a user after stopping recording was 1.8 s; Every tester who completed the feedback questionnaire agreed that the results showed up quickly; The majority of testers could consider recommending the demonstration to others
V	VoxCeleb1, VoxCeleb2	Text-independent speaker verification	i-vector, PLDA	GPU-accelerated i-vector extractor training (codes available); Comparison of training variations; Investigation of re-computing posteriors during i-vector extractor training; Documentation of Kaldi's [171] i-vector extractor;	25x faster i-vector model training with PyTorch GPU implementation when compared to Kaldi's CPU implementation; The difference in EER between the worst and best training variations is more than 10%, relatively

5.3 STUDIES ON SECURITY

Publications **VI**, **VII**, and **VIII** focus on security aspects related to automatic speaker recognition. Each of the studies is summarized below (Table 5.3).

Publication **VI** presented the ASVspoof 2019 anti-spoofing challenge and its results. The ASVspoof 2019 challenge was third in a series of biannually organized ASV anti-spoofing challenges. The earlier editions of ASVspoof challenges were organized in 2015 [41] and 2017 [42]. ASVspoof 2019 consisted of two separate challenge scenarios, namely the *logical access* (LA) and *physical access* (PA) scenarios. In the LA scenario, spoofing attacks are injected directly into the ASV system, bypassing the microphone. The spoofing attacks for LA are created using voice conversion or text-to-speech synthesis. In the PA scenario, the spoofing attacks are captured using the ASV system microphone in a reverberant room. The spoofing attacks considered in the PA scenario are *replay attacks*, in which the recorded speech of the target speaker is replayed to the ASV system through a loudspeaker. Unlike in ASVspoof 2017, the PA audio files in ASVspoof 2019 were not collected by replaying and recording speech; instead, the replay process was *simulated*. Simulation allows the study of the replay attacks in different acoustic environments and with different replay device characteristics in a more controlled way. In addition, simulation facilitates the creation of large replay datasets because simulation lessens the manual human work required.

In the earlier editions of the ASVspoof challenge in 2015 and 2017, spoofing countermeasures were evaluated using the EER of the spoofing detection task. This metric, however, does not provide satisfactory insight on how well a *combined system* of a spoofing countermeasure and ASV system would perform. For example, even if the spoofing countermeasure EERs are high for a specific attack, the combined system might still work fine if the attack cannot deceive the plain ASV system. To this end, the ASVspoof 2019 was the first challenge to include the *tandem* DCF (t-DCF) metric [6,172], which evaluates spoofing countermeasures with an ASV system. The ASV system (specifically, the *scores* of the ASV system) were provided by the challenge organizers so that the challenge participants did not need to develop their own ASV systems. The ASV system in the 2019 challenge was an x-vector system based on the Kaldi recipe for VoxCeleb data.

The ASVspoof 2019 was successful in activating anti-spoofing research: more than 150 teams registered for the challenge from all over the globe (including academic and industry participants). A total of 63 teams managed to submit their countermeasure scores, and over half of these submissions performed better than the baseline countermeasures provided by the organizers. The best teams managed to reach spoofing detection EERs lower than 1% in both LA and PA scenarios.

Publication **VII** presents the study of two kinds of attacks that were not included in the ASVspoof challenges or any other challenges to the best of the knowledge of the author. These are *impersonation (mimicry) attacks* and attacks using *technology-assisted target speaker selection*. In an impersonation attack, an impersonator aims to deceive the ASV system by changing his or her voice to sound like the target speaker. The technology-assisted target speaker selection refers to an act of searching for similar voices to the attacker's voice from a large set of speakers using ASV technology. Then, by presenting him or herself to the ASV system as the closest sounding target, the attacker has a higher chance to deceive the ASV system (or human listener). The work aimed to study whether impersonation attacks can be improved by combining them with technology-assisted target selection. To this end, we first selected suitable targets for the recruited impersonators using an ASV system. Then, the impersonators were asked to impersonate the voices of the selected targets. These impersonation recordings were then used to attack another ASV system.

The results of the study suggest that technology-assisted target speaker selection helps create stronger attacks. In other words, the scores of one ASV system were correlated to the scores of another ASV system. Contrary to our expectations, the impersonation attacks were not found to be helpful, at least when the target speaker was already close to the impersonator in terms of speaker recognition scores.

The examination of technology-assisted target speaker selection attacks is continued in Publication **VIII**. This work estimates the likelihood of finding a speaker (worst-case impostor) among a large speaker population who could be verified by an ASV system to be a specific target speaker. This is achieved with the proposed performance metric *worst-case false alarm rate among N impostors* (P_{FA}^{N}). The proposed metric is similar to the standard FA rate but is computed with the most difficult impostors in terms of the ASV scores.

Because the number of speakers in publicly available datasets is limited, the estimation of P_{FA}^{N} is challenging for large values of N. Thus, the study proposes a generative model of ASV scores that can be used to extrapolate false alarm rates for a larger number of speakers than training datasets have. Using this model and the VoxCeleb data, it was estimated that P_{FA}^{N} is 54% for a population size of 100 000 with an x-vector system having a relatively strict threshold. That is, for a randomly selected target speaker, there is a 54% chance that the closest impostor from a population of 100 000 would be accepted as the target speaker.

Table 5.3: Summary of **security**-related studies.

Study	Datasets	ASV systems	Contributions	Selected results
VI	ASVspoof 2019 database [5], for ASV training: VoxCeleb1, VoxCeleb2	x-vector, PLDA	Organization of the anti-spoofing challenge (ASVspoof 2019) evaluating anti-spoofing methods against synthetic, converted, and replayed speech; The first challenge using the tandem DCF (t-DCF) metric	Sixty-three research teams participated; The best spoofing countermeasure systems are fusions of multiple subsystems reaching EERs under 1%
VII	VoxCeleb1, VoxCeleb2, Librispeech [173], WSJ [174], AVOID corpus [175], in-house data collection	i-vector, x-vector, PLDA	Evaluation of the vulnerability of the ASV system under *targeted* voice impersonation attacks (impersonation targets are selected with another ASV system); Evaluation of how well the ASV scores of one system (attacker's system) correspond to the ASV scores of another system (attacked system) with different technology; Fundamental frequency, formant, and speech rate analysis of impersonation efforts; Human evaluation of attacks	Target speaker selection using the attacker's ASV system helps create stronger attacks; Non-professional impersonators do not pose a risk to the attacked ASV system
VIII	VoxCeleb1, VoxCeleb2	i-vector, x-vector, PLDA	New *worst-case false alarm rate* performance metric; New generative model of non-target ASV scores; Model-based estimation of false alarm rates of ASV for large speaker populations	An x-vector system with a strict threshold obtained an estimated 54% worst-case false alarm rate for population size of 100 000

5.4 SOFTWARE IMPLEMENTATIONS

The research work for this dissertation involved implementing various pieces of software. Table 5.4 lists those ones that have been released into public. Apart from the web platform, the listed software was mostly developed by the undersigned.

Table 5.4: A list of published software implemented during the research work.

Related publications	Software title & link	Software description
III, **IV**	Supervector compression methods: `https://gitlab.com/ville.vestman/supervector-compression`	Contains MATLAB routines for compressing supervectors into smaller dimensional speaker embeddings using probabilistic principal component analysis (PPCA), factor analysis (FA), probabilistic partial least squares (PPLS), and supervised PPCA (SPPCA).
IV	Web platform for demonstrating speech processing: `https://github.com/bilalsoomro/speech-demo-platform`	A web platform based on PHP and Javascript. Sends a wav-file to the server-side, calls the specified speech processing method, and finally returns the output the to the client.
V	GPU accelerated i-vectors: `https://github.com/vvestman/pytorch-ivectors`	Contains PyTorch implementations of different variants of i-vector extractors including the one in the Kaldi Toolkit. Both the computation of alignment of frames to GMM components and the i-vector extractor training are accelerated with GPU.
V, **IX**	ASVtorch Toolkit: `https://gitlab.com/ville.vestman/asvtorch/`	ASVtorch is a PyTorch based speaker verification toolkit that includes the above i-vector implementations and many x-vector variants. The toolkit contains complete speaker verification pipelines for the VoxCeleb and Speakers in the Wild (SITW) datasets.

6 CONCLUSIONS AND FUTURE WORK

Speaker recognition technologies have been actively researched for many decades. The progress in the field has been remarkable, and there are no signs of slowing down. The progress has been fostered by both *developmental* and *research supporting* factors. The supporting factors are the increased availability of publicly available speaker recognition datasets and increasing computational resources. The developmental factors are the improvements produced through the research and development of the core recognition technology. These include, for example, improved feature extraction methods, speaker models, and back-end classifiers. The progress is also reflected in the work done in this doctoral dissertation. At the time of writing the first publication, state-of-the-art speaker recognition systems were still largely based on generative speaker embedding models. Since then, DNN-based embeddings have become increasingly powerful and are now the default choice for most speaker recognition tasks.

The improved technologies and digitalization of society have widened the scope of potential applications of automatic speaker recognition. The adoption of the recognition systems for these applications can be facilitated by continuing the development various aspects of the speaker recognition systems. These aspects include the robustness, security, and speed of speaker recognition, which are the main themes of this dissertation. The following sections present the conclusions from each studied theme along with ideas for future work.

6.1 ROBUSTNESS

The research presented in this dissertation contributed to the robustness of automatic speaker recognition in the following ways. The proposed *FDLP-TVLP features* (Publications **I** and **II**) exhibit the ability to mitigate problems caused by enrollment-test mismatch caused by reverberation and normal-to-whisper speaking-style changes. These results were obtained using the GMM-UBM speaker recognition system. Whether the proposed or similar feature extraction constructs are beneficial with modern DNN-based speaker recognition systems is left for future work. However, in the case of whispered speech, this might turn out to be challenging as the availability of whispered speech data might not be sufficient for DNN training. Thus, it could be interesting to try to improve the speaker recognition from whispered speech by creating artificial whispered speech data, for example, by using voice conversion technologies such as the ones proposed in [176]. The same idea could be also applied to improve robustness to high vocal effort levels, such as present in Lombard speech and shouting [177].

The last publication of this dissertation (Publication **IX**) explored new avenues in combining DNN models with generative models. To this end, the work used GMM-like pooling layers within a DNN embedding extractor network to use the network as a statistics extractor for the generative i-vector model. The proposed *neural i-vectors* outperformed the i-vector baselines reported in the literature but were not able to compete with purely DNN-based embeddings. Despite not achieving state-of-the-art results, the proposed framework could inspire future work in applications, such as ultra-short-duration speaker recognition. This hypothesis is supported by the fact that the studied framework could be modified to use a GMM-UBM-like frame-based scoring method, which could be useful, for instance, in fine-grained speaker diarization. The investigation of this idea is left as future work.

In the recent past, the ASV systems were relatively homogenous; most of them were using MFCC features, GMM-UBM or DNN-UBM, i-vectors, and PLDA. This period was followed by the large-scale adoption of DNNs, and as the consequence, design choices of the systems have become far more heterogenous. In the recent literature, the variation in the ASV system designs is prevalent in all parts of the system; in features, DNN architectures, and back-end solutions. Furthermore, while some systems retain the traditional division of system to frame level feature extractor, utterance level feature extractor, and back-end classifier, there are many systems that try to combine multiple parts together in more end-to-end fashion [178, 179]. In the long term, it will be interesting to see whether the current trend of diversification of system designs continues, and, if the system designs start to converge, to which specific design they converge to.

6.2 COMPUTATIONAL SPEED-UPS

Publications **III** and **V** contributed implementations and comparisons of variants of fast-to-train speaker recognition systems. The first study compares multiple supervector compression methods that are computationally less expensive than the traditional i-vector systems. The second study improves the training speed of traditional i-vector systems (including Kaldi's i-vector variation) by using GPU acceleration. The experiences from this work support the idea that probabilistic generative models can often be trained much faster by using GPUs. Then, contrary to the above two studies, Publication **IV** focused more on the testing speed. In this work, one of the methods (PPCA) in Publication **III** was adopted to create an online web demonstration of speaker recognition technology. The demonstration was optimized for test phase speed, and the response and computation times were measured and reported in the publication. The demonstration has been received well in various promotional and science popularization events, indicating that sometimes even relatively simple technology demonstrations can be very engaging to the audience.

The future work on computational speed-ups with modern ASV systems mostly involves finding the lightest, yet well performing DNN architectures. To automatize this task, ideas relating to *network pruning* [180], *differentiable neural architecture search* [181], and *neuroevolution* [182] could be investigated.

6.3 SECURITY

The contributions of this dissertation in the third and the final theme of security are various. Publication **VII** delivered new and updated knowledge about the rarely studied topic of technology-assisted mimicry attacks against ASV. In general, the attacks were not successful in misleading the ASV system. Nonetheless, the results suggest that the technology-assisted target speaker selection seems more helpful than the mimicry efforts by amateur mimickers in creating stronger attacks. This study used VoxCeleb as the target data, which due to the uncontrolled nature of the data, made both conducting the experiments (required manual cleaning) and the interpretation of the results more difficult. Another slight problem was caused by the nationality of mimickers (Finnish), which limited the number of possible target speakers of the same language to a relative small number of Finnish speakers present in the VoxCeleb corpus. Thus, in future, a study of a similar kind could be conducted but with a cleaner data, and possibly with native English speakers as mimickers.

Along the line of research of Publication **VII**, Publication **VIII** considered what happens if an impostor is the worst-case impostor (the closest speaker to the target speaker) — perhaps selected automatically by an ASV system from a large speaker population (similar to that used in Publication **IV**). To this end, the paper proposed a new *worst-case false alarm rate* metric. Another major novel idea in the paper is generative modeling of the scores of the ASV system to predict the false alarm rates with arbitrarily large speaker populations. This work has been recently continued in [7], which proposed discriminative training of various score models to improve the false alarm rate estimation.

Finally, Publication **VI** presented the ASVspoof 2019 challenge and its results. This was the third edition of the challenge and was once again highly successful in activating research on ASV anti-spoofing methods to detect replayed, synthesized, and converted speech. Plans and ideas for the future editions of ASVspoof exist and many of them will be highlighted in a new ASVspoof 2019 summary article, which is, at the time of writing, under review. To name a few, these ideas include the anti-spoofing under additive noise, inclusion of more diverse spoofing attacks, and inclusion of multi-channel data to reflect the use cases with devices having microphone arrays.

Whereas the ASV systems are expected to get extremely powerful within the next few decades, the anti-spoofing side may remain as a bottleneck due to the continuous technological arms race between spoofing attacks and countermeasures. Despite this, anti-spoofing research is valuable to prevent if not all, then at least the most easy to detect spoofing attacks. Furthermore, when ASV anti-spoofing countermeasures are combined with other modalities such as face and lip movement detection, ASV systems may become too challenging to attack by any practical means.

BIBLIOGRAPHY

[1] A. Kanervisto, V. Vestman, M. Sahidullah, V. Hautamäki, and T. Kinnunen, "Effects of gender information in text-independent and text-dependent speaker verification," in *Proc. ICASSP* (2017), pp. 5360–5364.

[2] K. A. Lee and SRE'16 I4U Group, "The I4U Mega Fusion and Collaboration for NIST Speaker Recognition Evaluation 2016," in *Proc. Interspeech* (2017), pp. 1328–1332.

[3] T. Kinnunen, R. G. Hautamäki, V. Vestman, and M. Sahidullah, "Can we use speaker recognition technology to attack itself? enhancing mimicry attacks using automatic target speaker selection," in *Proc. ICASSP* (2019), pp. 6146–6150.

[4] K. A. Lee, V. Hautamäki, T. H. Kinnunen, H. Yamamoto, K. Okabe, V. Vestman, J. Huang, G. Ding, H. Sun, A. Larcher, R. K. Das, H. Li, M. Rouvier, P.-M. Bousquet, W. Rao, Q. Wang, C. Zhang, F. Bahmaninezhad, H. Delgado, and M. Todisco, "I4U Submission to NIST SRE 2018: Leveraging from a Decade of Shared Experiences," in *Proc. Interspeech* (2019), pp. 1497–1501.

[5] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee, L. Juvela, P. Alku, Y.-H. Peng, H.-T. Hwang, Y. Tsao, H.-M. Wang, S. L. Maguer, M. Becker, F. Henderson, R. Clark, Y. Zhang, Q. Wang, Y. Jia, K. Onuma, K. Mushika, T. Kaneda, Y. Jiang, L.-J. Liu, Y.-C. Wu, W.-C. Huang, T. Toda, K. Tanaka, H. Kameoka, I. Steiner, D. Matrouf, J.-F. Bonastre, A. Govender, S. Ronanki, J.-X. Zhang, and Z.-H. Ling, "ASVspoof 2019: A large-scale public database of synthetized, converted and replayed speech," *Computer Speech & Language* **64**, 101114 (2020).

[6] T. Kinnunen, H. Delgado, N. Evans, K. A. Lee, V. Vestman, A. Nautsch, M. Todisco, X. Wang, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "Tandem Assessment of Spoofing Countermeasures and Automatic Speaker Verification: Fundamentals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **28**, 2195–2210 (2020).

[7] A. Sholokhov, T. Kinnunen, V. Vestman, and K. A. Lee, "Extrapolating False Alarm Rates in Automatic Speaker Verification," in *Proc. Interspeech (to appear)* (2020).

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS* (2012), pp. 1097–1105.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proc. ICLR* (2015).

[10] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., "Deep neural networks for

acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine* **29**, 82–97 (2012).

[11] A. K. Jain and S. Z. Li, *Handbook of face recognition*, Vol. 1, (Springer, 2011).

[12] D. Yu and L. Deng, *Automatic speech recognition.* (Springer, 2016).

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

[14] J. Lemley, S. Bazrafkan, and P. Corcoran, "Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision.," *IEEE Consumer Electronics Magazine* **6**, 48–56 (2017).

[15] A. K. Jain, J. Feng, and K. Nandakumar, "Fingerprint matching," *Computer* **43**, 36–44 (2010).

[16] J. H. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal processing magazine* **32**, 74–99 (2015).

[17] R. González Hautamäki, *Human-induced voice modification and speaker recognition: Automatic, perceptual and acoustic perspectives*, PhD thesis (University of Eastern Finland, 2017).

[18] J. P. Campbell, W. Shen, W. M. Campbell, R. Schwartz, J.-F. Bonastre, and D. Matrouf, "Forensic speaker recognition," *IEEE Signal Processing Magazine* **26**, 95–103 (2009).

[19] V. Ramasubramanian, "Speaker spotting: Automatic telephony surveillance for homeland security," in *Forensic Speaker Recognition* (Springer, 2012), pp. 427–468.

[20] P. Tresadern, T. F. Cootes, N. Poh, P. Matejka, A. Hadid, C. Levy, C. McCool, and S. Marcel, "Mobile biometrics: Combined face and voice verification for a mobile platform," *IEEE pervasive computing* 79–87 (2013).

[21] S. Larson, "Google Home now recognizes your individual voice," `https://money.cnn.com/2017/04/20/technology/google-home-voice-recognition/index.html` [Accessed: 10 June 2020] (2017).

[22] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker diarization: A review of recent research," *IEEE Transactions on Audio, Speech, and Language Processing* **20**, 356–370 (2012).

[23] S. E. Tranter and D. A. Reynolds, "An overview of automatic speaker diarization systems," *IEEE Transactions on audio, speech, and language processing* **14**, 1557–1565 (2006).

[24] S. Cumani and P. Laface, "Factorized sub-space estimation for fast and memory effective i-vector extraction," *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **22**, 248–259 (2013).

[25] L. Xu, K. A. Lee, H. Li, and Z. Yang, "Generalizing I-vector estimation for rapid speaker recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **26**, 749–759 (2018).

[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks* **61**, 85–117 (2015).

[27] S. A. Zollinger and H. Brumm, "The evolution of the Lombard effect: 100 years of psychoacoustic research," *Behaviour* **148**, 1173–1198 (2011).

[28] D. Garcia-Romero, X. Zhang, A. McCree, and D. Povey, "Improving speaker recognition performance in the domain adaptation challenge using deep neural networks," in *Proc. SLT* (2014), pp. 378–383.

[29] S. Shum, D. Reynolds, D. Garcia-Romero, and A. Mccree, "Unsupervised Clustering Approaches for Domain Adaptation in Speaker Recognition Systems," in *Proc. Odyssey* (2014), pp. 265–272.

[30] V. Vestman, "Modeling temporal characteristics of line spectral frequencies with an application to automatic speaker verification," MSc thesis (University of Eastern Finland, 2016).

[31] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," in *Proc. ICLR* (2018), pp. 1021–1028.

[32] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech & Language* **60**, 101027 (2020).

[33] Y. Fan, J. W. Kang, L. T. Li, K. C. Li, H. L. Chen, S. T. Cheng, P. Y. Zhang, Z. Y. Zhou, Y. Q. Cai, and D. Wang, "CN-Celeb: A Challenging Chinese Speaker Recognition Dataset," in *Proc. ICASSP* (2020), pp. 7604–7608.

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

[35] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *Proc. ICASSP* (2018), pp. 5329–5333.

[36] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE transactions on speech and audio processing* **2**, 578–589 (1994).

[37] R. Saeidi, J. Pohjalainen, T. Kinnunen, and P. Alku, "Temporally weighted linear prediction features for tackling additive noise in speaker verification," *IEEE Signal Processing Letters* **17**, 599–602 (2010).

[38] R. Jones, "Voice recognition: is it really as secure as it sounds?," `https://www.theguardian.com/money/2018/sep/22/voice-recognition-is-it-really-as-secure-as-it-sounds` [Accessed: 10 June 2020] (2018).

[39] Z. Wu, S. Gao, E. S. Cling, and H. Li, "A study on replay attack and anti-spoofing for text-dependent speaker verification," in *Proc. APSIPA* (2014), pp. 1–5.

[40] T. Nakamura, Y. Saito, S. Takamichi, Y. Ijima, and H. Saruwatari, "V2S attack: building DNN-based voice conversion from automatic speaker verification," in *Proc. 10th ISCA Speech Synthesis Workshop* (2019), pp. 161–165.

[41] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Proc. Interspeech* (2015), pp. 2037–2041.

[42] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," in *Proc. Interspeech* (2017), pp. 2–6.

[43] G. Lavrentyeva, S. Novoselov, A. Tseren, M. Volkova, A. Gorlanov, and A. Kozlov, "STC Antispoofing Systems for the ASVspoof2019 Challenge," in *Proc. Interspeech* (2019), pp. 1033–1037.

[44] B. Chettri, D. Stoller, V. Morfi, M. A. M. Ramírez, E. Benetos, and B. L. Sturm, "Ensemble Models for Spoofing Detection in Automatic Speaker Verification," in *Proc. Interspeech* (2019), pp. 1018–1022.

[45] J. Pelecanos, U. Chaudhari, and G. Ramaswamy, "Compensation of utterance length for speaker verification," in *Proc. Odyssey* (2004).

[46] P. Rajan, A. Afanasyev, V. Hautamäki, and T. Kinnunen, "From single to multiple enrollment i-vectors: Practical PLDA scoring variants for speaker verification," *Digital Signal Processing* **31**, 93–101 (2014).

[47] G. Liu and J. H. Hansen, "An investigation into back-end advancements for speaker recognition in multi-session and noisy enrollment scenarios," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **22**, 1978–1992 (2014).

[48] J. Fortuna, P. Sivakumaran, A. M. Ariyaeeinia, and A. Malegaonkar, "Relative effectiveness of score normalisation methods in open-set speaker identification," in *Proc. Odyssey* (2004).

[49] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing* **10**, 19–41 (2000).

[50] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing* **19**, 788–798 (2010).

[51] S. Ioffe, "Probabilistic linear discriminant analysis," in *European Conference on Computer Vision* (Springer, 2006), pp. 531–542.

[52] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Proc. Interspeech* (2009).

[53] C. M. Bishop, *Pattern recognition and machine learning* (Springer, 2006).

[54] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing* **28**, 357–366 (1980).

[55] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE* **63**, 561–580 (1975).

[56] M.-W. Mak and H.-B. Yu, "A study of voice activity detection techniques for NIST speaker recognition evaluations," *Computer Speech & Language* **28**, 295–313 (2014).

[57] A. Sholokhov, M. Sahidullah, and T. Kinnunen, "Semi-supervised speech activity detection with an application to automatic speaker verification," *Computer Speech & Language* **47**, 132–156 (2018).

[58] L. Ferrer, M. Graciarena, and V. Mitra, "A phonetically aware system for speech activity detection," in *Proc. ICASSP* (2016), pp. 5710–5714.

[59] A. V. Oppenheim and R. W. Schafer, *Digital signal processing* (Prentice Hall, 1975).

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

[61] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition* (Prentice Hall, 1993).

[62] J. C. Brown, "Calculation of a constant Q spectral transform," *The Journal of the Acoustical Society of America* **89**, 425–434 (1991).

[63] M. Grimaldi and F. Cummins, "Speaker identification using instantaneous frequencies," *IEEE transactions on audio, speech, and language processing* **16**, 1097–1111 (2008).

[64] S. Thomas, S. Ganapathy, and H. Hermansky, "Recognition of reverberant speech using frequency domain linear prediction," *IEEE Signal Processing Letters* **15**, 681–684 (2008).

[65] S. Ganapathy, S. H. Mallidi, and H. Hermansky, "Robust feature extraction using modulation filtering of autoregressive models," *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **22**, 1285–1295 (2014).

[66] B. J. Shannon and K. K. Paliwal, "A comparative study of filter bank spacing for speech recognition," in *Microelectronic engineering research conference*, Vol. 41 (2003).

[67] T. Kinnunen, M. J. Alam, P. Matejka, P. Kenny, J. Cernockỳ, and D. D. O'Shaughnessy, "Frequency warping and robust speaker verification: a comparison of alternative mel-scale representations.," in *Proc. Interspeech* (2013), pp. 3122–3126.

[68] B. S. Atal, "Automatic recognition of speakers from their voices," *Proceedings of the IEEE* **64**, 460–475 (1976).

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

[70] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *the Journal of the Acoustical Society of America* **87**, 1738–1752 (1990).

[71] C. Nadeu, J. Hernando, and M. Gorricho, "On the decorrelation of filter-bank energies in speech recognition," in *Fourth European Conference on Speech Communication and Technology* (1995).

[72] H. Aghajan, J. C. Augusto, and R. L.-C. Delgado, *Human-centric interfaces for ambient intelligence* (Academic Press, 2009).

[73] A. E. Rosenberg, C.-H. Lee, and F. K. Soong, "Cepstral channel normalization techniques for HMM-based speaker verification," in *Third International Conference on Spoken Language Processing* (1994).

[74] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Communication* **25**, 133–147 (1998).

[75] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on acoustics, speech, and signal processing* **29**, 254–272 (1981).

[76] S. Young, E. Gunnar, G. Mark, T. Hain, and D. Kershaw, "The HTK Book version 3.5 alpha," *Cambridge University* (2015).

[77] P. Matějka, O. Plchot, O. Glembek, L. Burget, J. Rohdin, H. Zeinali, L. Mošner, A. Silnova, O. Novotnỳ, M. Diez, et al., "13 years of speaker recognition research at BUT, with longitudinal analysis of NIST SRE," *Computer Speech & Language* **63**, 101035 (2020).

[78] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, L. P. García-Perera, F. Richardson, R. Dehak, et al., "State-of-the-art speaker recognition with neural network embeddings in nist sre18 and speakers in the wild evaluations," *Computer Speech & Language* **60**, 101026 (2020).

[79] C. E. Shannon, "Communication in the presence of noise," in *Proc. of the IRE*, Vol. 37 (1949), pp. 10–21.

[80] B. Gold, N. Morgan, and D. Ellis, *Speech and audio signal processing: processing and perception of speech and music* (John Wiley & Sons, 2011).

[81] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A Large-Scale Speaker Identification Dataset," in *Proc. Interspeech* (2017), pp. 2616–2620.

[82] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep Speaker Recognition," in *Proc. Interspeech* (2018), pp. 1086–1090.

[83] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The Speakers in the Wild (SITW) speaker recognition database," in *Proc. Interspeech* (2016), pp. 818–822.

[84] K. A. Lee, A. Larcher, G. Wang, P. Kenny, N. Brümmer, D. v. Leeuwen, H. Aronowitz, M. Kockmann, C. Vaquero, B. Ma, et al., "The RedDots data collection for speaker recognition," in *Proc. Interspeech* (2015).

[85] A. Larcher, K. A. Lee, B. Ma, and H. Li, "Text-dependent speaker verification: Classifiers, databases and RSR2015," *Speech Communication* **60**, 56–77 (2014).

[86] "NIST 2016 Speaker Recognition Evaluation Plan," (2016 [accessed May 19, 2020]), `https://www.nist.gov/system/files/documents/2016/10/07/sre16_eval_plan_v1.3.pdf`.

[87] "NIST 2018 Speaker Recognition Evaluation Plan," (2018 [accessed January 24, 2020]), `https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf`.

[88] "NIST 2019 Speaker Recognition Evaluation Plan," (2019 [accessed January 24, 2020]), `https://www.nist.gov/system/files/documents/2019/08/16/2019_nist_multimedia_speaker_recognition_evaluation_plan_v3.pdf`.

[89] J. S. Chung, A. Nagrani, E. Coto, W. Xie, M. McLaren, D. A. Reynolds, and A. Zisserman, "VoxSRC 2019: The first VoxCeleb Speaker Recognition Challenge," *arXiv preprint arXiv:1912.02522* (2019).

[90] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The 2016 Speakers in the Wild Speaker Recognition Evaluation.," in *Proc. Interspeech* (2016), pp. 823–827.

[91] M. Przybocki and A. F. Martin, "NIST speaker recognition evaluation chronicles," in *Proc. Odyssey* (2004).

[92] M. K. Nandwana, J. Van Hout, M. McLaren, C. Richey, A. Lawson, and M. A. Barrios, "The voices from a distance challenge 2019 evaluation plan," *arXiv preprint arXiv:1902.10828* (2019).

[93] H. Zeinali, K. A. Lee, J. Alam, and L. Burget, "Short-duration Speaker Verification (SdSV) Challenge 2020: the Challenge Evaluation Plan," (2019).

[94] A. F. Martin, G. R. Doddington, T. Kamm, M. Ordowski, and M. A. Przybocki, "The DET curve in assessment of detection task performance," in *EUROSPEECH* (1997).

[95] Z. Tu, "Learning generative models via discriminative approaches," in *2007 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2007), pp. 1–8.

[96] W. Cai, Z. Cai, X. Zhang, X. Wang, and M. Li, "A novel learnable dictionary encoding layer for end-to-end language identification," in *Proc. ICASSP* (2018), pp. 5189–5193.

[97] N. Chen, J. Villalba, and N. Dehak, "Tied mixture of factor analyzers layer to combine frame level representations in neural speaker embeddings," *Proc. Interspeech* 2948–2952 (2019).

[98] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13* **14**, 28–29 (2005).

[99] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE signal processing letters* **13**, 308–311 (2006).

[100] S. R. Madikeri, "A fast and scalable hybrid FA/PPCA-based framework for speaker recognition," *Digital Signal Processing* **32**, 137–145 (2014).

[101] K. P. Murphy, *Machine learning: a probabilistic perspective* (MIT press, 2012).

[102] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE transactions on speech and audio processing* **2**, 291–298 (1994).

[103] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)* **39**, 1–22 (1977).

[104] P. Kenny, "A small footprint i-vector extractor," in *Proc. Odyssey* (2012), pp. 1–6.

[105] Y. Jiang, K. Lee, Z. Tang, B. Ma, A. Larcher, and H. Li, "PLDA modeling in i-vector and supervector space for speaker verification," (2012).

[106] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **61**, 611–622 (1999).

[107] A. Sizov, K. A. Lee, and T. Kinnunen, "Unifying probabilistic linear discriminant analysis variants in biometric authentication," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (Springer, 2014), pp. 464–475.

[108] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision* (IEEE, 2007), pp. 1–8.

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

[110] N. Brümmer and E. De Villiers, "The speaker partitioning problem," in *Proc. Odyssey* (2010), pp. 194–201.

[111] P. Kenny, "Bayesian speaker verification with heavy-tailed priors.," in *Proc. Odyssey* (2010).

[112] A. Sizov, *Secure and robust speech representations for speaker and language recognition*, PhD thesis (University of Eastern Finland, 2017).

[113] L. Chen, K. A. Lee, B. Ma, W. Guo, H. Li, and L. R. Dai, "Local variability vector for text-independent speaker verification," in *The 9th International Symposium on Chinese Spoken Language Processing* (IEEE, 2014), pp. 54–58.

[114] L. Chen, K. A. Lee, B. Ma, W. Guo, H. Li, and L.-R. Dai, "Exploration of local variability in text-independent speaker verification," *Journal of Signal Processing Systems* **82**, 217–228 (2016).

[115] N. Dehak, *Discriminative and generative approaches for long-and short-term speaker characteristics modeling: application to speaker verification*, PhD thesis (École de technologie supérieure, 2009).

[116] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE transactions on speech and audio processing* **13**, 345–354 (2005).

[117] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification.," in *Proc. Interspeech* (2017), pp. 999–1003.

[118] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature* **521**, 436–444 (2015).

[119] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).

[120] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research* **18**, 5595–5637 (2017).

[121] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 315–323.

[122] R. E. Wengert, "A simple automatic derivative evaluation program," *Communications of the ACM* **7**, 463–464 (1964).

[123] S. Linnainmaa, "The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (In Finnish: Algoritmin kumulatiivinen pyöristysvirhe yksittäisten pyöristysvirheiden Taylor-kehitelmänä," *Master's Thesis (in Finnish), Univ. Helsinki* (1970), `http://www.idsia.ch/~juergen/linnainmaa1970thesis.pdf`.

[124] A. Cauchy, "Méthode générale pour la résolution des systemes d'équations simultanées," *Comp. Rend. Sci. Paris* **25**, 536–538 (1847).

[125] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics* **4**, 1–17 (1964).

[126] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $O(1/k^2)$," in *Dokl. akad. nauk Sssr*, Vol. 269 (1983), pp. 543–547.

[127] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning* (2013), pp. 1139–1147.

[128] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning* **4**, 26–31 (2012).

[129] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR* (2015).

[130] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc. NIPS* (1992), pp. 950–957.

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

[132] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Vol. 37, ICML'15 (2015), p. 448–456.

[133] H. Zeinali, L. Burget, J. Rohdin, T. Stafylakis, and J. H. Cernocky, "How to improve your speaker embeddings extractor in generic toolkits," in *Proc. ICASSP* (2019), pp. 6141–6145.

[134] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," in *Proc. NIPS* (2018), pp. 2483–2493.

[135] J. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, and K. Neymeyr, "Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization," in *Proceedings of Machine Learning Research*, Vol. 89 (2019), pp. 806–815.

[136] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. NIPS* (2016), pp. 901–909.

[137] Y. Song, B. Jiang, Y. Bao, S. Wei, and L.-R. Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters* **49**, 1569–1570 (2013).

[138] A. K. Sarkar, C.-T. Do, V.-B. Le, and C. Barras, "Combination of cepstral and phonetically discriminative features for speaker verification," *IEEE Signal Processing Letters* **21**, 1040–1044 (2014).

[139] P. Matejka, L. Zhang, T. Ng, O. Glembek, J. Ma, B. Zhang, and S. H. Mallidi, "Neural Network Bottleneck Features for Language Identification," in *Proc. Odyssey* (2014), pp. 299–304.

[140] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Proc. ICASSP* (2014), pp. 4052–4056.

[141] J. S. Chung, J. Huh, S. Mun, M. Lee, H. S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In defence of metric learning for speaker recognition," *arXiv preprint arXiv:2003.11982* (2020).

[142] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Proc. ICASSP* (2014), pp. 1695–1699.

[143] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (IEEE, 2015), pp. 92–97.

[144] M.-Y. Hwang and X. Huang, "Subphonetic modeling with Markov states — Senone," in *Proc. ICASSP*, Vol. 1 (1992), pp. 33–36.

[145] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[146] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing* **37**, 328–339 (1989).

[147] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science* **313**, 504–507 (2006).

[148] F. Richardson, D. Reynolds, and N. Dehak, "A unified deep neural network for speaker and language recognition," in *Proc. Interspeech* (2015), pp. 1146–1150.

[149] D. Garcia-Romero, G. Sell, and A. Mccree, "MagNetO: X-vector Magnitude Estimation Network plus Offset for Improved Speaker Recognition," in *Proc. Odyssey* (2020), pp. 1–8.

[150] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition* **90**, 119–133 (2019).

[151] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level Aggregation for Speaker Recognition in the Wild," in *Proc. ICASSP* (2019), pp. 5791–5795.

[152] Z. Huang, S. Wang, and K. Yu, "Angular Softmax for Short-Duration Text-independent Speaker Verification," in *Proc. Interspeech* (2018), pp. 3623–3627.

[153] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin Matters: Towards More Discriminative Deep Neural Network Embeddings for Speaker Recognition," in *Proc. APSIPA ASC* (2019), pp. 1652–1656.

[154] C. Zhang, K. Koishida, and J. H. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **26**, 1633–1644 (2018).

[155] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized End-to-End Loss for Speaker Verification," in *Proc. ICASSP* (2018), pp. 4879–4883.

[156] J. Herre and J. D. Johnston, "Enhancing the performance of perceptual audio coders by using temporal noise shaping (TNS)," in *Audio Engineering Society Convention 101* (Audio Engineering Society, 1996).

[157] R. Kumaresan and A. Rao, "Model-based approach to envelope and positive instantaneous frequency estimation of signals with speech applications," *The Journal of the Acoustical Society of America* **105**, 1912–1924 (1999).

[158] M. Athineos and D. P. Ellis, "Autoregressive modeling of temporal envelopes," *IEEE Transactions on Signal Processing* **55**, 5237–5245 (2007).

[159] M. G. Hall, A. V. Oppenheim, and A. S. Willsky, "Time-varying parametric modeling of speech," *Signal processing* **5**, 267–285 (1983).

[160] D. Rudoy, T. F. Quatieri, and P. J. Wolfe, "Time-varying autoregressions in speech: Detection theory and applications," *IEEE Transactions on audio, Speech, and Language processing* **19**, 977–989 (2010).

[161] C. Magi, J. Pohjalainen, T. Bäckström, and P. Alku, "Stabilised weighted linear prediction," *Speech Communication* **51**, 401–411 (2009).

[162] M. Athineos, H. Hermansky, and D. P. Ellis, "PLP^2: Autoregressive modeling of auditory-like 2-D spectro-temporal patterns," in *Workshop on Statistical and Perceptual Audio Processing (SAPA)* (2004).

[163] H. Zhang, J. Xue, and K. Dana, "Deep ten: Texture encoding network," in *Proc. of the IEEE conference on computer vision and pattern recognition* (2017), pp. 708–717.

[164] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition* (2016), pp. 5297–5307.

[165] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 7132–7141.

[166] A. G. Roy, N. Navab, and C. Wachinger, "Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2018), pp. 421–429.

[167] F. Cummins, M. Grimaldi, T. Leonard, and J. Simko, "The chains corpus: Characterizing individual speakers," in *Proc of SPECOM*, Vol. 6 (Citeseer, 2006), pp. 431–435.

[168] C. Chen, J. Han, and Y. Pan, "Speaker Verification via Estimating Total Variability Space Using Probabilistic Partial Least Squares.," in *Proc. Interspeech* (2017), pp. 1537–1541.

[169] Y. Lei and J. H. Hansen, "Speaker recognition using supervised probabilistic principal component analysis," in *Proc. Interspeech* (2010), pp. 382–385.

[170] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NIPS* (2019), pp. 8024–8035.

[171] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al., "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding* (IEEE Signal Processing Society, 2011).

[172] T. Kinnunen, K. A. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "t-DCF: a Detection Cost Function for the Tandem Assessment of Spoofing Countermeasures and Automatic Speaker Verification," in *Proc. Odyssey* (2018), pp. 312–319.

[173] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP* (2015), pp. 5206–5210.

[174] D. B. Paul and J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proceedings of the workshop on Speech and Natural Language* (Association for Computational Linguistics, 1992), pp. 357–362.

[175] "AVOID corpus," (2019), `http://urn.fi/urn:nbn:fi:lb-2018060621`.

[176] M. Cotescu, T. Drugman, G. Huybrechts, J. Lorenzo-Trueba, and A. Moinet, "Voice Conversion for Whispered Speech Synthesis," *IEEE Signal Processing Letters* **27**, 186–190 (2020).

[177] S. Seshadri, L. Juvela, J. Yamagishi, O. Räsänen, and P. Alku, "Cycle-consistent adversarial networks for non-parallel vocal effort based speaking style conversion," in *Proc. ICASSP* (IEEE, 2019), pp. 6835–6839.

[178] J. Rohdin, A. Silnova, M. Diez, O. Plchot, P. Matějka, L. Burget, and O. Glembek, "End-to-end DNN based text-independent speaker recognition for long and short utterances," *Computer Speech & Language* **59**, 22–35 (2020).

[179] J. weon Jung, H.-S. Heo, J. ho Kim, H. jin Shim, and H.-J. Yu, "RawNet: Advanced End-to-End Deep Neural Network Using Raw Waveforms for Text-Independent Speaker Verification," in *Proc. Interspeech* (2019), pp. 1268–1272.

[180] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the State of Neural Network Pruning?," in *Proceedings of Machine Learning and Systems* (2020), pp. 129–146.

[181] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. E. Yu, T. Xu, K. Chen, P. Vajda, and J. Gonzalez, "FBNetV2: Differentiable Neural Architecture Search for Spatial and Channel Dimensions," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 12962–12971.

[182] K. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence* **1**, 24–35 (2019).

A ERRATA

In Publication **III**, there is an error in the formula regarding the computation of MAP adapted GMM mean vectors. The erroneous formula is

$$\hat{\boldsymbol{\mu}}_c = \alpha_c \boldsymbol{f}_c + (1 - \alpha_c)\boldsymbol{\mu}_c.$$

Here the first order statistics $\boldsymbol{f}_c$ should be divided by the zeroth order statistics (soft counts) n_c to obtain the correct formula:

$$\hat{\boldsymbol{\mu}}_c = \alpha_c \frac{\boldsymbol{f}_c}{n_c} + (1 - \alpha_c)\boldsymbol{\mu}_c.$$

In Publication **IX**, the correct form of equation $\boldsymbol{m} = (\boldsymbol{m}_1^\mathsf{T}, \boldsymbol{m}_2^\mathsf{T}, \ldots \boldsymbol{m}_T^\mathsf{T})^\mathsf{T}$, which is shown below equation (5), is $\boldsymbol{m} = (\boldsymbol{m}_1^\mathsf{T}, \boldsymbol{m}_2^\mathsf{T}, \ldots, \boldsymbol{m}_C^\mathsf{T})^\mathsf{T}$.

Paper I

INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Time-Varying Autoregressions for Speaker Verification in Reverberant Conditions

Ville Vestman[1], *Dhananjaya Gowda*[2], *Md Sahidullah*[1], *Paavo Alku*[3], *Tomi Kinnunen*[1]

[1]School of Computing, University of Eastern Finland, Finland
[2]DMC R&D Center, Samsung Electronics, Seoul, Korea
[3]Department of Signal Processing and Acoustics, Aalto University, Finland

{vvestman, sahid, tkinnu}@cs.uef.fi, d.gowda@samsung.com, paavo.alku@aalto.fi

Abstract

In poor room acoustics conditions, speech signals received by a microphone might become corrupted by the signals' delayed versions that are reflected from the room surfaces (e.g. wall, floor). This phenomenon, reverberation, drops the accuracy of automatic speaker verification systems by causing mismatch between the training and testing. Since reverberation causes temporal smearing to the signal, one way to tackle its effects is to study robust feature extraction, particularly based on long-time temporal feature extraction. This approach has been adopted previously in the form of 2-dimensional autoregressive (2DAR) feature extraction scheme by using frequency domain linear prediction (FDLP). In 2DAR, FDLP processing is followed by time domain linear prediction (TDLP). In the current study, we propose modifying the latter part of the 2DAR feature extraction scheme by replacing TDLP with time-varying linear prediction (TVLP) to add an extra layer of temporal processing. Our speaker verification experiments using the proposed features with the text-dependent RedDots corpus show small but consistent improvements in clean and reverberant conditions (up to 6.5%) over the 2DAR features and large improvements over the MFCC features in reverberant conditions (up to 46.5%).

Index Terms: speaker recognition, autoregressive modeling, autocorrelation domain time-varying linear prediction

1. Introduction

An automatic speaker verification system is said to be *robust* if it tolerates external distortion caused by environmental noise or transmission channel, or internal signal variation caused by, for example, different speaking styles. A large part of previous speaker verification studies have focused on robustness with respect to noise (e.g. [1]), while robustness with respect to varying room acoustics conditions has remained less studied. Room acoustics, however, might have a large effect on the accuracy of a speaker recognition system: If the training and testing data are recorded in different room environments, there will be a mismatch between the training and testing which will degrade recognition accuracy. Room acoustics affect speech signals particularly in the form of *reverberation* [2]: the signal received by a microphone becomes a sum of the direct component and its delayed versions that arrive at the microphone after being reflected from the surfaces (e.g. walls, floor, ceiling) of the room. Mismatch caused by reverberation can in principle be tackled by modifying the speaker verification front-end or back-end. In our view, advances on both sides are necessary to reach the best possible performance. On the back-end side, techniques such as multi-condition training [3], where multiple speaker models for different reverberation conditions are created for each speaker, can be utilized. In the current study, however, we focus on the system front-end by studying features that aim at reducing the mismatch caused by reverberation.

Speech features robust to reverberation have been previously investigated in a few speaker verification studies. In [4], the use of *locally normalized cepstral coefficients* (LNCCs) was studied. LNCC features modify the conventional MFCC features by using an additional filterbank to perform local normalization in the spectral domain. LNCCs were found to improve the recognition accuracy particularly when reverberation was severe. A different approach, based on the *blind spectral weighting* (BSW) technique, was proposed in [5] to handle the reverberation mismatch. In addition to these two previous methods, there is a family of reverberation-robust features based on smoothing of subband Hilbert envelopes. For example, *mean Hilbert envelope coefficient* (MHEC) feature extraction scheme, proposed in [6], takes advantage of low-pass filtering of Hilbert envelopes of Gammatone filterbank outputs. Smoothing of Hilbert envelopes can also be conducted using *frequency domain linear prediction* (FDLP) [7, 8, 9], a method to compute all-pole estimates for Hilbert envelopes. FDLP processing can be used on its own [8] or in conjunction with *time domain linear prediction* (TDLP) [10]. The technique where FDLP is followed by TDLP, known as *2-dimensional autoregressive model (2DAR)*, has been reported to provide better speaker verification results in reverberant conditions than when using FDLP alone [10]. Besides being efficient in tackling the reverberation mismatch problem, 2DAR processing has been found to improve verification in the presence of background noise as well [10].

In this study, we propose to modify the 2DAR model by replacing TDLP with *time-varying linear prediction* (TVLP) [11, 12] which is a generalization of conventional *linear prediction* (LP) [13]. TVLP can be used to analyze non-stationarity of speech signals by allowing the underlying all-pole model to be time-varying. In TVLP, temporal trajectories of the all-pole filter coefficients are represented with basis functions such as polynomials or trigonometric functions. This introduces an additional temporal constraint to 2DAR models that we hypothesize to be useful in tackling reverberation mismatch between training and testing.

Our contributions are as follows: First, we present a modification of TVLP that enables us to apply TVLP in the autocorrelation domain. Traditionally, TVLP is applied in the time-domain but to use TVLP after FDLP, it is necessary to modify the model to be applicable for spectro-temporal representations. Second, we modify the 2DAR model by replacing TDLP with TVLP. Third, we conduct speaker verification experiments with the recent text-dependent RedDots corpus to compare the proposed features with the 2DAR and MFCC features. Finally, we study the effect of RASTA filtering [14] combined with the 2DAR model, which was not included in [10].

This study was partly funded by Academy of Finland projects #284671 and #288558.

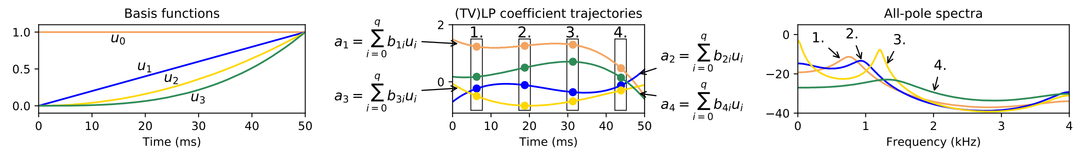


Figure 1: *An example of time-varying linear predictive (TVLP) modeling. The graph on the left shows the monomial basis functions used in TVLP-modeling of the predictor filter coefficient trajectories of 50 ms long speech signal depicted in the middle graph. In this illustration, prediction order of 4 is used, leading to only four trajectories. Graph on the right shows examples of four all-pole spectra that are all obtained from within the single 50 ms long TVLP window.*

2. Time-varying linear prediction

2.1. Classical time-domain formulation

In conventional LP analysis [13], the current speech sample $x[n]$ is predicted as a linear weighted sum of the past p samples given by $\hat{x}[n] = -\sum_{k=1}^{p} a_k x[n-k]$ where $\{a_k\}_{k=1}^{p}$ are the predictor coefficients. The solution to this formulation can be obtained by minimizing the cost function $E = \sum_n e^2[n]$, where $e[n] = x[n] - \hat{x}[n]$, which in turn leads to solving a set of normal equations given by

$$\sum_{k=1}^{p} a_k r_{ki} = -r_{0i}, \qquad i = 1, \ldots, p, \tag{1}$$

where r_{ki} denotes the *correlation coefficients* given by

$$r_{ki} = \sum_n x[n-k]x[n-i]. \tag{2}$$

The above formulation provides a piecewise constant approximation of the vocal tract system over short time intervals (frames) by assuming the system to be quasi-stationary. However, the vocal tract is a slowly but continuously varying system and therefore better modeled using *time-varying* linear prediction (TVLP) analysis over longer time intervals. TVLP introduces a time-continuity constraint on the predictor filter coefficients by expressing the prediction as

$$\hat{x}[n] = -\sum_{k=1}^{p} a_k[n]x[n-k] \tag{3}$$

where $a_k[n]$ denotes the k^{th} time-varying filter coefficient that is approximated as a linear combination of $q+1$ basis functions $u_i[n]$ as follows:

$$a_k[n] = \sum_{i=0}^{q} b_{ki}u_i[n]. \tag{4}$$

Different sets of basis functions such as monomials, trigonometric functions, or Legendre polynomials can be used for the approximation. In this paper, a simple monomial basis $u_i[n] = n^i$, $i = 0, \ldots, 3$, is adopted. An illustration of TVLP analysis using monomial basis functions is given in Figure 1.

Minimization of the cost function with respect to each polynomial coefficient leads to a set of normal equations given by

$$\sum_{k=1}^{p}\sum_{i=0}^{q} b_{ki}c_{ij}[k,l] = -c_{0j}[0,l] \tag{5}$$

for $1 \le l \le p$ and $0 \le j \le q$ [11]. Here $c_{ij}[k,l]$ denotes the *generalized correlation coefficients* defined as

$$c_{ij}[k,l] = \sum_n u_i[n]u_j[n]x[n-k]x[n-l]. \tag{6}$$

2.2. Proposed autocorrelation domain formulation

In several applications, including robust feature extraction, the signal may have been converted into a spectro-temporal representation using a filter-bank or spectrogram analysis. In such a scenario, conventional TVLP modeling of the processed signal would require a reconstruction of the time-domain signal from the spectro-temporal representation. In order to avoid any such reconstruction requiring careful handling of phase information, we propose a new *autocorrelation domain time-varying linear prediction* (AD-TVLP) analysis.

Any given spectro-temporal representation $X(t,f)$ can be converted into a sequence of autocorrelation functions $r_\tau(t) = \int_f X(t,f)\exp(-j2\pi f\tau)df$ by computing the inverse Fourier transform of the power spectrum $X(t,f)$ at each time instant t. Now, the conventional LP analysis can be applied independently on correlation function at each time instant by solving the normal equations similar to that in Eq. (1). However, a time-continuity constraint can be imposed on the LP coefficients derived at each time instant by modifying the normal equations in Eq. (1) to take advantage of the availability of a sequence of autocorrelation functions. The resulting normal equations with the continuity constraint is given by

$$\sum_{k=1}^{p} a_k[n]r_{ki}[n] \approx -r_{0i}[n], \qquad \begin{array}{l} i = 1, \ldots, p, \\ n = 0, \ldots, N-1, \end{array} \tag{7}$$

where $r_{ki}[n]$, $a_k[n]$, and N denote the time-varying autocorrelation coefficients, the time-varying LP coefficients, and the window length for the time-varying analysis, respectively.

Substituting Eq. (4) into Eq. (7), we can compute the least squares solution to the resulting set of linear equations (of the form $\boldsymbol{Rb} = -\boldsymbol{r}$) as follows:

$$\hat{\boldsymbol{b}} = \operatorname*{argmin}_{\boldsymbol{b}} ||\boldsymbol{r} + \boldsymbol{Rb}||_2^2 \tag{8}$$

where

$$\boldsymbol{r} = [r_{01}[0], \ldots, r_{0p}[0], \ldots, r_{01}[N-1], \ldots, r_{0p}[N-1]]^T_{Np\times 1} \tag{9}$$

$$\boldsymbol{b} = [b_{10}, \ldots, b_{1q}, \ldots, b_{p0}, \ldots, b_{pq}]^T_{p(q+1)\times 1} \tag{10}$$

$$\boldsymbol{R} = [R_0, R_1, \ldots, R_{N-1}]^T_{Np\times p(q+1)} \tag{11}$$

where R_n is a $p(q+1) \times p$ matrix whose i^{th} column is given by

$$R_{ni} = \boldsymbol{r}_i[n] \otimes \boldsymbol{u}[n]. \tag{12}$$

Here, $\otimes$ denotes the Kronecker product of $\boldsymbol{r}_i[n] = [r_{1i}[n] \ldots r_{pi}[n]]^T$ and $\boldsymbol{u}[n] = [u_0[n] \ldots u_q[n]]^T$.

3. 2-D autoregressive models

3.1. Background

Two dimensional autoregressive speech modeling (2DAR) was first introduced in 2004 [15]. The 2DAR model provides a refreshing way of building speech spectrograms: instead of applying Fourier transform or AR modeling to short-time windows, the speech signal is first transformed into frequency domain and then AR modeling is applied to frequency windows followed by the usual temporal AR modeling. This idea was first adopted to speaker recognition in [16] and extended later in [17] and [10]. These studies indicate that 2DAR-processed features give consistent and considerable improvements over standard MFCC features for speaker verification in noisy conditions without compromising performance in clean conditions.

Autoregressive modeling is also known as LP modeling and hence its applications in frequency and time domain are known as *frequency domain linear prediction* (FDLP) and *time domain linear prediction* (TDLP), respectively. The former is less known, but nonetheless, it is the key concept behind the 2DAR model allowing temporal processing of speech without first splitting the signal into short-time windows.

The left side of Figure 2 shows the steps for 2DAR-processing of speech. The first step is to transform the input speech with the discrete cosine transform (DCT) and then to window this DCT-transformed signal into many overlapping frequency bands (we used 100 bands). Then, FDLP is applied to obtain all-pole estimates of Hilbert envelopes of each subband. These envelopes represent signal's energy in the subband-specific frequency range as a function of time, which allows us to form a spectrogram of the speech by stacking the information from all of the envelopes. At this stage, we perform energy integrations over the subband envelopes using 25 ms long Hamming windows at 10 ms intervals. By doing so, the spectrogram is effectively subsampled to obtain a frame rate that is similar to that used in the conventional MFCC feature extraction. This ends the FDLP part of 2DAR processing, where the data is processed along the temporal dimension.

The second part of 2DAR is to apply TDLP to the FDLP-processed spectrogram. The autocorrelation coefficients needed for computing the LP coefficients are obtained from the power spectra by using inverse Fourier transform. As a result of successive application of FLDP and TVLP, we obtain a 2DAR spectrogram that has been processed in both temporal and spectral dimensions and from which we can then extract MFCC features in the usual way.

3.2. Proposed method

We propose a modification to the 2DAR model by replacing TDLP with the autocorrelation domain TVLP. This will, in addition to spectral processing, add an extra constraint for the LP-coefficients in the temporal domain, preventing them to change too abruptly from frame to frame. This effect is demonstrated in Figure 3.

Figure 2 shows the differences between 2DAR and its modified version 2DAR-TVLP. In 2DAR-TVLP, after obtaining the autocorrelation sequences, we proceed by forming “superframes” of autocorrelation sequences by using an 11 frames long window that is shifted one frame at a time. We then apply autocorrelation domain TVLP to each of the superframes. This gives us 11 spectra per superframe and because superframes are shifted 1 autocorrelation sequence at a time, we must select only one spectrum from each superframe to keep the frame rate at the original rate (100 Hz). Thus, we extract MFCCs only from the middle frame of each superframe.

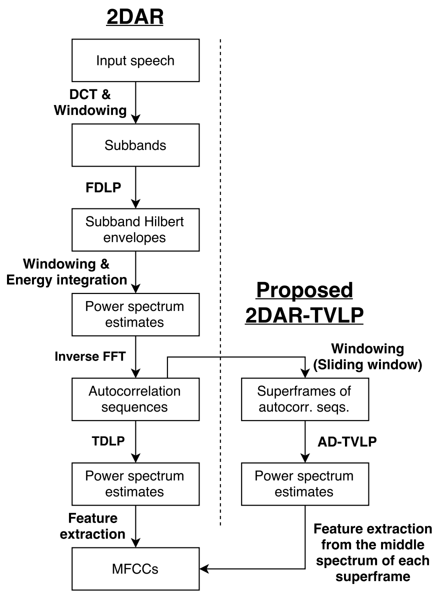

2DAR
Input speech
DCT & Windowing
Subbands
FDLP
Subband Hilbert envelopes
Windowing & Energy integration
Power spectrum estimates
Inverse FFT
Autocorrelation sequences
TDLP
Power spectrum estimates
Feature extraction
MFCCs
Proposed 2DAR-TVLP
Windowing (Sliding window)
Superframes of autocorr. seqs.
AD-TVLP
Power spectrum estimates
Feature extraction from the middle spectrum of each superframe

Figure 2: *Diagram showing the differences between the 2DAR and the 2DAR-TVLP models.*

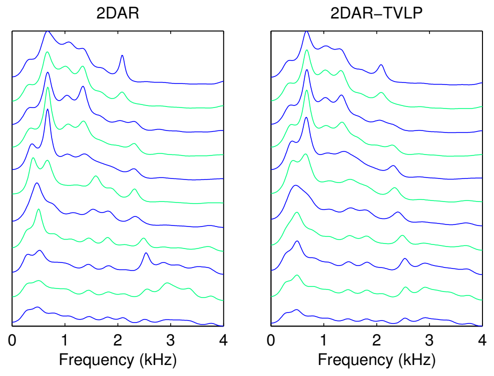


Figure 3: *11 consecutive magnitude spectra (10 ms step) obtained from 2DAR and 2DAR-TVLP processing. TVLP leads to smoother transitions between successive spectra.*

4. Experimental setup

4.1. Speech corpus

We performed speaker verification experiments using the male speakers of common phrase task of the RedDots challenge corpus [18] following the protocol for text-dependent speaker verification. This task consists of 320 pass-phrase specific target speaker models from 35 unique speakers. For enrollment, data of the same pass-phrase from three different sessions are used. The target and non-target speakers utter the same pass-phrase during enrollment and verification. The average duration of each pass-phrase is three seconds. The common phrase task has total 3242 genuine and 120086 impostor trials. As a background data, we used male speakers from TIMIT. Both corpora have a sample rate of 16 kHz.

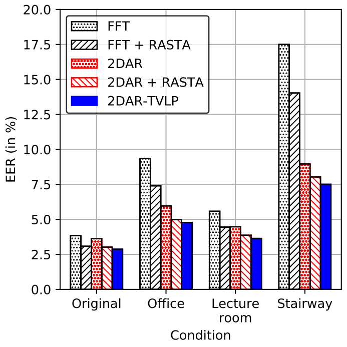


Figure 4: *Speaker verification results for different features in different conditions.*

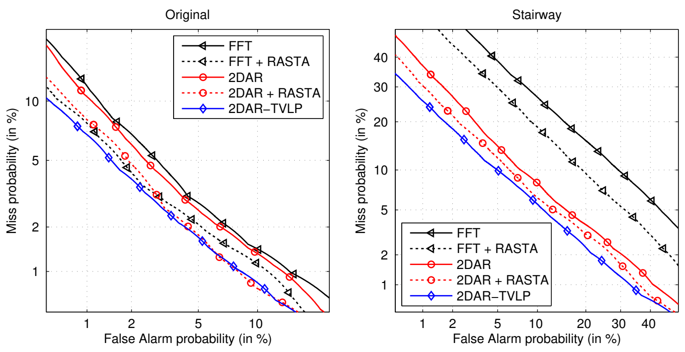


Figure 5: *Detection error tradeoff graphs for the original and stairway-reverberation conditions.*

4.2. Speech reverberation

In addition to the original RedDots data, we experimented on artificially reverberated RedDots data. Reverberation was performed by convolving original speech files with room impulse responses (RIRs) obtained from the Aachen impulse response (AIR) database [19]. We selected three different RIRs, one measured form an office room, second from a lecture room, and the last one from a stairway. Reverberation times (RT60) for these impulse responses are 0.35s, 0.28s, and 0.81s, respectively. Reverberation was only applied to the test data and not to the enrollment or background data.

4.3. Features and classifier

In our speaker verification experiments, we used the MFCC features with a configuration shown in Table 1. These features were computed from spectrograms that were obtained either by Fourier-transforming Hamming-windowed frames or by applying 2DAR or 2DAR-TVLP.

We found that minimum EER was achieved with 2DAR by using prediction orders of 24 and 42 for FDLP and TDLP, respectively. For 2DAR-TVLP, the corresponding optimal prediction orders were 24 and 38. Here, FDLP prediction orders are given for 1 second long segments and they are normalized according to the segment length. Long utterances were split into 3 second long segments before FDLP processing.

We used a classic *Gaussian mixture model – universal background model* (GMM-UBM) system [20], for which we trained a 256-component UBM from the background data. Speaker models were obtained by MAP adapting component means of UBM using relevance factor 3. We chose a lightweight GMM-UBM-based system to conduct rapid parameter experimentation with computationally heavy 2DAR models. As demonstrated in [21], GMM-UBM provides a competitive accuracy on the RedDots data consisting of short utterances.

Table 1: *Configuration of MFCC features.*

Frame length / step	25 ms / 10 ms
Number of cepstral coefficients	19
Center frequency of the first mel-filter	100 Hz
Center frequency of the last mel-filter	5400 Hz
Energy coefficient	Not included
Velocity and acceleration coefficients	Included
RASTA filter pole position (if applied)	$z = 0.97$
Speech activity detection	Energy-based [22]
Feature normalization	Cepstral mean and variance norm. (CMVN)

5. Results

Figure 4 presents the results for the speaker verification experiments on RedDots corpus in terms of EER. We compared performances of traditional FFT-based MFCCs, 2DAR-processed MFCCs, and 2DAR-TVLP-processed MFCCs. Additionally, we studied the effect of cepstral level RASTA filtering on these feature types. In preliminary experiments, we found that RASTA improves performance of FFT and 2DAR features, but does not provide benefit with 2DAR-TVLP features and hence the last combination is omitted from the figure. 2DAR-TVLP outperforms other features in all the tested conditions. Differences between 2DAR-TVLP and 2DAR with RASTA are relatively small but nevertheless consistent. Differences between 2DAR-TVLP and FFT with RASTA are larger and especially so in the reverberant conditions. Table 2 contains the exact numbers for these comparisons for the original and Stairway-reverberation conditions. Detection error tradeoff graphs in Figure 5 reveal that the good performance of 2DAR-TVLP is not restricted only to the proximity of the EER-point.

Table 2: *Speaker verification equal error rates (EER (%)) for the best performing feature configurations in the original and in the stairway-reverberation conditions. The last two columns show relative changes obtained by using the proposed features.*

	(1) FFT + RASTA	**(2) 2DAR + RASTA**	**(3) 2DAR-TVLP**	**(1)→(3) change (%)**	**(2)→(3) change (%)**
Original	3.08	3.02	2.87	-6.8	-5.0
Stairway	14.03	8.03	7.51	-46.5	-6.5

6. Conclusions

We studied the possibility of incorporating time-varying autoregressive modeling to the 2DAR feature extraction scheme to improve speaker verification in reverberant conditions. This required us to develop a new time-varying linear prediction formulation, AD-TVLP, that is applicable to the spectro-temporal representations of signals. We adopted this formulation to the existing 2DAR feature extraction method and obtained promising results. In comparison to the baseline 2DAR and MFCC features, the proposed 2DAR-TVLP features improved speaker verification performance in both original and reverberated test conditions. These promising results encourage us to further explore adopting time-varying autoregressive models for speech feature extraction in adverse conditions.

7. References

[1] J. Ming, T. J. Hazen, J. R. Glass, and D. A. Reynolds, "Robust speaker recognition in noisy conditions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 5, pp. 1711–1723, 2007.

[2] H. Kuttruff, *Room Acoustics*, 5th ed. Spon Press, 2009.

[3] A. R. Avila, M. O. S. Paja, F. J. Fraga, D. D. O'Shaughnessy, and T. H. Falk, "Improving the performance of far-field speaker verification using multi-condition training: the case of GMM-UBM and i-vector systems." in *INTERSPEECH*, 2014, pp. 1096–1100.

[4] V. Poblete, J. P. Escudero, J. Fredes, J. Novoa, R. M. Stern, S. King, and N. B. Yoma, "The use of locally normalized cepstral coefficients (LNCC) to improve speaker recognition accuracy in highly reverberant rooms," *Interspeech 2016*, pp. 2373–2377, 2016.

[5] S. O. Sadjadi and J. H. Hansen, "Blind spectral weighting for robust speaker identification under reverberation mismatch," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 5, pp. 937–945, 2014.

[6] ——, "Hilbert envelope based features for robust speaker identification under reverberant mismatched conditions," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5448–5451.

[7] M. Athineos and D. P. Ellis, "Autoregressive modeling of temporal envelopes," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5237–5245, 2007.

[8] J. Herre and J. D. Johnston, "Enhancing the performance of perceptual audio coders by using temporal noise shaping (TNS)," in *Audio Engineering Society Convention 101*. Audio Engineering Society, 1996.

[9] R. Kumaresan and A. Rao, "Model-based approach to envelope and positive instantaneous frequency estimation of signals with speech applications," *The Journal of the Acoustical Society of America*, vol. 105, no. 3, pp. 1912–1924, 1999.

[10] S. Ganapathy, S. H. Mallidi, and H. Hermansky, "Robust feature extraction using modulation filtering of autoregressive models," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 8, pp. 1285–1295, 2014.

[11] M. G. Hall, A. V. Oppenheim, and A. S. Willsky, "Time-varying parametric modeling of speech," *Signal Processing*, vol. 5, no. 3, pp. 267–285, 1983.

[12] D. Rudoy, T. F. Quatieri, and P. J. Wolfe, "Time-varying autoregressions in speech: Detection theory and applications," *IEEE Transactions on audio, Speech, and Language processing*, vol. 19, no. 4, pp. 977–989, 2011.

[13] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, 1975.

[14] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE transactions on speech and audio processing*, vol. 2, no. 4, pp. 578–589, 1994.

[15] M. Athineos, H. Hermansky, and D. Ellis, "PLP^2: Autoregressive modeling of auditory-like 2-D spectro-temporal patterns," in *ISCA Tutorial and Research Workshop (ITRW) on Statistical and Perceptual Audio Processing*, 2004.

[16] S. Ganapathy, S. Thomas, and H. Hermansky, "Feature extraction using 2-D autoregressive models for speaker recognition." in *Odyssey*, 2012, pp. 229–235.

[17] S. H. R. Mallidi, S. Ganapathy, and H. Hermansky, "Robust speaker recognition using spectro-temporal autoregressive models." in *INTERSPEECH*, 2013, pp. 3689–3693.

[18] "Reddots project," https://sites.google.com/site/thereddotsproject/home, accessed: 2017-03-15.

[19] M. Jeub, M. Schäfer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," *Proceedings of the 16th International Conference on Digital Signal Processing (DSP)*, July 2009.

[20] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[21] H. Delgado, M. Todisco, M. Sahidullah, A. K. Sarkar, N. Evans, T. Kinnunen, and Z.-H. Tan, "Further optimisations of constant Q cepstral processing for integrated utterance and text-dependent speaker verification," in *IEEE Spoken Language Technology (SLT) Workshop*, 2016.

[22] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech communication*, vol. 52, no. 1, pp. 12–40, 2010.

Paper **II**

Speech Communication 99 (2018) 62–79

Contents lists available at ScienceDirect

Speech Communication

journal homepage: www.elsevier.com/locate/specom

Speaker recognition from whispered speech: A tutorial survey and an application of time-varying linear prediction[☆]

Ville Vestman[*,a], Dhananjaya Gowda[b], Md Sahidullah[a], Paavo Alku[c], Tomi Kinnunen[a]

[a] *School of Computing, University of Eastern Finland, Finland*
[b] *DMC R&D Center, Samsung Electronics, Seoul, Korea*
[c] *Department of Signal Processing and Acoustics, Aalto University, Finland*

A R T I C L E   I N F O

Keywords:
Speaker recognition
Speaking style mismatch
Disguise
Whisper
2-Dimensional autoregression (2D-AR)
Time-varying linear prediction (TVLP)

A B S T R A C T

From the available biometric technologies, automatic speaker recognition is one of the most convenient and accessible ones due to abundance of mobile devices equipped with a microphone, allowing users to be authenticated across multiple environments and devices. Speaker recognition also finds use in forensics and surveillance. Due to the acoustic mismatch induced by varied environments and devices of the same speaker, leading to increased number of identification errors, much of the research focuses on compensating for such technology-induced variations, especially using machine learning at the statistical back-end. Another much less studied but at least as detrimental source of acoustic variation, however, arises from mismatched speaking styles induced by the speaker, leading to a substantial performance drop in recognition accuracy. This is a major problem especially in forensics where perpetrators may purposefully disguise their identity by varying their speaking style. We focus on one of the most commonly used ways of disguising one's speaker identity, namely, whispering. We approach the problem of normal-whisper acoustic mismatch compensation from the viewpoint of robust feature extraction. Since whispered speech is intelligible, yet a low-intensity signal and therefore prone to extrinsic distortions, we take advantage of robust, long-term speech analysis methods that utilize slow articulatory movements in speech production. In specific, we address the problem using a novel method, *frequency-domain linear prediction with time-varying linear prediction* (FDLP-TVLP), which is an extension of the *2-dimensional autoregressive* (2DAR) model that allows vocal tract filter parameters to be time-varying, rather than piecewise constant as in classic short-term speech analysis. Our speaker recognition experiments on the whisper subset of the CHAINS corpus indicate that when tested in normal-whisper mismatched conditions, the proposed FDLP-TVLP features improve speaker recognition performance by 7–10% over standard MFCC features in relative terms. We further observe that the proposed FDLP-TVLP features perform better than the FDLP and 2DAR methods for whispered speech.

1. Introduction

Research in *automatic speaker recognition* (Reynolds and Rose, 1995) has focused increasingly on enhancing robustness in adverse conditions induced by background noise, reverberation and low-quality recordings. Many approaches have been studied to tackle these challenges, one of the most successful being the *i-vector* technology (Dehak et al., 2011) used jointly with the *probabilistic discriminant analysis* (PLDA) back-end (Prince and Elder, 2007; Rajan et al., 2014). In addition to the new utterance level features (i-vectors) and back-ends, improvements have been achieved in the first part of the speech processing chain by developing *robust acoustic features* (Liu et al., 2012; Ganapathy et al., 2014; Saeidi et al., 2016). Recent advances in both topics have brought the performance of speaker recognition systems closer to the level expected in applications such as forensics, surveillance, and authentication.

In addition to the environment-related and technology-related acoustic variations, another major, yet much less studied problem arises from *within-speaker* variations caused by differences in speaking styles. Changes in the speaking style occur, for instance, when the speaker shouts (Saeidi et al., 2016) or whispers (Fan and Hansen, 2011b). Current recognition systems that are typically trained with speech of normal speaking style can tolerate only small changes in the speaking style, and reliable speaker recognition has turned out to be very

[☆] This work contains limited portions of Vestman et al. (2017).
[*] Corresponding author.
E-mail address: ville.vestman@uef.fi (V. Vestman).

https://doi.org/10.1016/j.specom.2018.02.009
Received 6 July 2017; Received in revised form 22 December 2017; Accepted 13 February 2018
Available online 09 March 2018

challenging if the mode of speaking changes considerably from normal (Fan and Hansen, 2011b; Hansen et al., 2017; Saeidi et al., 2016). Speaking style mismatched speaker recognition has applications for example in forensics, where recorded audio excerpts can be used as evidence. For instance, a crime might be committed in an agitated state of mind, leading to shouting or screaming. Similarly, a perpetrator might deliberately disguise his or her identity in order to avoid being identified (Hautamäki et al., 2017). In (Masthoff, 1996), whispering was found to be the most common way of changing the mode of speech production to disguise the speaker identity. Furthermore, Masthoff (1996) and Künzel (2000) report that disguise is commonly found in criminal action, especially in blackmailing cases. Whispering can also be used in public places to prevent others from hearing private information or to avoid disturbing others in places where silent behavior is expected. Conversely, people tend to use loud, high-effort voice in noisy environments in order to make their speech more intelligible in background noise. The tendency of the speaker to change his or her speaking style in noisy environments is known as the *Lombard effect* (Junqua, 1993).

As described above, various speaking styles are expected to be encountered in real-world speaker recognition applications. This imposes a considerable challenge to the existing systems whose performance has been shown to drastically decline due to changes in the speaking style (Fan and Hansen, 2011b; Hansen et al., 2017; Saeidi et al., 2016). In order to improve the performance of speaker recognition in real-world scenarios including various speaking styles, the current study focuses on a specific style of speaking, whispering, which differs vastly from normal speech in its acoustic properties. In addition to being lower in intensity, whispered speech lacks the vibration of the vocal folds (even in case of voiced sounds such as vowels) when the sound excitation is generated in the larynx (Ito et al., 2005). In addition to the absence of the vocal fold vibration, it has been observed that whispered vowels show an upward shift in formant frequencies when compared to vowels of the normal speaking style and that whispered consonants show increased spectral flatness (Ito et al., 2005).

In principle, suppressing the unwanted within-speaker variations induced by speaking style mismatch could be addressed using statistical back-end methods. In fact, most modern speaker recognition back-ends (as reviewed in Hansen and Hasan, 2015) include some kind of a within-speaker variation model, intended to quantify the extent of allowed variation in any pair of utterances of the same speaker, before they are considered more likely to have been spoken by different speakers. Dating back to Kenny's pioneering work on *joint factor analysis* (JFA) (Kenny, 2006), which later inspired the i-vector paradigm (Dehak et al., 2011), these techniques are realized as various flavors of subspace models where the between- and within-speaker subspaces are modeled using separate factor loading matrices. To exemplify, the *simplified* PLDA model (Kenny, 2010) assumes a Gaussian within-speaker variation model shared across all the speakers, parameterized as a residual covariance matrix. The hyperparameters of such back-end models are trained off-line using, typically, thousands of utterances from hundreds of *development speakers*. In order to adopt these back-ends for explicit style variation compensation, a corpus is needed that contains, per each development speaker, utterances spoken in various speaking styles. Unfortunately, this kind of speech data is prohibitively expensive and difficult to collect in quantities required by PLDA back-ends. Moreover, to the best of our knowledge, such large corpora are not publicly available at present. For these reasons, and since the present study addresses speaker recognition using short utterances, we adopt instead the classic Gaussian mixture model- universal background model (GMM-UBM) (Reynolds et al., 2000) back-end approach which, in fact, produces competitive accuracy — or even surpasses the i-vector based approach (Li et al., 2016; Zeinali et al., 2017; Liu et al., 2015) — in the duration conditions considered in this study.

Another commonly applied back-end recipe to enhance the speaker recognition accuracy across varied conditions is *multicondition training* (Garcia-Romero et al., 2012; Rajan et al., 2013). It utilizes data obtained from different conditions to prepare the back-end components to expect different variations of the speech data. Again, however, since data collection for multiple conditions takes lots of resources and usually is not a realistic requirement for speaker enrollment, a common practice is to artificially generate data by, for example, digitally adding noise. In case of variation caused by the speaking style (such as whispering), however, generation of realistic artificial data is not easy. Therefore, the current study focuses on an alternative approach, robust extraction of features, to tackle the deteriorating effect caused by the speaking style variation. Feature extraction, as the first step in the speech processing chain of any speaker recognition system, has a key role as it provides inputs to the back-end that can be based, for instance, on the GMM-UBM (Reynolds et al., 2000), i-vectors (Dehak et al., 2011), or *deep neural networks* (DNNs) (Heigold et al., 2016; Snyder et al., 2016). Thus, we find it important to develop and study features that show good performance across a wide variety of settings to make speaker recognition systems less dependent on large amounts of training data from different conditions. To this end, we propose using two recent feature extraction methods (Ganapathy et al., 2014; Vestman et al., 2017) for whispered speech that have already shown good results in other studies.

Traditionally, most features used in speaker recognition are computed from short-term analysis using frames that span about 25 ms of speech (Kinnunen and Li, 2010). While this approach is effective in capturing instantaneous acoustical features of the vocal tract, it ignores long-term properties of speech such as prosody. In addition, the traditional short-term analysis is not capable of taking into account articulation variations, and it lacks other possible benefits of longer-term processing including improved robustness against noise and reverberation (Ganapathy et al., 2014). These limitations of the traditional short-term analysis are important factors to consider especially when dealing with whispered speech as whispering has lower intensity than normal speech (Zhang and Hansen, 2007), which makes it more prone to extrinsic disturbances, such as additive noise. Whispered speech also tends to show widening of formant bandwidths (see Fig. 1), which makes it harder to accurately detect formants. We hypothesize that a better utilization of contextual information observed over long-time frames can be used to improve formant modeling accuracy over standard short-time analysis.

To study feature extraction based on long-term processing, we propose using *2-dimensional autoregressive features* (2DAR) for whispered speech speaker recognition. In the 2DAR scheme, speech is processed in temporal domain before feeding it to the typical short-term feature extraction pipeline. The temporal processing is achieved using *frequency domain linear prediction* (FDLP) (Herre and Johnston, 1996; Kumaresan and Rao, 1999), a method that produces smoothed, parametric time-domain Hilbert envelopes of the individual frequency subbands. The smoothed, parametric representation of the subband Hilbert envelopes provides robustness against noise and temporal smearing caused by reverberation (Ganapathy et al., 2014).

As one of our key contributions, we propose a novel modification of the 2DAR processing by replacing conventional *linear prediction* (LP), conducted after FDLP, with *time-varying linear prediction (TVLP)* (Vestman et al., 2017). In TVLP, the coefficients of the linear predictive filter are not considered to be stationary but they are time-varying (*i.e.* non-stationary) and expressed using basis functions (such as polynomials or trigonometric functions). The type and number of the basis functions can be tuned to control the rate of change of the underlying vocal tract model. As a result of adopting TVLP, linear prediction filter coefficients follow slowly-varying time-continuous contours, modelled by the basis functions. Therefore, the corresponding features are less prone to change abruptly over time, which is a phenomenon that degrades, for example, conventional LP-based features when speech is of low-intensity (as in whispers) or corrupted by noise. To be able to apply TVLP after FDLP, we modify the original TVLP model (Hall et al., 1983;

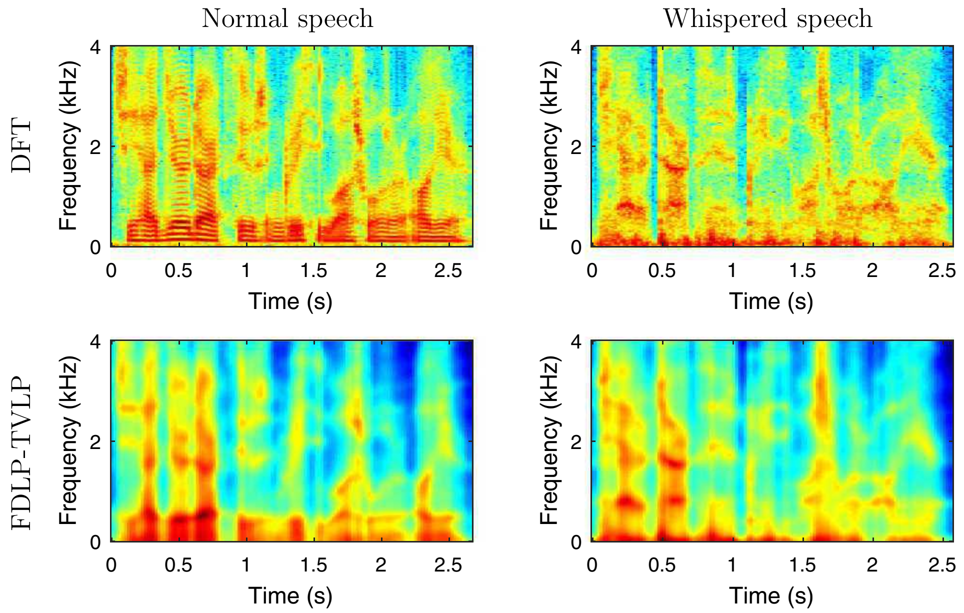


Fig. 1. Spectrograms computed using *discrete Fourier transform* (DFT) and the proposed *frequency-domain linear prediction with time-varying linear prediction* (FDLP-TVLP), for the sentence *"She had your dark suit in greasy wash water all year."* uttered by a male speaker in normal and whispered speaking styles. Due to the mismatch in speaking style, the conventional DFT spectrograms between normal and whispered speech differ substantially from each other, while the proposed spectrogram exhibits relatively less variation.

Rudoy et al., 2011), which assumes raw waveform as an input, to be applicable to spectro-temporal representations produced by FDLP.

Next, in Section 2, we present an extensive literature review on automatic speaker recognition from whispered speech and describe available corpora of whispered speech. In Section 3, we discuss the properties of whispered speech and propose a method for automatically comparing formants of normal and whispered speech for a large speech corpus. The proposed method aligns normally spoken sentences with their whispered counterparts by matching the speech content. The method does not require speech transcription with time alignments, so it can be readily applied to any corpus containing parallel data of two speaking styles. In Sections 4, 5, and 6.1, we extend our preliminary study (Vestman et al., 2017) in many ways both regarding the methodology and experiments. Firstly, we provide a more thorough and self-contained exposition of the used methodologies. Secondly, unlike in (Vestman et al., 2017), we study the choice of the basis functions in TVLP. Thirdly, we verify our early positive findings with a new dataset and a different problem domain ((Vestman et al., 2017) focuses on reverberation robustness rather than speaking style mismatch). Finally, for the comparison, we include a broader set of state-of-the-art reference features including *power-law adjusted linear prediction* (LP-α) (Saeidi et al., 2016) and *minimum variance distortionless response* (MVDR) (Murthi and Rao, 2000) features.

2. Speaker recognition from whispered speech

In Table 1, we have summarized the main characteristics of the speaker recognition studies on whispered speech that we found. We hope that this table gives the interested reader a better understanding of the studies on the topic. For those conducting their own research on whispered speech, we also recommend (Lee et al., 2014) to get more insight of available speech corpora containing whispered speech.

The performance numbers in the table indicate that the task at hand is a rather difficult one. In the identification task, the accuracy can be as low as 50 percent when the speaking style mismatch induced by whispering is present, while the accuracy in the non-mismatched case is near 100 percent. Similar drop in performance is naturally present in the speaker verification studies where results are reported as equal error rates (EERs).

We found that the comparison of previous studies and their methods is difficult as they have differences in corpora, recognition task (ID/verification), evaluation metric, and in the general approach to address the problem. Some studies focus on reducing the speaking style mismatch by developing features capturing properties that are present in the two speaking styles (*i.e.* enrollment vs. testing), while other studies develop complementary features to be fused in order to tackle the mismatch. Another approach is to transform features of normal speech to resemble those in whispered speech (Fan and Hansen, 2013), and to use the transformed features in the enrollment phase (Fan and Hansen, 2013; Sarria-Paja et al., 2016).

As is apparent from the spectrograms in Fig. 1, there are clear differences between normal and whispered speech. The differences are especially evident at low frequencies ($<$ 1.5 kHz) due to the lack of the periodic voiced excitation in whisper (Ito et al., 2005). This property of whispered speech has inspired many previous feature extraction technologies. Some of the features have been extracted from a *limited bandwidth* that exclude lower frequencies altogether. Also, because lower frequencies do not have as important role for whispered speech as for normal speech, features with different frequency warping strategies, such as *linear frequency cepstral coefficients* (LFCCs) and *exponential frequency cepstral coefficients* (EFCCs), have been adopted (Fan and Hansen, 2011b).

Being the default feature extraction scheme in speaker recognition, *mel-frequency cepstral coefficients* (MFCCs) (Davis and Mermelstein, 1980) have been included in every study listed in Table 1. Therefore, to avoid repetition, we have excluded MFCCs from the listed methods. Some of the features studied were not proposed originally for whispered speech, but as they have shown good performances in other tasks, they have later been adopted to whispered speech in many experiments. These features include *weighted instantaneous frequencies* (WIFs) (Sarria-Paja et al., 2013),

Table 1
Overview of previous studies on automatic whispered speech speaker identification or verification. Performances are given for cases: 1) speakers are enrolled and tested with normal speech (n-n) and 2) speakers are enrolled with normal speech but tested using whispered speech (n-w). Even though all the numbers are not exact, they can be used to get a general idea of how much n-w mismatch affects the performance. The number of speakers (#Speakers) tells how many speakers were enrolled in the recognition experiments.

Ref.	Year	Corpus	#Speakers (f/m)	Methods	Backend	Performance (n-n / n-w)	Metric
(Jin et al., 2007)	2007	Described in (Jin et al., 2007)	22 (n/a)	Feature warping, GMM score combination	GMM	- / -	-
(Grimaldi and Cummins, 2008)	2008	CHAINS (Cummins et al., 2006)	36 (16/20)	Pykfec features	GMM	80 – 95% / 15 – 40%	Acc.
(Fan and Hansen, a)	2008	UT-VE I (Zhang and Hansen, 2009)	10 (0/10)	Frequency warping, GMM score competition, LP power spectrum, unvoiced consonant detection	GMM	92% / 53–80%	Acc.
(Fan and Hansen, 2009)	2009	UT-VE I	10 (0/10)	Limited band LFCCs, feature mapping, LP power spectrum, unvoiced consonant detection	GMM	94% / 48 – 68%	Acc.
(Fan and Hansen, b)	2009	UT-VE I	10 (0/10)	Modified temporal patterns	GMM	- / 44.1–70.4%	Acc.
(Fan and Hansen, 2011b)	2011	UT-VE II (Zhang and Hansen, 2009)	28 (28/0)	LFCC, EFCC, unvoiced consonant detection	GMM	99.2% / 79.3 – 88.4%	Acc.
(Fan and Hansen, 2011a; 2013)	2011, 2013	UT-VE I & II	28 (28/0)	Feature transformation from neutral to whisper ((Fan and Hansen, 2013) extends (Fan and Hansen, 2010))	GMM-UBM	99.1% / 79.3% – 88.9%	Acc.
(Jawarkar et al., 2013)	2013	Described in (Jawarkar et al., 2013)	25 (n/a)	TESBCC, TTESBCC, WIF	GMM	99% / 56%	Acc.
(Sarria-Paja et al., 2013)	2013	CHAINS	36 (16/20)	WIF	GMM-UBM	- / -	-
(Sarria-Paja and Falk, 2015)	2015	CHAINS	36 (16/20)	frequency & feature warping, LFCC, WIF, limited band features	GMM-UBM	about 2% / about 30%	EER
(Sarria-Paja et al., 2015)	2015	CHAINS, wTIMIT (Lim, 2011)	60 (n/a)	WIF	GMM-UBM, i-vector / PLDA	1.6 – 4.4% / 25.8 – 29.2%	EER
(Sarria-Paja et al., 2016)	2016	CHAINS, wTIMIT	60 (n/a)	WIF, feature mapping and fusion	i-vector / PLDA	2.9% / 28.0%	EER
(Sarria-Paja and Falk, 2017)	2017	CHAINS, wTIMIT	60 (n/a)	AAMF, Residual MFCC, limited band features, fusion	i-vector / PLDA	0.9% / 17.8 – 27.3%	EER
(Sharma et al., 2017)	2017	CHAINS	36 (16/20)	EMD-based feature	GMM	8.75 – 9.18% / 13.8 – 14.81%	EER

pyknogram frequency estimate coefficients (pykfecs) (Grimaldi and Cummins, 2008) and *temporal energy subband cepstral coefficients* (TESBCCs, TTESBCCs) (Jawarkar et al., 2013).

Some of the methods presented in Table 1 harvest long-term properties of speech to the features but this is done in different ways. In (Fan and Hansen, 2009), subband specific features are extracted with a technique called *modified temporal patterns* (m-TRAPs). More precisely, features are extracted from the horizontal strides of a spectrogram obtained by filtering the spectra with 13 linear filters. In contrast, in the *auditory-inspired amplitude modulation feature* (AAMF) extraction scheme (Sarria-Paja and Falk, 2017), features are extracted from blocks of spectrograms consisting of multiple consecutive short-time frames. AAMFs characterize the rate of change in long-term subband envelopes. As this leads to high-dimensional feature representations, feature selection and principal component analysis (PCA) have been adopted. Feature selection is also used to select features that share the highest amount of mutual information across different speaking styles. The third previous method adopting contextual information extracts features known as the *mean Hilbert envelope coefficients* (MHECs) (Sadjadi and Hansen, 2015). These features are closest to the 2DAR based features studied in the present investigation as the MHEC extraction includes smoothing of subband Hilbert envelopes and, similarly to 2DAR, it finally outputs features that resemble standard short-term features.

Recently, *empirical mode decomposition* (EMD) based features have been investigated to extract complementary speech information (Sharma et al., 2017). Features are extracted from *intrinsic mode functions* (IMFs) and they have shown to boost whispered speaker recognition performance when combined with MFCCs.

3. Properties of whispered speech

The lack of voiced excitation (i.e. periodic glottal flow) in whispered speech (Tartter, 1989) is the main aspect that makes whispered speech different from normal speech. The lack of voicing (and thereby also the lack of fundamental frequency and its harmonics) in whispered speech results in reduction of sound energy at low frequencies, which in turn increases spectral flatness. The lack of voicing together with low intensity of the sound makes whispered speech less intelligible than normal speech. Therefore, speakers tend to adapt their voice production mechanisms in other ways to enhance speech intelligibility. These adaptations can be carried out, for example, by changing the vocal tract configuration (affecting formant center frequencies and their bandwidths), speaking rate, or phone durations.

In addition to the lack of voiced excitation, a number of other acoustic differences between normal and whispered speech have been reported. For instance, frequencies of the lowest three formants (F1–F3) tend to be higher in whispered speech (Ito et al., 2005; Heeren, 2015; Higashikawa et al., 1996) with the largest increase in F1. In (Ito et al., 2005), two other observations were made. First, whispered speech sounds were found to have less energy in frequencies below 1.5 kHz. Second, and rather expectedly, by comparing the average cepstra of individual phone segments, it was shown in (Ito et al., 2005) that the cepstral distance between voiced utterances of normal speaking style and the corresponding sounds in whispers is greater than the distance between unvoiced sounds in normal speaking style and the corresponding phones in whispers.

In this study, we analyze first how formant (center) frequencies and formant bandwidths differ between whispered and normal speech. Differently from (Ito et al., 2005; Heeren, 2015; Higashikawa et al., 1996) where formants were analyzed either from recordings of isolated vowels or from automatically segmented speech sounds relying on manually segmented training data, we automatically align whispered sentences to their normally spoken counterparts without requiring any speech transcription or manual annotation of segments. After the alignment, the aligned frames are compared to measure differences in formants between whispered and normal speech. Our method can be used not only for isolated vowel utterances but also in processing of realistic, continuous speech, and neither requires it performing manual or automatic speech sound segmentation.

3.1. Corpus description

To analyze formants via aligning normal and whispered speech, a parallel corpus containing utterances spoken in both speaking styles is needed. To this end, we adopt the CHAINS (*CHAracterizing INdividual Speakers*) corpus (Cummins et al., 2006), used especially in recent speaker recognition studies involving whispered speech. The CHAINS corpus is, importantly, also publicly available.[1] The corpus is targeted for advancing the study of speaker identification by investigating unique characteristics of speakers and it contains recordings from 16 females and 20 males speaking in various styles, including normal and whispered speech. Majority of the speakers share the same dialect, spoken in the Eastern part of Ireland. For the formant analysis we included 12 speakers from both genders from the described dialect region. We utilized 33 utterances available in the corpus for each speaker for normal and whispered speaking styles. All the normally spoken samples originate from a single recording session and all the whispered recordings from another session. These sessions were held about two months apart. Speech is sampled at 44.1 kHz, and this sample rate is also used in the formant analysis.

3.2. Analysis of changes in formants via speech alignment

We used VoiceSauce (Shue et al., 2011) with Praat back-end (Boersma, 2017) (Burg's algorithm) to extract formant (center) frequencies and the corresponding formant bandwidths for the lowest three formants for all the utterances. Formants were extracted using 20 ms frame every 2 ms. To make formant tracks less noisy, both formant frequency and bandwidth tracks were median filtered using a 9-frame window.

After extracting the formant data, we considered all pairs of whispered and normal speech where the same speaker spoke the same sentence. Since we have 33 sentence pairs from 12 speakers for both genders (except for one file missing from the original corpus), the total number of sentence pairs is $33 \cdot 12 \cdot 2 - 1 = 791$. We aligned pairs of whispered and normal sentences by using *dynamic time warping* (DTW) (Ellis, 2003). It is apparent from Fig. 2 that aligned sentences contain parts where either the alignment can be imprecise or the formant tracks are not reliably estimated. Therefore, we use an automatic detection of reliable segments containing no alignment or formant tracking errors. For details of DTW and the automatic detection of reliable segments, see Appendix A. In panels 2 and 3 of the figure, the segments that are detected to be well aligned are marked with yellow bars. These segments provide aligned formant frequency and bandwidth pairs to be used in analyzing differences in formants between normal and whispered speech.

3.3. Analysis results

We pooled aligned frame pairs and the corresponding aligned formant frequencies and their bandwidths over all speakers and sentences for both genders. Then, we computed histograms of the center frequencies (F1-F3) of the lowest three formants and their bandwidths (B1-B3) for both speaking styles and genders using a 20 Hz bin size. The histograms of the formant frequencies and bandwidths are depicted in Figs. 3 and 4, respectively. Further, Table 2 summarizes the mean statistics.

Distributions in Fig. 3 show that formant frequencies tend to be higher in whispered speech. Differences between normal and whispered speech are more prominent for F1 and less so for F2 and F3. On average, for both genders, F1 is about 150 Hz higher and F2 and F3 about 100 Hz higher in whispered speech. An exception is F3 of male speakers, where there is little difference between the two speaking styles. By in large, these observations are in line with the earlier results obtained using different analysis methods (Ito et al., 2005; Heeren, 2015; Higashikawa et al., 1996) for other corpora.

The analysis of formant bandwidths shows that whispered speech tends to have higher B1 whereas B3 tends be higher in normal speech. Bandwidth B2 is similar in both speaking styles.

4. Features for whispered speech speaker recognition

4.1. Two-dimensional autoregressive features

Two-dimensional autoregressive speech modeling (2DAR), introduced in (Athineos et al., 2004), provides a way to construct speech spectrograms that are smoothed both in time and frequency dimensions. As a result of such smoothing, 2DAR spectrograms can be used to extract features that are robust against noise and reverberation without losing relevant information that is used for recognition purposes (Ganapathy et al., 2014). The smoothing is first applied in the temporal domain using *frequency domain linear prediction* (FDLP) (Herre and Johnston, 1996; Kumaresan and Rao, 1999), after which spectral smoothing is done using *time domain linear prediction* (TDLP) (Makhoul, 1975). In the following, a brief description of both of these techniques is provided by starting from TDLP, more commonly known as LP (*linear prediction*).

4.1.1. Linear prediction

In conventional LP analysis (Makhoul, 1975), the current speech sample $x[n]$ is predicted as a weighted sum of the past p samples given by

$$\hat{x}[n] = -\sum_{k=1}^{p} a_k x[n-k] \tag{1}$$

where a_k, $k = 1, \ldots, p$, are known as the *predictor coefficients*. The most common way of solving the predictor coefficients is to minimize the prediction error in the least squares sense. That is, we minimize

$$E = \sum_n e^2[n] \tag{2}$$

where $e[n] = x[n] - \hat{x}[n]$. The minimum of (2) is found by calculating partial derivatives with respect to all predictor coefficients a_k and equating them to zero. As a result, we obtain a set of linear equations

$$\sum_{k=1}^{p} a_k r_{ki} = -r_{0i}, \qquad i = 1, \ldots, p, \tag{3}$$

where r_{ki} denotes the *correlation coefficients* given by

$$r_{ki} = \sum_n x[n-k]x[n-i].$$

In 2DAR, the *autocorrelation method* (Makhoul, 1975) is used to solve the predictor coefficients from (3). After solving the coefficients, an *all-pole* estimate of the magnitude spectrum can be obtained as a frequency response of the filter

$$H(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}},$$

where G is the gain coefficient.

4.1.2. Frequency domain linear prediction

The first part of 2DAR-modeling, FDLP (Herre and Johnston, 1996; Kumaresan and Rao, 1999), can be regarded as the frequency domain

[1] http://chains.ucd.ie/ (URL accessed 2018/03/09 09:44:44).

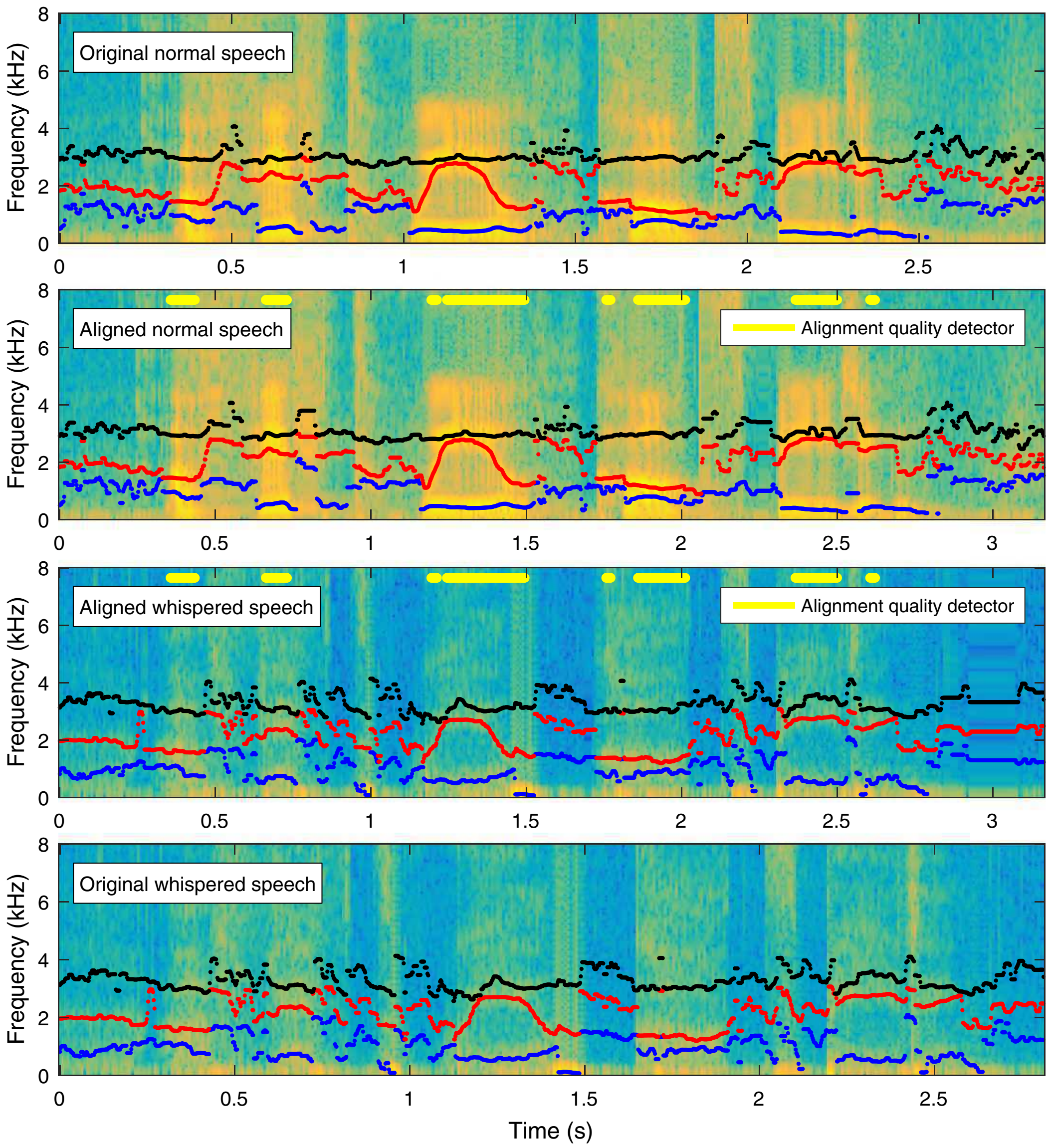


Fig. 2. Alignment of normal and whispered speech using dynamic time warping. The first and the last panel of the figure display the spectrograms and formant tracks of the original (non-aligned) pair of normal and whispered sentences spoken by the same speaker. The second and third panel show these sentences after DTW alignment. The aligned sentences are of the same duration, which is longer than the durations of the original sentences because DTW has repeated certain frames multiple times in the aligned speech. After the alignment, an automatic alignment quality detection is applied to the aligned sentences to discard sections of speech where the alignment is unreliable due to, for example, noisy formant tracks or low energy content. We retain the aligned and detected high-quality segments for subsequent analyses.

counterpart of LP. It was shown in (Athineos and Ellis, 2007) that by applying LP to a signal computed by discrete cosine transform (DCT) provides an all-pole estimate of the squared Hilbert envelope of the original time domain signal. In 2DAR, however, FDLP is not applied to the full-band speech signal, but instead to individual frequency bands obtained by windowing the DCT-transformed signal. As a result, all-pole models of the Hilbert envelopes are obtained for each frequency band, and these all-pole models can be used to approximate frequency band energies at regular time instants (*e.g.* once in 10 ms).

4.1.3. *Two-dimensional autoregressive modeling*

In classical speech analysis, LP is applied by predicting time domain samples within short frames of speech. The 2DAR model, in contrast, models longer-term properties of speech. It achieves this by first reversing the time and frequency domains. This leads to obtaining temporal all-pole power estimates for long-term subbands instead of all-pole spectrum estimates for short-time frames.

The processing steps of 2DAR are depicted in Fig. 5. The first step in 2DAR is to transform speech into the frequency domain with DCT. Then, the DCT signal is windowed into subbands using rectangular

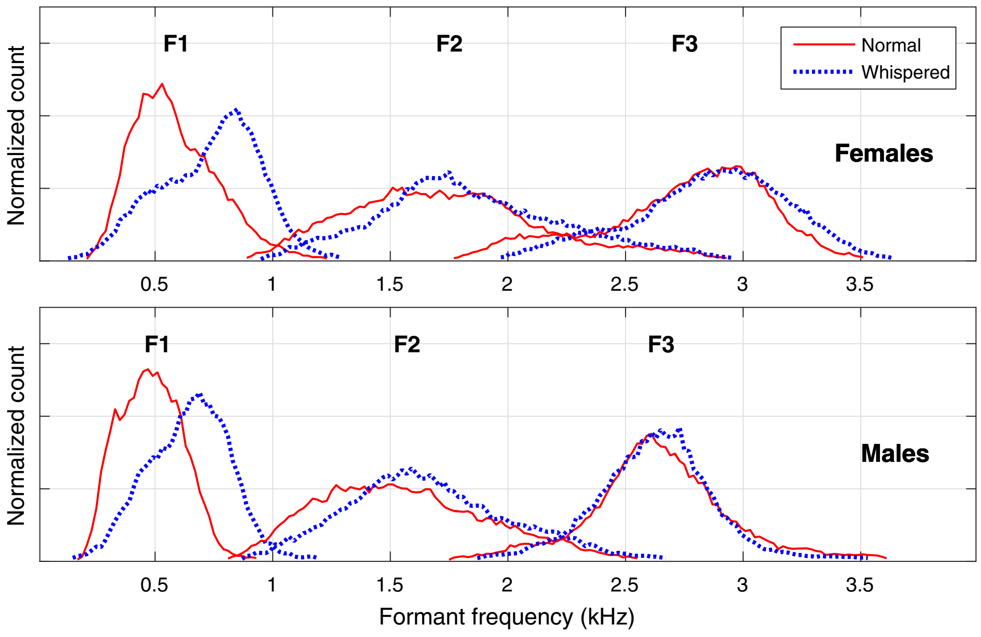


Fig. 3. Formant frequency histograms of the lowest three formants (F1–F3) estimated from normal and whispered speech of female and male talkers. The histograms are computed using a 20-Hz bin size. F1 shows the largest change between the two speaking styles.

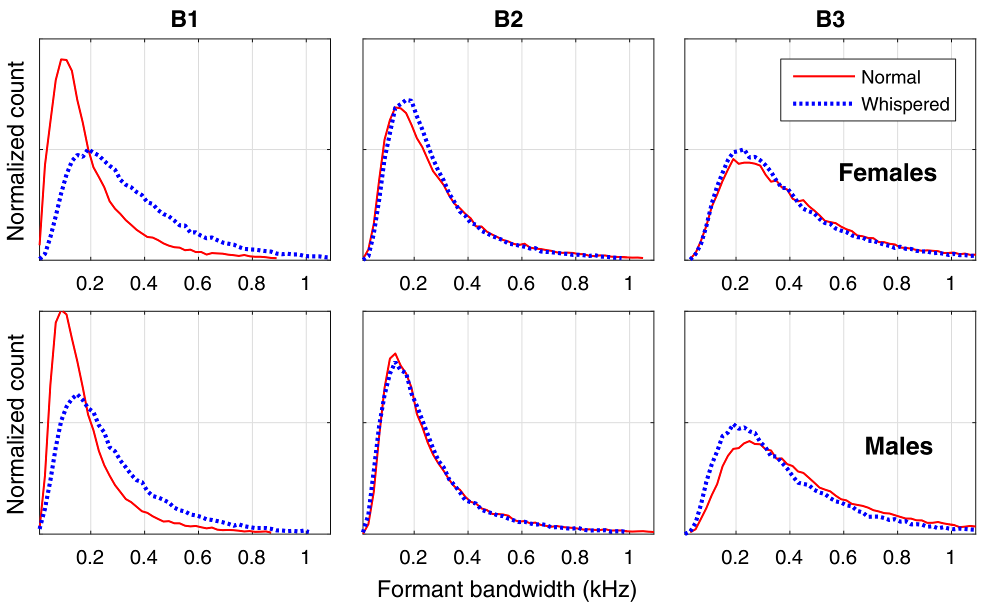


Fig. 4. Formant bandwidth histograms of the lowest three formants (B1–B3) estimated from normal and whispered speech of female and male talkers. B1 shows the largest change between the two speaking styles.

windows. In this study, we use 100 bands with an overlap of 60% between adjacent bands. These bands are then subjected to the FDLP modeling (LP in frequency domain) to obtain models of the Hilbert envelopes in each band. These envelopes are, in turn, windowed using 25 ms Hamming windows with 60% overlap. Samples of each windowed envelope are integrated to obtain power estimates for each 25 ms time interval of the corresponding frequency band. By stacking power estimates over different subbands, we obtain power spectral estimates for all time-frames. As the next step, the power spectral estimates are inverse Fourier-transformed to compute the autocorrelation

Table 2
Mean formant frequencies (F1–F3) and bandwidths (B1–B3) in Hz. The standard error of the mean for all values is about 1 Hz.

		F1	F2	F3	B1	B2	B3
Females	Whispered	746	1853	2861	353	277	396
	Normal	595	1743	2753	206	290	436
	Difference	151	110	108	147	−13	−43
Males	Whispered	665	1675	2642	286	259	382
	Normal	509	1572	2652	195	276	480
	Difference	156	103	−10	91	−17	−98

function. The autocorrelation functions are used in the TDLP, which outputs the final spectral estimates that are used in 2DAR spectrograms and in feature extraction.

4.2. *Two-dimensional time-varying autoregressive features*

In our preliminary work (Vestman et al., 2017), we have proposed a modification to the 2DAR method that uses *time-varying* linear prediction in the place of conventional LP. In the present study, we cover this technique in a more elaborate manner in both theoretical and experimental means.

4.2.1. *Classical time-varying linear prediction*

The conventional LP analysis (Makhoul, 1975) assumes the underlying vocal tract model of a speech signal to remain constant over each short-time interval (frame) of speech. Depending on the frame increment, the model can have abrupt changes from frame to frame. In reality, however, the vocal tract is a continuously varying system that changes even within a single 25 ms frame. TVLP model (Hall et al., 1983; Rudoy et al., 2011) takes into account the non-stationarity of the vocal tract by allowing the predictor coefficients a_k to be time-varying. Thus, in the case of TVLP (1) becomes

$$\hat{x}[n] = -\sum_{k=1}^{p} a_k[n]x[n-k]. \tag{4}$$

By itself, (4) does not prevent the occurrence of models that change rapidly in time because no constraint has yet been imposed on the change of the predictor coefficients. In TVLP, the rate of the change is constrained by representing the time trajectories of the predictor coefficients as a linear combination of $q+1$ basis functions $\{u_i[n]\}_{i=0}^{q}$ as follows:

$$a_k[n] = \sum_{i=0}^{q} b_{ki}u_i[n]. \tag{5}$$

Typically, basis functions are selected so that they provide smooth, low-pass type of predictor coefficient trajectories. A high number of such basis functions allows for more rapid changes in the predictor coefficients and in the vocal tract model. Conversely, using only one constant basis function, $u_0[n] = 1$, makes the model equivalent to LP. An example of using simple monomial basis function in TVLP modeling of a 50 ms speech frame is given in Fig. 6.

In TVLP, minimization of (2) with respect to each basis coefficient b_{ki} leads to a set of equations given by

$$\sum_{k=1}^{p}\sum_{i=0}^{q} b_{ki}c_{ij}[k,l] = -c_{0j}[0,l] \tag{6}$$

for $1 \le l \le p$ and $0 \le j \le q$ (Hall et al., 1983). Here $c_{ij}[k, l]$ denotes the *generalized* correlation coefficients defined as

$$c_{ij}[k,l] = \sum_{n} u_i[n]u_j[n]x[n-k]x[n-l]. \tag{7}$$

4.2.2. *Proposed autocorrelation domain time-varying linear prediction*

In classical TVLP formulation, as can be seen from Eq. (7), the computation is directly based on the time-domain signal. However, this

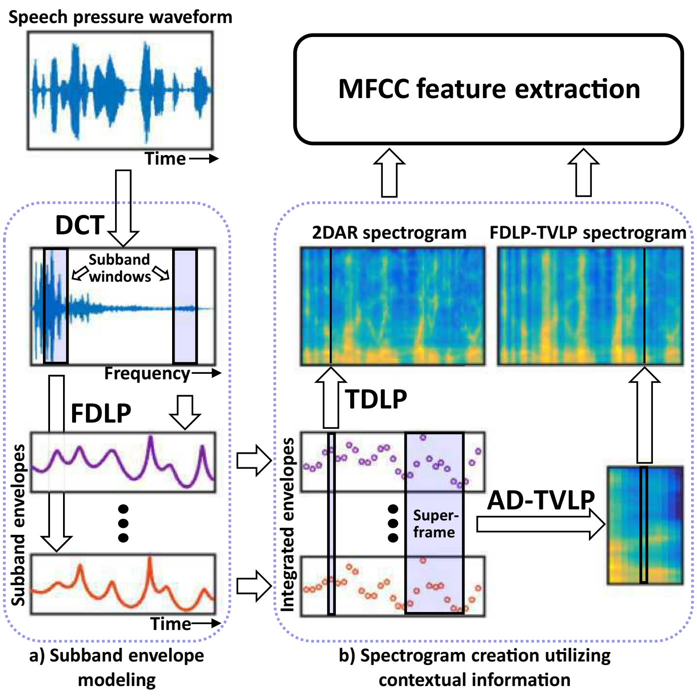


Fig. 5. Process flows in creating 2DAR and FDLP-TVLP spectrograms from sentence “If it doesn’t matter who wins, why do we keep score?”. FDLP applied to subband windows provides time domain envelopes for the subbands. The integrated envelopes shown in the bottom are obtained by summing up values of subband envelopes over 25 ms long Hamming windows that have a 15 ms overlap. Integrated envelopes provide power spectral estimates that are transformed to autocorrelation values and used either in TDLP or autocorrelation domain TVLP (AD-TVLP) modeling. TDLP processing is performed for individual frames, while in AD-TVLP, modeling is performed in superframes that consist of multiple consecutive short-time frames. As the superframe slides forward one frame at a time, only the center spectrum resulting from AD-TVLP modeling of the superframe is retained to the FDLP-TVLP spectrogram.

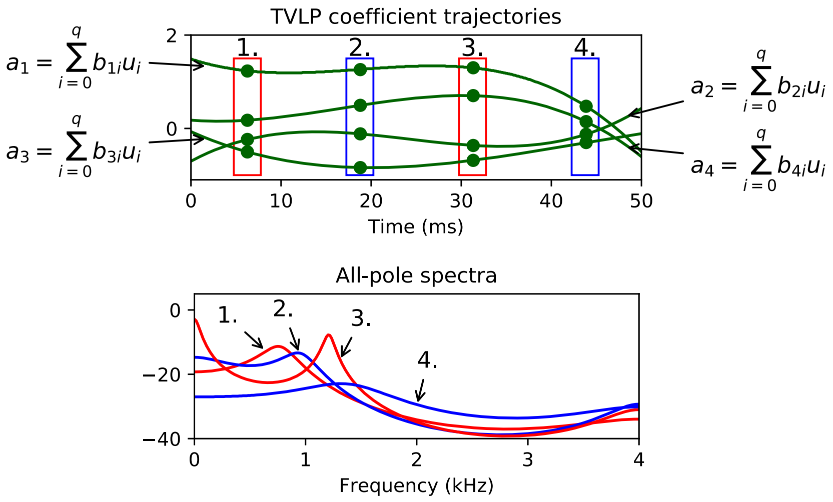


Fig. 6. An example of time-varying linear predictive (TVLP) modeling. In this illustration, TVLP coefficient trajectories of a 50 ms long speech frame are modeled using four ($q = 3$) basis functions u_i, $i = 0, \ldots, q$. The model order of TVLP is set to $p = 4$ resulting in four trajectories shown in the upper graph. Because the coefficients are time-varying within the frame, an unique set of predictor coefficients can be sampled at any time instant. The lower graph shows examples of all-pole spectra obtained from four different time instants.

can be a serious constraint if one wants to combine the advantages of TVLP with some other robust signal processing techniques. In several robust feature extraction techniques, such as FDLP, *non-negative matrix factorization* (NMF) based feature enhancement (Weninger et al., 2011), and missing data imputation, the signal is converted into a spectro-temporal representation before processing or enhancing it further. In such a scenario, the use of classical TVLP requires the spectro-temporal representations to be converted back to time domain samples. This in turn would require a careful handling of the phase information. In order to avoid distortions that may occur due to phase reconstruction, we propose a modified TVLP analysis which can performed *directly* on the spectro-temporal representations.

Any given spectro-temporal representation $X(t, f)$ can be converted into a sequence of autocorrelation functions by computing the inverse Fourier transform of the power spectrum $X(t, f)$ at each time instant t, given by

$$r_\tau(t) = \frac{1}{2\pi}\int_f X(t,f)\exp(j2\pi f\tau)df \tag{8}$$

At this stage, one can use the conventional LP or TDLP independently on the correlation functions at each time instant by solving the normal equations similar to that in Eq. (3). In such a scenario, the LP coefficients derived at each time instant are prone to errors induced by non-stationary noise and do not take advantage of the fact that the speech production apparatus is a slowly varying inertial system. In order to take advantage of this inertial property of the speech production system, we propose a new TVLP formulation that operates directly on the autocorrelation sequences. This is achieved by imposing a time-continuity constraint on the LP coefficients derived at each time instant by modifying the normal equations in Eq. (3). The resulting normal equations with the continuity constraint, making use of the sequence of autocorrelation functions, is given by

$$\sum_{k=1}^{p} a_k[n]r_{ki}[n] = -r_{0i}[n], \qquad i = 1,\ldots,p, \qquad n = 0,\ldots,N-1, \tag{9}$$

where $r_{ki}[n]$, $a_k[n]$, and N denote the time-varying autocorrelation coefficients, the time-varying LP coefficients, and the window length for the time-varying analysis, respectively. This expression is similar to Eq. (3), except that both the filter coefficients as well as the correlation coefficients are now functions of time.

Now approximating the piecewise constant filter coefficients $a_k[n]$ using Eq. (5), the above expression in Eq. (9) can be written as

$$\sum_{k=1}^{p}\sum_{j=0}^{q} b_{kj}u_j[n]r_{ki}[n] = -r_{0i}[n], \qquad i = 1,\ldots,p, \qquad n = 0,\ldots,N-1, \tag{10}$$

This expression is similar to Eq. (6), except that the autocorrelation coefficients $r_{ki}[n]$ do not include basis functions $u_j[n]$ in their computation, as is the case with $c_{ij}[k, l]$ in Eq. (7). This expression can also be interpreted as modeling a piecewise constant filter coefficients $\{a_k[n];\ k = 1\cdots p,\ n = 0\cdots N-1\}$ using a smooth continuous time-varying model represented by $\{b_{ki};\ k = 1\cdots p,\ i = 0\cdots q\}$.

The above set of linear equations can be written in matrix form, given by

$$\boldsymbol{Rb} = -\boldsymbol{r} \tag{11}$$

where

$$\boldsymbol{r} = [r_{01}[0],\cdots,r_{0p}[0],\cdots,r_{01}[N-1],\cdots,r_{0p}[N-1]]^T_{Np\times 1} \tag{12}$$

$$\boldsymbol{b} = [b_{10},\cdots,b_{1q},\cdots,b_{p0},\cdots,b_{pq}]^T_{p(q+1)\times 1} \tag{13}$$

$$\boldsymbol{R} = [R_0,\ R_1,\cdots,R_{N-1}]^T_{Np\times p(q+1)}. \tag{14}$$

Here R_n is a $p(q+1)\times p$ matrix whose ith column is given by

$$R_{ni} = \boldsymbol{r}_i[n] \otimes \boldsymbol{u}[n], \tag{15}$$

and $\otimes$ denotes the Kronecker product of $\boldsymbol{r}_i[n] = [r_{1i}[n],\cdots,r_{pi}[n]]^T$ and $\boldsymbol{u}[n] = [u_0[n],\cdots,u_q[n]]^T$, given by

$$R_{ni} = [r_{1i}[n]u_0[n],\cdots,r_{1i}[n]u_q[n],\cdots,r_{pi}[n]u_0[n],\cdots,r_{pi}[n]u_q[n]]^T_{p(q+1)\times 1}. \tag{16}$$

The least square solution to the set of linear equations in Eq. (11) can be computed as

$$\hat{\boldsymbol{b}} = \underset{\boldsymbol{b}}{\arg\min}\, \|\boldsymbol{r} + \boldsymbol{R}\boldsymbol{b}\|_2^2. \tag{17}$$

The above TVLP formulation starting with a sequence of autocorrelation functions is refered to as *autocorrelation domain time-varying linear prediction* (AD-TVLP).

4.2.3. Proposed feature extraction method

Fig. 5 illustrates the difference between 2DAR and the proposed TVLP-enhanced version of 2DAR that we call as FDLP-TVLP. After FDLP processing, 2DAR models individual frames with LP, while in our method, we form "superframes" consisting of multiple consecutive frames and feed them to autocorrelation domain TVLP. While LP processing smooths the spectrogram only in the spectral dimension, TVLP is computed *simultaneously* in spectral and temporal domains making the final spectrogram more robust to noise.

The use of TVLP in the proposed AD-TVLP differs from previous TVLP studies (e.g. Rudoy et al., 2011; Hall et al., 1983) because speech is not modeled in sample-level precision but in frame-precision. While the conventional TVLP formulation uses only the sample-precision in time domain, AD-TVLP can be applied either for sample-precision subband envelopes or for frame-precision integrated envelopes. In our early experiments (Vestman et al., 2017), we found that operating on the frame-level provides better results in SID tasks. Therefore the current study focuses on the frame-level application of AD-TVLP.

4.3. Reference features

We compare 2DAR and and FDLP-TVLP features against multiple reference features. As a first reference feature, we use standard mel-frequency cepstral coefficients (MFCCs) (Davis and Mermelstein, 1980) with delta and double-delta coefficients appended. We provide spectral estimates for MFCCs with discrete Fourier transform (DFT). In this work, all the studied features differ only in the spectral estimation part. That is, for every feature, including 2DAR and FDLP-TVLP, we use the same MFCC feature extraction configuration except for the spectral estimation part.

Next, we evaluate features using conventional, short-term LP spectral estimation, which is a justified baseline for the long-term 2DAR and FDLP-TVLP features. We also include *power-law adjusted* LP (Saeidi et al., 2016) (LP-α) features, which showed positive results for shouted speech SID in a recent study (Saeidi et al., 2016). The LP-α is a spectral compression technique to reduce the effect of the spectral tilt difference between normal and shouted speech. Since the spectral tilt varies also between normal and whispered speech, it was justified to select also (LP-α) as one robust reference feature extraction method in the current study. Then, the FDLP method without TDLP or TVLP is included as a reference method containing temporal processing but without the spectral processing. Finally, as a last reference spectral estimation method, we use *minimum variance distortionless response* (MVDR) (Murthi and Rao, 2000) spectrum.

5. Experimental set-up for speaker recognition

5.1. Speech corpus and experimental protocols

To conduct speaker recognition experiments from whispered speech, a suitable evaluation corpus needs to be identified first. Unlike studying speech of normal speaking style, for which a large supply of corpora and associated standard evaluation protocols are available, there are fewer databases available for studying speaker recognition from whispered speech. With the help of data given in Table 1, we decided to adopt the CHAINS corpus (Cummins et al., 2006) in the current study. In the recognition experiments, the original speech data, sampled at 44.1 kHz, were downsampled to 16 kHz.

To cover a broad set of possible application scenarios, we designed two speaker recognition evaluation protocols. The first one, the speaker identification (SID) protocol, is relevant in applications such as personalized control of smart devices. The second one, the automatic speaker verification (ASV), is relevant in applications such as user authentication for access control, forensics, and surveillance. While there are several prior studies on speaker recognition from whispered speech (Table 1), there exists, unfortunately, no commonly used standard protocol. Hence, we decided to design our own protocols with an intention of maximizing the number of recognition trials with a limited amount of data.

5.1.1. Speaker identification protocol

The SID protocol utilizes recordings from 12 females and 12 males from the same dialect region (Eastern Ireland). For each speaker, we utilized 32 spoken sentences available in the corpus for normal and whispered speaking styles. The original corpus, in fact, contains 33 utterances, but we excluded one of them since one audio file was missing from the original corpus distribution.

Similar to (Saeidi et al., 2016), we adopt a *leave-one-out* protocol to increase the number SID evaluation trials: we leave one utterance at a time, to be used as the test trial, and use the remaining 31 utterances to train the target speaker model. As the average duration of an utterance is 2.81 s, the average duration of speech data to train the speaker model is about 87 s.

One SID trial consists of comparing the test utterance against all the 12 speaker models of the same gender. The identified speaker is the one whose target speaker model reaches the highest SID score. This way, we have in total $12 \times 32 = 384$ SID trials per gender. We conducted SID trials in two ways. First, by using normal speech for both speaker enrollment and testing and second, by using normal speech in enrollment but whispered speech in testing.

As our objective measure of performance, we compute speaker identification rate, defined as the proportion of correctly classified test segments to the total number of scored test segments, computed separately per each gender.

5.1.2. Speaker verification protocol

The ASV protocol utilizes all of the normal and whispered speech data in the CHAINS corpus in order to obtain more trials and to increase the reliability of the results. That is, we use all 33 sentences and 4 fables (typically 30 – 60 s long) from 36 speakers (16 females, 20 males). The first fable, with an average duration of 56 s, is used for training the target speaker models. The remaining 3 fables are cut into 3 s long clips and they are used together with the 33 sentences as test segments in the verification experiments. By testing all test segments against all speaker models, we obtain trial statistics summarized in Table 3. Again, trials were conducted in two ways by always enrolling speakers with normal speech, but testing either with normal or whispered speech. The designed protocol is similar to the one in (Sarria-Paja and Falk, 2015), with a difference that our test set is somewhat larger.

We report verification performances in terms of equal error rate (EER), the rate at which false alarm and miss rates are equal. When comparing the proposed features to the reference features, we also report 95% confidence intervals of EERs, computed using the methodology of Bengio and Mariéthoz (2004). That is, confidence interval around EER is EER $\pm\ c$, where

Table 3
Number of trials in the speaker verification protocol. The numbers of same-speaker trials are given in parentheses.

	Normal	Whispered
Females	16,752 (1,047)	17,136 (1,071)
Males	25,520 (1,276)	26,000 (1,300)
All	42,272 (2,323)	43,136 (2,371)

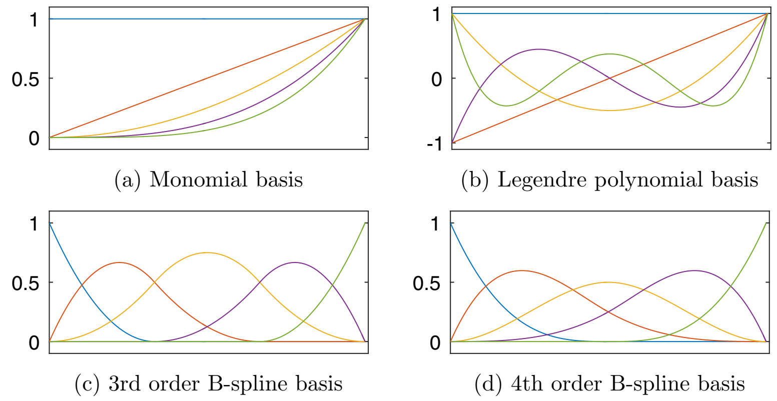


Fig. 7. Different types of basis functions used in AD-TVLP.

$$c = 1.96\sqrt{\frac{\mathrm{EER}(1-\mathrm{EER})}{4N_i} + \frac{\mathrm{EER}(1-\mathrm{EER})}{4N_s}},$$

where N_i is the number of impostor trials and N_s is the number of same speaker trials.

5.2. Speaker recognition system

We performed speaker recognition experiments with a classic *Gaussian mixture model – universal background model* (GMM-UBM) system (Reynolds et al., 2000). While there are many other possible choices for the back-end, including i-vectors, the GMM-UBM tends to provide comparative (or higher) accuracy for short utterances (Delgado et al., 2016; Zeinali et al., 2016; Dey et al., 2017) and is suitable with limited development datasets, requiring only UBM training data specification besides the enrollment and test samples. For each of the feature extraction techniques, we train a 256-component UBM using the TIMIT corpus, which is sampled at 16 kHz and recorded in quiet environments. To make the UBM training data gender-balanced, we use 192 speakers for both genders. The target speaker models are obtained by *maximum a posteriori* (MAP) adaptation (Lee and Gauvain, 1993) of the UBM using the training sentences of a particular speaker. A relevance factor of 2 was used to adapt the Gaussian component means of the UBM.

5.3. Feature configurations

The feature extraction techniques compared in this study differ substantially in their internal computations. At the output, however, they all yield estimates of the power spectrum (or power-spectrum like presentation) that are computed in 25 ms frames, incremented in 10-ms steps. For the power spectrum estimation, we study the following five reference methods besides the proposed FDLP-TVLP method: discrete Fourier transform (DFT), linear prediction (LP) (Makhoul, 1975), power-law adjusted LP (LP-α) (Saeidi et al., 2016), frequency domain linear prediction (FDLP) (Thomas et al., 2008), minimum variance distortionless response (MVDR) (Murthi and Rao, 2000), and 2-dimensional autoregressive model (2DAR) (Ganapathy et al., 2014). The power spectrum, estimated using one of these methods, is used as input to the MFCC computation chain in the standard way (Pohjalainen et al., 2014). In the identification experiments, the center frequency of the first and last mel-filter were set to 200 Hz and 7800 Hz, respectively, whereas in verification experiments, we adopt a narrower frequency range between 200 Hz and 5600 Hz (In Section 6.4, we study how the feature extraction bandwidth affects the system performance.) We use 19 MFCCs without the energy coefficient, appended with delta and double delta coefficients, yielding 57-dimensional feature vectors. MFCCs are RASTA-filtered (Hermansky and Morgan, 1994) except when temporal processing with FDLP is used, as it had a negative effect on the SID performance. Including both RASTA and FDLP could cause too much temporal smoothing of speech information. For the other spectrum estimation methods, RASTA had a positive or neutral effect. Finally, MFCCs of non-speech frames are discarded and the remaining MFCCs are normalized to have zero mean and unit variance per utterance.

Each of the feature extraction techniques have a number of control parameters that need to be set. For LP, we found the model order of at least 40 to yield the highest SID accuracy. Thus, in this study, we use $p = 40$ for LP, LP-α, and MVDR. We use the same model order for TDLP in 2DAR and for AD-TVLP in FDLP-TVLP. For both 2DAR and FDLP-TVLP, we found that a FDLP model order of 24 or higher for one second long segments provides the best performance for both normal and whispered speech. Because of the varying utterance lengths, we normalize the FDLP prediction order according to the length of the processed utterance. In the identification protocol, we use an FDLP model order of $p = 24$ and for the verification, we set the model order to $p = 48$. For LP-α, the best α value was found to be 0.05.

6. Speaker recognition results

In this section, we provide results of the conducted speaker recognition experiments. First, we optimize the control parameters of the proposed FDLP-TVLP feature extraction method and then continue by comparing the method to the reference methods. We provide results for two kinds of speaker recognition tasks, speaker identification (SID) and speaker verification. Further, we address the SID task in more detail by analyzing SID accuracies at the level of individual speakers.

6.1. The choice of basis functions for time-varying linear prediction

In TVLP, temporal contours of LP filter coefficients are modeled as a linear combination of basis functions. Many types of functions, such as *Monomial functions* (Liporace, 1975), *trigonometric functions* (Hall et al., 1983), and *Legendre polynomials* (Grenier, 1983), have been used previously. The choice of basis functions, however, has not been studied for the AD-TVLP formulation used in this study where we model the LP predictor coefficient trajectories at frame precision instead of sample-by-sample basis as is done in the classic TVLP. Therefore, we analyze the impact of the choice with four kinds of basis functions illustrated in Fig. 7.

First, we study the effect of superframe size (N) and the number of used basis functions ($q + 1$) together with a monomial basis (Fig. 7(a))

Table 4
Effect of the superframe size and the number of basis functions to the speaker verification performance (EER (%)) using monomial basis.

Superframe size (# frames (ms))	Normal vs. normal				Normal vs. whisper			
	Number of basis functions ($q + 1$)							
	3	4	5	6	3	4	5	6
7 (85 ms)	3.4 ± 0.4	3.4 ± 0.4	3.3 ± 0.4	3.5 ± 0.4	27.9 ± 0.9	28.3 ± 0.9	28.4 ± 0.9	28.9 ± 0.9
11 (125 ms)	3.7 ± 0.4	3.3 ± 0.4	**3.1** ± 0.4	3.5 ± 0.4	28.0 ± 0.9	**27.4** ± 0.9	28.1 ± 0.9	28.1 ± 0.9
15 (165 ms)	5.0 ± 0.5	4.5 ± 0.4	3.5 ± 0.4	3.5 ± 0.4	29.2 ± 0.9	29.0 ± 0.9	28.0 ± 0.9	28.7 ± 0.9
19 (205 ms)	6.0 ± 0.5	5.1 ± 0.5	4.2 ± 0.4	3.7 ± 0.4	30.4 ± 1.0	29.2 ± 0.9	28.8 ± 0.9	28.4 ± 0.9
23 (245 ms)	7.7 ± 0.6	6.5 ± 0.5	5.2 ± 0.5	4.6 ± 0.4	32.4 ± 1.0	30.5 ± 1.0	29.6 ± 0.9	28.8 ± 0.9

$$u_i[n] = n^i, \quad i = 0, ..., q, \quad n = 0, ..., N - 1, \tag{18}$$

used in our previous work (Vestman et al., 2017). The superframe size determines the number of adjacent frames being fed to the AD-TVLP model at once. The results presented in Table 4 indicate that the parameter choice is not critical, provided that the superframe size is sufficiently large and that the number of basis functions is large enough for a given superframe size. A suitable number of basis functions seems to be around one-third of the superframe size.

We fix the superframe size to $N = 11$ for the remaining experiments with other basis function types, namely Legendre polynomials (Fig. 7(b)) and B-splines (Fig. 7(c) and (d)) (Friedman et al., 2009). In contrast to monomials and Legendre polynomials, B-splines have a local support. However, the results in Table 5 indicate that this does not provide benefits to the given ASV task. Further, the monomial and Legendre bases provide equal results. Therefore, we consider only the monomial basis with 4 basis functions for all the remaining experiments.

6.2. Model orders for spectral and temporal processing of speech

As the 2DAR scheme performs linear prediction in both frequency (FDLP) and time (TDLP) domains, it has two main parameters to be optimized. The model order for FDLP determines the amount of smoothing in temporal subband envelopes; lower value resulting in more smoothed spectrograms in time. TDLP model order, in turn, is used to control the amount of details present in the frequency dimension. In (Ganapathy et al., 2014), the effect of model orders of 2DAR to speaker verification performance was studied for clean and noisy speech.

Similarly, the proposed FDLP-TVLP contains two model order parameters, one for FDLP and the other for AD-TVLP. In Fig. 8, we jointly vary both in the speaker verification task. Regarding to the FDLP model order, our results are very similar to the ones in (Ganapathy et al., 2014). In specific, the model order has to be at least 24 to obtain good results.

Concerning the model order in spectral processing, there are some differences largely explained by the differences in data. In (Ganapathy et al., 2014), the experiments performed with TDLP revealed that a high model order (> 20) is better for clean speech while for a noisy speech, a low model order (< 15) improves the performance. As our data is clean and has twice as high sampling rate (16 kHz), we find that for the AD-TVLP and for 16 kHz sampling rate, the best performance is obtained with model orders higher than 32.

6.3. Comparison of features in speaker identification task

Table 6 presents the identification results for all the evaluated spectrum estimation methods described in Sections 4 and 5.3. All the methods provide close to or equal to 100% identification rate when there is no speaking style mismatch present between enrollment and testing. With whispering-induced mismatch, however, all the accuracies drop to around 50%. Unlike in many other speaker recognition studies (*e.g.* Saeidi et al., 2016, Pohjalainen et al., 2014), we curiously find that female speakers obtain higher identification accuracies than males for whispered speech. From the whispered SID studies listed in Table 1, only (Sarria-Paja and Falk, 2015) reports SID results per gender basis and did not find considerable differences in performances between males and females on the same corpus and same type of back-end. The difference might be partly explained by differing sampling rate (16 vs. 8 kHz) and the evaluation protocol. In addition, unlike in the current study, gender dependent UBMs were used in (Sarria-Paja and Falk, 2015).

We have grouped the methods in Table 6 into three categories. The first group consists of the two standard short-term methods, DFT and LP, from which LP provides higher SID accuracy. Then, in the second group, the LP-α and MVDR methods have been introduced to provide added robustness to short-term features. From these two, only LP-α outperforms or matches the DFT and LP baselines when subjected to whispered speech SID. The last group consists of the FDLP-derived methods that use long-term speech processing. The FDLP method, by itself, is behind most of the short-term methods but improves substantially when combined with the spectral processing provided by LP (2DAR).

Finally, the proposed FDLP-TVLP method has a moderate margin to 2DAR and provides the best overall performance for whispered test cases. We also tried to include the α-compression of the power spectrum to the FDLP-TVLP method prior to the TVLP processing step, but as the

Table 5
Speaker verification equal error rates (%) for different basis types (superframe size = 11).

Basis type	Normal vs. normal				Normal vs. whisper			
	Number of basis functions							
	3	4	5	6	3	4	5	6
Monomial	3.7 ± 0.4	3.3 ± 0.4	**3.1** ± 0.4	3.5 ± 0.4	28.0 ± 0.9	**27.4** ± 0.9	28.1 ± 0.9	28.1 ± 0.9
Legendre	3.7 ± 0.4	3.3 ± 0.4	**3.1** ± 0.4	3.5 ± 0.4	28.0 ± 0.9	**27.4** ± 0.9	28.1 ± 0.9	28.1 ± 0.9
3rd order B-spline	8.3 ± 0.6	3.5 ± 0.4	3.4 ± 0.4	3.3 ± 0.4	32.6 ± 1.0	28.1 ± 0.9	27.8 ± 0.9	28.6 ± 0.9
4th order B-spline	–[a]	6.8 ± 0.5	3.3 ± 0.4	3.2 ± 0.4	–[a]	31.7 ± 1.0	28.2 ± 0.9	28.5 ± 0.9

[a] *Undefined configuration

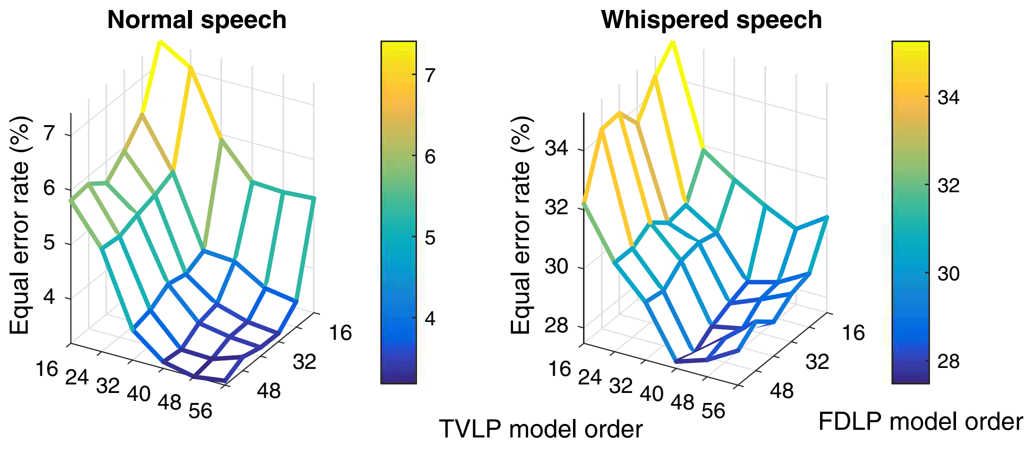


Fig. 8. Speaker verification equal error rates (%) for normal and whispered speech using different model orders in FDLP-TVLP modeling.

Table 6
SID accuracies (%) for different spectrum estimation methods.

Method	Normal vs. normal			Normal vs. whisper		
	Females	Males	All	Females	Males	All
DFT	100.0	100.0	100.0	52.1	40.4	46.2
LP	100.0	100.0	100.0	51.3	44.8	48.0
LP-α (α = 0.05)	100.0	100.0	100.0	53.9	42.4	48.2
MVDR	100.0	100.0	100.0	51.0	35.4	43.2
FDLP	96.1	94.8	95.4	39.3	32.6	35.9
2DAR	99.7	99.5	99.6	55.2	44.0	49.6
FDLP-TVLP	99.7	99.7	99.7	**56.5**	**45.3**	**50.9**
FDLP-TVLP-α	99.7	99.5	99.6	54.9	41.4	48.2

results show, we did not find this to be beneficial. This might be due to both methods already having similar beneficial effects by themselves, achieved through different means. An aggressive α-compression can be used to make spurious spectral peaks less prominent, but similar effect can be achieved using contextual information, as in FDLP-TVLP, by smoothing the spectra over time.

The obtained SID results, as a whole, imply more benefits being gained by improving spectral processing as opposed to temporal processing. This is supported by the good results obtained with LP and LP-α and by the large performance difference between FDLP and the other FDLP-based methods that include LP-based spectral processing. On the other hand, the proposed FDLP-TVLP achieved the best performance by including two layers of temporal processing, one by FDLP, and the other by AD-TVLP. This suggests that TVLP methodology, in the context of style mismatch compensation, is worthwhile of further studies.

6.4. The choice of frequency range in feature extraction

Next, we studied how the frequency range used in the MFCC extraction affects the identification results of DFT and FDLP-TVLP for whispered speech. We kept the first mel-filter centered at 200 Hz and changed the position of the other filters according to the position of the last mel-filter, which was varied between 4000 Hz and 7600 Hz. The results are presented in Fig. 9. We find that as the frequency range decreases the identification accuracy drops. Furthermore, we find that FDLP-TVLP does not seem to benefit from the inclusion of higher frequencies (> 5000 Hz) as much as DFT.

In prior studies (Sarria-Paja and Falk, 2015; Fan and Hansen, 2009), the frequency range has been limited by increasing the frequency of the first mel-filter. It has been found that discarding spectral information below 1 kHz improves system performance in normal-whispered mismatched test cases, since the spectral differences between the two speaking modes are largest in the low frequency range.

6.5. Speaker-by-speaker analysis

Up to this point, we have shown the results in a pooled form over all the speakers. With an aim to provide further insights into SID from whispered speech, we analyze results on a speaker-by-speaker basis. Fig. 10 displays the SID results of DFT and FDLP-TVLP methods for each speaker, sorted according to the number of correct identifications obtained using FDLP-TVLP. The results indicate large differences between individuals; some speakers are correctly identified almost every time, while others are almost always misidentified. From informal listening of the most difficult speakers, we could not identify any obvious abnormal speech characteristics or recording quality related issues. Hence, instead, we decided to analyze whether a change in formant values between normal and whispered speech explains the differences between SID performances of individual speakers. In Fig. 11, we present correlations between the number of correct identifications for a speaker and the average difference of formant frequencies between whispered and normal speech. We find a weak correlation between these two variables. Interestingly, the correlation is stronger for F2 than for F1 and it is also stronger for DFT than for FDLP-TVLP, which suggests that FDLP-TVLP might tolerate formant changes slightly better.

6.6. Comparison of features in speaker verification task

The results for the speaker verification task are presented in Table 7. As expected, the results resemble those obtained from the identification experiments. For normally spoken speech, male and female performances are close to each other. For whispered speech, however, there is a clear gap between genders; in specific, females show 3–6% lower EERs in absolute terms. As before, DFT and LP show the best performance in the normal speaking style, while for whispered speech, FDLP-TVLP gives the lowest error rates although it compromises performance in normal speech by about 0.5% (absolute EER). As a general finding, all the methods yield high error rates when tested under speaking style mismatch. Our results are similar to those reported in (Sarria-Paja and Falk, 2015).

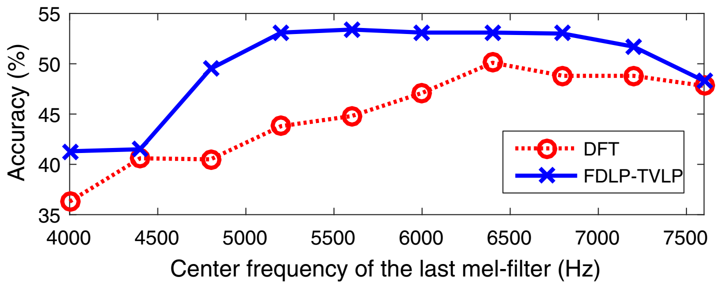


Fig. 9. Speaker identification accuracies (%) for whispered speech using different frequency ranges in the MFCC computation.

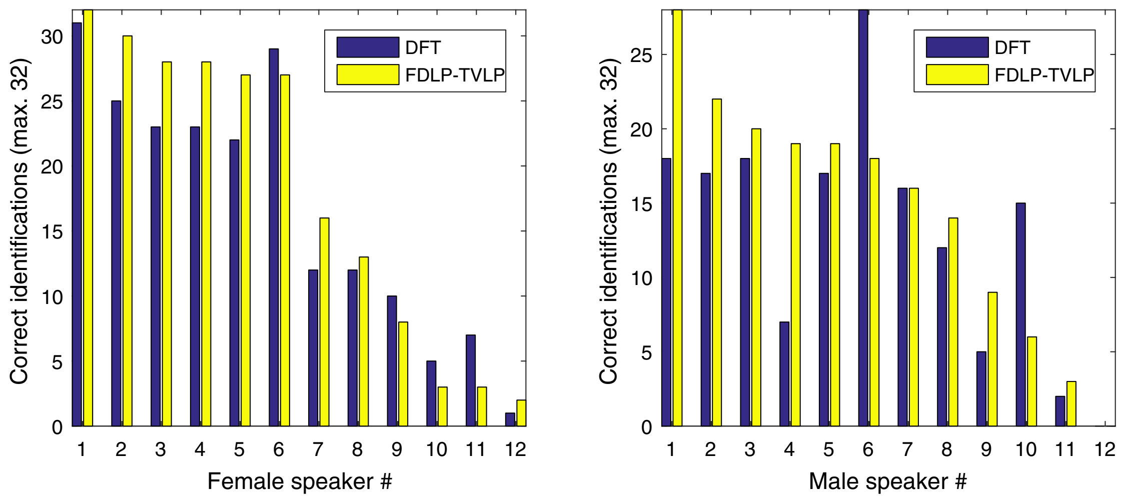


Fig. 10. Number of correct identifications for each individual speaker in normal-whisper mismatched case.

6.7. Analysis of local variability in spectrogram estimation methods

In machine learning, the well-known over-fitting vs. under-fitting trade-off, also know as the *bias-variance* trade-off, relates to generalization of models beyond a given training set. Models with a comparatively larger number of degrees of freedom tend to produce good results on training data (low bias) but fail to generalize (high variance). Being constrained by the low-order polynomial functions, the TVLP models addressed in this study are intuitively more rigid in comparison to the traditional way of extracting MFCCs. For this reason, we expect them to be less sensitive to acoustic mismatch between enrollment and test utterances, including changes in speaking style.

As our last analysis, we are interested to objectively quantify the degree of feature rigidness directly from the spectral representations. To this end, inspired by the widespread use of *Laplace operator* in image processing (Wang, 2007), and by viewing spectrograms as images, we adopt discrete Laplacians to quantify the rigidness of spectrogram representation of speech signals both in time and frequency variables obtained by different spectrum estimators. In specific, we use the average value of absolute values of Laplacian evaluated at all points of spectrograms (excluding non-speech segments). The discrete Laplacian $\mathscr{L}$ is defined as,

$$\mathscr{L}(t,f) = S(t-1,f) + S(t+1,f) + S(t,f-1) + S(t,f+1) - 4S(t,f),$$

where *t* and *f* refer to indices of time and frequency values, respectively, and where $S(t, f)$ is a speech spectrogram. Furthermore, to measure variability along one dimension only, we similarly use,

$$\begin{aligned}\mathscr{L}_t(t,f) &= S(t-1,f) + S(t+1,f) - 2S(t,f) \quad \text{and}\\ \mathscr{L}_f(t,f) &= S(t,f-1) + S(t,f+1) - 2S(t,f).\end{aligned}$$

We computed the spectrograms of the UBM data (TIMIT) using DFT, LP, and FDLP-TVLP methods. The average absolute Laplacians are presented in Table 8. In comparison to DFT, we find that LP helps to reduce the local variability in time (mean($|\mathscr{L}_f|$)) as it smooths spectra in frequency. As a side product, it also reduces variability in time (mean ($|\mathscr{L}_t|$)) because the noisy values of spectrogram get removed. The smoothing in time in FDLP-TVLP causes a large drop to variability in time while the variability in frequency is similar to the LP method.

7. Conclusions

Significant advancements on speaker recognition research have been made in recent years by speaker modeling using i-vector and DNN technology, yet mismatch conditions due to the intrinsic and extrinsic variabilities remain as a major cause of performance degradation. In the current study, we addressed the problem of mismatch arising from a specific speaking style, whispering. Besides providing an up-to-date and self-contained tutorial survey on speaker recognition from whispered speech, we introduced a new speech modeling technique that involves a

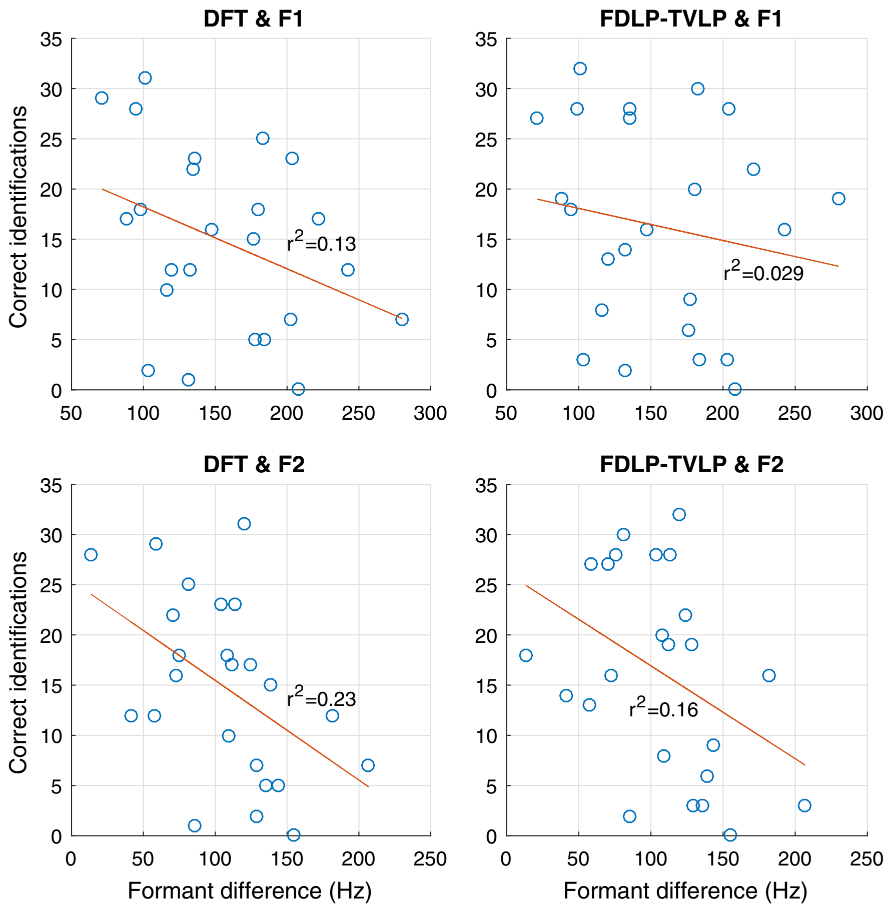


Fig. 11. The number of correct identifications for a speaker versus mean difference of formants frequencies of whispered and normal speech. Results are shown for F1 (top) and F2 (bottom) and for feature extraction methods DFT (left) and FDLP-TVLP (right). If the formant differences are large for a speaker, the number of correct identifications is more likely to be small.

long-term speech analysis based on a joint utilization of *frequency domain linear prediction* and *time-varying linear prediction* (FDLP-TVLP).

Our speaker recognition experiments on the CHAINS corpus indicate that speaker recognition from whispered speech can benefit from using FDLP-TVLP when the control parameters of the model are properly set. We made the following conclusions regarding the parameter choices. The number of basis functions for the proposed TVLP method should be about one third of the number of short-time frames in the superframe. With an experiment using four basis functions, we conclude that recognition performances do not depend much on the type of the basis function. We recommend to use simple monomial bases. Further, we experimented with different model orders for FDLP and TVLP, and we found that as long as the model orders are above 24 and 32 (assuming 16 kHz sampling rate) respectively, performance remains high.

In comparison with baseline reference features, we have found that the FDLP-TVLP feature performs considerably better than standard MFCC and LP-based MFCC features for speaker recognition from whispered speech. On the other hand, we observe a small performance degradation with normal voice. Also, the results obtained with the proposed feature have shown considerable improvement over closely related FDLP and 2DAR feature, and these indicate that speech modeling including the *time-varying* form of linear prediction helps for the recognition of whispering speaker. Interestingly, the recognition performance for whispered female voice is better than for the male voice. This finding contradicts with the usual observation in speaker recognition experiments where recognition of female speakers is more difficult than male speakers.

From speaker-by-speaker analysis of speaker identification performance, we observed considerable accuracy differences across the speakers. This suggests that the articulatory process for producing whispered voice is highly dependent on the individual person and evidently, some speakers are naturally good at disguising themselves by producing close to unidentifiable whispered voice. From the more detailed analysis, we found that a small part of individual differences can be explained by the amount of changes in formant frequencies between the normal and whispered speaking styles.

While our preliminary study on whispered speech showed promising results, we are aware of the following limitations planned to be addressed in future studies. Firstly, although the current study shows moderate improvement over baseline, the identification accuracy for *normal vs. whisper* condition is almost half of the accuracy obtained for the non-mismatched *normal vs. normal* condition. One reason for this is the absence of whispered data in the back-end training where we used

Table 7
Speaker verification equal error rates (%) with 95% confidence intervals (Bengio and Mariéthoz, 2004) for different spectrum estimation methods.

Method	Normal vs. normal		
	Females	Males	All
DFT	2.67 ± 0.50	**2.50** ± 0.44	**2.57** ± 0.33
LP	**2.48** ± 0.49	3.11 ± 0.49	2.84 ± 0.35
LP-α (α = 0.05)	3.15 ± 0.55	3.37 ± 0.51	3.45 ± 0.38
MVDR	3.56 ± 0.58	3.61 ± 0.52	3.61 ± 0.39
FDLP	3.82 ± 0.60	4.58 ± 0.59	4.30 ± 0.42
2DAR	3.34 ± 0.56	2.97 ± 0.48	3.10 ± 0.36
FDLP-TVLP	3.06 ± 0.54	3.33 ± 0.50	3.27 ± 0.37
FDLP-TVLP-α	3.78 ± 0.60	3.91 ± 0.55	3.86 ± 0.40

Method	Normal vs. whisper		
	Females	Males	All
DFT	26.24 ± 1.36	29.89 ± 1.28	29.69 ± 0.95
LP	26.32 ± 1.36	30.38 ± 1.28	28.38 ± 0.93
LP-α (α = 0.05)	26.12 ± 1.36	30.38 ± 1.28	28.91 ± 0.94
MVDR	24.90 ± 1.34	31.45 ± 1.29	28.13 ± 0.93
FDLP	26.70 ± 1.37	31.69 ± 1.30	30.33 ± 0.95
2DAR	25.38 ± 1.35	29.69 ± 1.27	28.43 ± 0.93
FDLP-TVLP	**24.84** ± 1.34	**29.08** ± 1.27	**27.48** ± 0.92
FDLP-TVLP-α	25.96 ± 1.36	30.32 ± 1.28	28.64 ± 0.94

the TIMIT data, which is well suited only for *normal vs. normal* condition. Secondly, for processing whispered speech, we have applied exactly same processing steps as used for the normal speech. One advantage of this approach is that it does not use knowledge of the underlying speaking style during processing, however, at the cost of a possible performance loss when compared to speaking style specific processing used jointly with a speaking style detector. Thirdly, the current study uses GMM-UBM framework which does not explicitly consider any channel or session variability compensation technique as used in i-vector-PLDA frameworks. For such advanced systems, preparing suitable data recipe for training parameters and hyper-parameters is difficult due to the lack of appropriate and adequate data. Finally, the study was conducted on a relatively small dataset requiring further experiments to be conducted to generalize the existing results.

Table 8
Analysis of local variability in spectrograms obtained using DFT, LP, and FDLP-TVLP spectrum estimators. Variabilities are measured as average absolute values of Laplacians extracted from speech spectrograms. Laplacian $\mathscr{L}$ is used the measure variability jointly in both dimensions and $\mathscr{L}_t$ and $\mathscr{L}_f$ are used to measure variability independently in time an frequency, respectively. As the means are computed over a large dataset, standard errors of the means in all cases are less than 0.01, making all the values significantly different from each other.

	mean($\lvert\mathscr{L}\rvert$)	mean($\lvert\mathscr{L}_t\rvert$)	mean($\lvert\mathscr{L}_f\rvert$)
DFT	33.36	18.52	18.76
LP	13.88	9.49	5.20
FDLP-TVLP	7.09	2.90	4.78

We found that comparison of findings with the existing studies on speaker recognition from whispered speech is difficult due to the lack of commonly used data sets and evaluation protocols. In addition to having standard evaluation protocols, the research community would benefit from a large publicly available corpus containing recordings of both normal and whispered speech. A larger corpus would allow the use of more data-intensive methods and would make the evaluation of research findings more reliable.

Acknowledgements

This work was partially funded by Academy of Finland (projects 309629 and 312490).

Appendix A. Details on aligning normal and whispered speech

A1. Frame-to-frame distance function

To perform time alignment of normal and whispered speech with dynamic time warping (DTW), we defined a frame-to-frame distance function d given by

$$d(\text{frame}_i, \text{frame}_j) = |\text{F1}_i - \text{F1}_j| + |\text{F2}_i - \text{F2}_j| + |\text{F3}_i - \text{F3}_j| + |E_i - E_j| + 500,$$

where F1–F3 are the formant (center) frequencies in Hz and E is the log energy of a frame computed between 4 kHz and 8 kHz. That is, the distances are based on computing absolute differences of the formant frequencies and the log energy values. The reason for excluding low frequencies (0–4 kHz) from the energy computation is that, due to lack of the fundamental frequency, the energy in whispered speech differs more from that of normal speech in the low frequency range than in the high frequency range. Before distance computation, the log energies are shifted and scaled so that for each sentence the minimum log energy is 0 and the maximum log energy is 1000. In addition, we add a constant term 500 to all of the distances to reduce the amount of time stretching in the DTW algorithm.

A2. Automatic alignment quality detection

The detection of well aligned segments consists of three steps. First, energy based speech activity detection is performed for both normal and whispered speech to discard non-speech frames. Second, we discard those segments whose formant tracks can be considered unreliable. The algorithm for detecting formant tracking quality uses 30-frame long sliding window to discard windows that contain too many sudden jumps (more than 200 Hz) between the consecutive formant frequencies. Finally, we discard segments that contain considerable amount of time stretching (repeated frames). More precisely (within a 30-frame window), if the sum of the repeated frames in aligned normal and aligned whispered speech is more than 8, the window will be discarded. An example of a segment that contains too much time stretching can be seen near the 3 s mark in the third panel of Fig. 2.

References

Athineos, M., Ellis, D.P., 2007. Autoregressive modeling of temporal envelopes. IEEE Trans. Signal Process. 55 (11), 5237–5245.

Athineos, M., Hermansky, H., Ellis, D., 2004. PLP^2: autoregressive modeling of auditory-like 2-D spectro-temporal patterns. ISCA Tutorial and Research Workshop (ITRW) on Statistical and Perceptual Audio Processing.

Bengio, S., Mariéthoz, J., 2004. A statistical significance test for person authentication. Proceedings of Odyssey 2004: The Speaker and Language Recognition Workshop.

Boersma, P., 2017. Praat: doing phonetics by computer (version 6.0.29. http://www.praat.org/.

Cummins, F., Grimaldi, M., Leonard, T., Simko, J., 2006. The chains corpus: characterizing individual speakers. Proc of SPECOM. Vol. 6. pp. 431–435.

Davis, S., Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. 28 (4), 357–366.

Dehak, N., Kenny, P.J., Dehak, R., Dumouchel, P., Ouellet, P., 2011. Front-end factor analysis for speaker verification. IEEE Trans. Audio Speech Lang. Process. 19 (4), 788–798.

Delgado, H., Todisco, M., Sahidullah, M., Sarkar, A.K., Evans, N., Kinnunen, T., Tan, Z.-H., 2016. Further optimisations of constant Q cepstral processing for integrated utterance and text-dependent speaker verification. IEEE Spoken Language Technology (SLT) Workshop.

Dey, S., Motlicek, P., Madikeri, S., Ferras, M., 2017. Template-matching for text-dependent speaker verification. Speech Commun. 88, 96–105.

Ellis, D., 2003. Dynamic time warp (dtw) in matlab. Web resource, available: http://www.ee.columbia.edu/dpwe/resources/matlab/dtw.

Fan, X., Hansen, J.H., 2008a. Speaker identification for whispered speech based on frequency warping and score competition. In: INTERSPEECH, pp. 1313–1316.

Fan, X., Hansen, J.H., 2009b. Speaker identification for whispered speech using modified temporal patterns and MFCCs. In: INTERSPEECH, pp. 896–899.

Fan, X., Hansen, J.H., 2009. Speaker identification with whispered speech based on modified LFCC parameters and feature mapping. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009). IEEE, pp. 4553–4556.

Fan, X., Hansen, J.H., 2010. Acoustic analysis for speaker identification of whispered speech. IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2010). IEEE, pp. 5046–5049.

Fan, X., Hansen, J.H., 2011. Speaker identification for whispered speech using a training feature transformation from neutral to whisper. INTERSPEECH. pp. 2425–2428.

Fan, X., Hansen, J.H., 2011. Speaker identification within whispered speech audio streams. IEEE Trans. Audio Speech Lang. Process. 19 (5), 1408–1421.

Fan, X., Hansen, J.H., 2013. Acoustic analysis and feature transformation from neutral to whisper for speaker identification within whispered speech audio streams. Speech Commun. 55 (1), 119–134.

Friedman, J., Hastie, T., Tibshirani, R., 2009. The Elements of Statistical Learning, Second edition. Springer Series in Statistics, New York.

Ganapathy, S., Mallidi, S.H., Hermansky, H., 2014. Robust feature extraction using modulation filtering of autoregressive models. IEEE/ACM Trans. Audio Speech Lang. Process. 22 (8), 1285–1295.

Garcia-Romero, D., Zhou, X., Espy-Wilson, C.Y., 2012. Multicondition training of Gaussian PLDA models in i-vector space for noise and reverberation robust speaker recognition. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012). IEEE, pp. 4257–4260.

Grenier, Y., 1983. Time-dependent arma modeling of nonstationary signals. IEEE Trans. Acoust. 31 (4), 899–911.

Grimaldi, M., Cummins, F., 2008. Speaker identification using instantaneous frequencies. IEEE Trans. Audio Speech Lang. Process. 16 (6), 1097–1111.

Hall, M.G., Oppenheim, A.V., Willsky, A.S., 1983. Time-varying parametric modeling of speech. Signal Process. 5 (3), 267–285.

Hansen, J.H., Nandwana, M.K., Shokouhi, N., 2017. Analysis of human scream and its impact on text-independent speaker verification. J. Acoust. Soc. Am. 141 (4), 2957–2967.

Hansen, J.H.L., Hasan, T., 2015. Speaker recognition by machines and humans: a tutorial review. IEEE Signal Process. Mag. 32 (6), 74–99.

Hautamäki, R.G., Sahidullah, M., Hautamäki, V., Kinnunen, T., 2017. Acoustical and perceptual study of voice disguise by age modification in speaker verification. Speech Commun. 95, 1–15.

Heeren, W.F., 2015. Vocalic correlates of pitch in whispered versus normal speech. J. Acoust. Soc. Am. 138 (6), 3800–3810.

Heigold, G., Moreno, I., Bengio, S., Shazeer, N., 2016. End-to-end text-dependent speaker verification. Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on. IEEE, pp. 5115–5119.

Hermansky, H., Morgan, N., 1994. RASTA Processing of speech. IEEE Trans. Speech Audio Process. 2 (4), 578–589.

Herre, J., Johnston, J.D., 1996. Enhancing the performance of perceptual audio coders by using temporal noise shaping (TNS). Audio Engineering Society Convention 101. Audio Engineering Society.

Higashikawa, M., Nakai, K., Sakakura, A., Takahashi, H., 1996. Perceived pitch of whispered vowels-relationship with formant frequencies: a preliminary study. J. Voice 10 (2), 155–158.

Ito, T., Takeda, K., Itakura, F., 2005. Analysis and recognition of whispered speech. Speech Commun. 45 (2), 139–152.

Jawarkar, N.P., Holambe, R.S., Basu, T.K., 2013. Speaker identification using whispered speech. International Conference on Communication Systems and Network Technologies (CSNT 2013). IEEE, pp. 778–781.

Jin, Q., Jou, S.-C.S., Schultz, T., 2007. Whispering speaker identification. IEEE International Conference on Multimedia and Expo. IEEE, pp. 1027–1030.

Junqua, J.-C., 1993. The lombard reflex and its role on human listeners and automatic speech recognizers. J. Acoust. Soc. Am. 93 (1), 510–524.

Kenny, P., 2006. Joint factor analysis of speaker and session variability: theory and algorithms. Technical Report CRIM-06/08-14.

Kenny, P., 2010. Bayesian speaker verification with heavy-tailed priors. Odyssey 2010: The Speaker and Language Recognition Workshop, Brno, Czech Republic. pp. 14.

Kinnunen, T., Li, H., 2010. An overview of text-independent speaker recognition: from features to supervectors. Speech Commun. 52 (1), 12–40.

Kumaresan, R., Rao, A., 1999. Model-based approach to envelope and positive instantaneous frequency estimation of signals with speech applications. J. Acoust. Soc. Am. 105 (3), 1912–1924.

Künzel, H.J., 2000. Effects of voice disguise on speaking fundamental frequency. Int. J. Speech Lang. Law 7 (2), 149–179.

Lee, C.-H., Gauvain, J.-L., 1993. Speaker adaptation based on map estimation of HMM parameters. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-93), 1993. Vol. 2. IEEE, pp. 558–561.

Lee, P.X., Wee, D., Toh, H.S.Y., Lim, B.P., Chen, N.F., Ma, B., 2014. A whispered mandarin corpus for speech technology applications. INTERSPEECH. pp. 1598–1602.

Li, L., Wang, D., Zhang, C., Zheng, T.F., 2016. Improving short utterance speaker recognition by modeling speech unit classes. IEEE/ACM Trans. Audio Speech Lang. Process. 24 (6), 1129–1139.

Lim, B.P., 2011. Computational differences between whispered and non-whispered speech. University of Illinois at Urbana-Champaign.

Liporace, L.A., 1975. Linear estimation of nonstationary signals. J. Acoust. Soc. Am. 58 (6), 1288–1295.

Liu, G., Lei, Y., Hansen, J.H., 2012. Robust feature front-end for speaker identification. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012). IEEE, pp. 4233–4236.

Liu, Y., Qian, Y., Chen, N., Fu, T., Zhang, Y., Yu, K., 2015. Deep feature for text-dependent speaker verification. Speech Commun. 73, 1–13.

Makhoul, J., 1975. Linear prediction: a tutorial review. Proc. IEEE 63 (4), 561–580.

Masthoff, H., 1996. A report on a voice disguise experiment. Forensic Linguistics 3, 160–167.

Murthi, M.N., Rao, B.D., 2000. All-pole modeling of speech based on the minimum variance distortionless response spectrum. IEEE Trans. Speech Audio Process. 8 (3), 221–239.

Pohjalainen, J., Hanilçi, C., Kinnunen, T., Alku, P., 2014. Mixture linear prediction in speaker verification under vocal effort mismatch. IEEE Signal Process. Lett. 21 (12), 1516–1520.

Prince, S.J., Elder, J.H., 2007. Probabilistic linear discriminant analysis for inferences about identity. IEEE 11th International Conference on Computer Vision (ICCV 2007). IEEE, pp. 1–8.

Rajan, P., Afanasyev, A., Hautamäki, V., Kinnunen, T., 2014. From single to multiple enrollment i-vectors: practical PLDA scoring variants for speaker verification. Digit. Signal Process. 31, 93–101.

Rajan, P., Kinnunen, T., Hautamäki, V., 2013. Effect of multicondition training on i-vector PLDA configurations for speaker recognition. INTERSPEECH. pp. 3694–3697.

Reynolds, D.A., Quatieri, T.F., Dunn, R.B., 2000. Speaker verification using adapted gaussian mixture models. Digit. Signal Process. 10 (1–3), 19–41.

Reynolds, D.A., Rose, R.C., 1995. Robust text-independent speaker identification using gaussian mixture speaker models. IEEE Trans. Speech Audio Process. 3 (1), 72–83.

Rudoy, D., Quatieri, T.F., Wolfe, P.J., 2011. Time-varying autoregressions in speech: detection theory and applications. IEEE Trans. Audio Speech Lang. Process. 19 (4), 977–989.

Sadjadi, S.O., Hansen, J.H., 2015. Mean hilbert envelope coefficients (MHEC) for robust speaker and language identification. Speech Commun. 72, 138–148.

Saeidi, R., Alku, P., Bäckström, T., 2016. Feature extraction using power-law adjusted linear prediction with application to speaker recognition under severe vocal effort mismatch. IEEE/ACM Trans. Audio Speech Lang. Process. 24 (1), 42–53.

Sarria-Paja, M., Falk, T.H., 2017. Fusion of auditory inspired amplitude modulation spectrum and cepstral features for whispered and normal speech speaker verification. Comput. Speech Lang. 45, 437–456.

Sarria-Paja, M., Falk, T.H., O'Shaughnessy, D., 2013. Whispered speaker verification and gender detection using weighted instantaneous frequencies. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013). IEEE, pp. 7209–7213.

Sarria-Paja, M., Senoussaoui, M., Falk, T.H., 2015. The effects of whispered speech on state-of-the-art voice based biometrics systems. IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE 2015). IEEE, pp. 1254–1259.

Sarria-Paja, M., Senoussaoui, M., O'Shaughnessy, D., Falk, T.H., 2016. Feature mapping, score-, and feature-level fusion for improved normal and whispered speech speaker verification. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016). IEEE, pp. 5480–5484.

Sarria-Paja, M.O., Falk, T.H., 2015. Strategies to enhance whispered speech speaker verification: a comparative analysis. Canadian Acoust. 43 (4), 31–45.

Sharma, R., Prasanna, S., Bhukya, R., Das, R., 2017. Analysis of the intrinsic mode functions for speaker information. Speech Commun. 91, 1–16.

Shue, Y.-L., Keating, P., Vicenik, C., Yu, K., 2011. Voicesauce: A program for voice analysis. Proceedings of the ICPhS XVII. pp. 1846–1849.

Snyder, D., Ghahremani, P., Povey, D., Garcia-Romero, D., Carmiel, Y., Khudanpur, S., 2016. Deep neural network-based speaker embeddings for end-to-end speaker verification. Spoken Language Technology Workshop (SLT), 2016 IEEE. IEEE, pp. 165–170.

Tartter, V.C., 1989. Whats in a whisper? J. Acoust. Soc. Am. 86 (5), 1678–1683.

Thomas, S., Ganapathy, S., Hermansky, H., 2008. Recognition of reverberant speech using frequency domain linear prediction. IEEE Signal Process. Lett. 15, 681–684.

Vestman, V., Gowda, D., Sahidullah, M., Alku, P., Kinnunen, T., 2017. Time-varying autoregressions for speaker verification in reverberant conditions. INTERSPEECH. pp. 1512–1516.

Wang, X., 2007. Laplacian operator-based edge detectors. IEEE Trans. Pattern Anal. Mach. Intell. 29 (5), 886–890.

Weninger, F., Geiger, J., Wöllmer, M., Schuller, B., Rigoll, G., 2011. The munich 2011 chime challenge contribution: Nmf-blstm speech enhancement and recognition for reverberated multisource environments. In: Machine Listening in Multisource Environments.

Zeinali, H., Sameti, H., Burget, L., 2017. HMM-Based phrase-independent i-vector extractor for text-dependent speaker verification. IEEE/ACM Trans. Audio Speech Lang. Process. 25 (7), 1421–1435.

Zeinali, H., Sameti, H., Burget, L., Cernockỳ, J., Maghsoodi, N., Matejka, P., 2016. i-vector/HMM based text-dependent speaker verification system for reddots challenge. INTERSPEECH. pp. 440–444.

Zhang, C., Hansen, J.H., 2007. Analysis and classification of speech mode: whispered through shouted. INTERSPEECH. Vol. 7. pp. 2289–2292.

Zhang, C., Hansen, J.H., 2009. Advancements in whisper-island detection within normally phonated audio streams. INTERSPEECH. pp. 860–863.

Paper **III**

Odyssey 2018 The Speaker and Language Recognition Workshop
26-29 June 2018, Les Sables d'Olonne, France

Supervector Compression Strategies to Speed up I-Vector System Development

Ville Vestman, Tomi Kinnunen

University of Eastern Finland, Finland
vvestman@cs.uef.fi, tkinnu@cs.uef.fi

Abstract

The front-end factor analysis (FEFA), an extension of principal component analysis (PPCA) tailored to be used with Gaussian mixture models (GMMs), is currently the prevalent approach to extract compact utterance-level features (i-vectors) for automatic speaker verification (ASV) systems. Little research has been conducted comparing FEFA to the conventional PPCA applied to maximum a posteriori (MAP) adapted GMM supervectors. We study several alternative methods, including PPCA, factor analysis (FA), and two supervised approaches, supervised PPCA (SPPCA) and the recently proposed probabilistic partial least squares (PPLS), to compress MAP-adapted GMM supervectors. The resulting i-vectors are used in ASV tasks with a probabilistic linear discriminant analysis (PLDA) back-end. We experiment on two different datasets, on the telephone condition of NIST SRE 2010 and on the recent VoxCeleb corpus collected from YouTube videos containing celebrity interviews recorded in various acoustical and technical conditions. The results suggest that, in terms of ASV accuracy, the supervector compression approaches are on a par with FEFA. The supervised approaches did not result in improved performance. In comparison to FEFA, we obtained more than hundred-fold (100x) speedups in the total variability model (TVM) training using the PPCA and FA supervector compression approaches.

1. Introduction

Modern text-independent automatic speaker verification (ASV) relies heavily on the use of *identity vectors* (i-vectors) [1, 2]. I-vectors are compact representations of speech utterances containing useful information for speech-related classification tasks. The i-vector extraction pipeline involves many steps starting from the extraction of acoustic features such as *Mel-frequency cepstral coefficients* (MFCCs), followed by the extraction of *sufficient statistics* with the aid of an *universal background model* (UBM), typically a *Gaussian mixture model* (GMM) [3] or a *deep neural network* (DNN) model [4]. Sufficient statistics are then used to extract an i-vector, a fixed-length representation of an utterance, using a pre-trained *total variability model* (TVM) that models the distribution of utterance-specific GMM supervectors.

The development and optimization of an i-vector based ASV system consists of a multiple time-consuming steps. The most notable ones are extraction of acoustic features and sufficient statistics, and training of UBM and TVM. The two former require processing of a large number of speech files and TVM training is among the most time consuming parts of the system development process. Thus, by reducing TVM training time, a meaningful positive effect to the total development time of ASV system can be achieved [5, 6]. This is particularly beneficial in studies focused on the acoustic front-end optimizations when one has to retrain the entire system when feature extractor is changed [7].

Previous studies on rapid i-vector extraction have primarily optimized computations in the standard *front-end factor analysis* (FEFA) approach [1, 2] by adopting new computational algorithms, often approximative in nature [6, 8, 9]. In this study, however, we focus on an alternative and straightforward compression of classic *maximum a posteriori* (MAP) [3] adapted GMM supervectors with a goal of obtaining fast execution times without compromising on ASV accuracy. In fact, before FEFA, and its predecessor, *joint factor analysis* (JFA) [10], became prevalent, MAP adapted supervectors were commonly used with *support vector machine* (SVM) to do speaker classification [11]. Recently, however, the use of MAP adapted supervectors has been less common.

Supervector compression, for example by using *probabilistic principal component analysis* (PPCA) [12, 13], provides a large computational saving in TVM estimation over FEFA [14]. In FEFA, the posterior covariance matrix needed for i-vector extraction is *utterance-dependent*, while in the supervector compression methods addressed in this study, covariance is shared among all speech utterances, which greatly reduces computation. As the TVM training set may consist of tens of thousands of utterances, the resulting computational saving is considerable [14].

The closest prior work similar in spirit to ours are [14] and [15], which we extend in many ways. In these two studies, the training of TVM is performed using PPCA. The acoustic feature vectors are *hardly aligned* to a single UBM component. If they were *softly aligned*, this approach would equal to using MAP adapted supervectors with a *relevance factor* of 0 [16]. Differing from [14] and [15], we use MAP adapted supervectors to train TVM and we study how the choice of relevance factor affects the system performance.

Recently [17], TVM estimation using *probabilistic partial least squares* (PPLS) was proposed as an alternative to FEFA. PPLS compresses supervectors in a *supervised* way by taking advantage of the speaker labels in the training set. In the current study, we attempt to validate the positive results [17] obtained for Chinese mandarin corpus by using datasets containing English speech instead. In [18], supervision is added to the total variability matrix training by deploying *supervised* PPCA (SPPCA) [19]. The SPPCA model is fundamentally the same as the PPLS model, with a difference in what has been used as the supervision data; PPLS has been used directly with speaker labels [17], while SPPCA has been utilized with speaker-specific (not just utterance-specific) sufficient statistics [18]. In the current work, we study the use of SPPCA for supervector compression.

In addition to the above models, we adopt standard *fac-*

This work was partially funded by Academy of Finland (project 309629).

tor analysis (FA) for supervector compression. Note, that this differs from the FEFA framework: In FEFA, the TVM training is based on the maximization of posterior probabilities of acoustic feature vectors, assumed to have been generated by a GMM [20], whereas in FA (and PPCA), maximization is performed with respect to posterior probabilities of supervectors [21].

We conduct comparisons of different methods in ASV setting using a recently released *VoxCeleb* corpus [22]. The corpus contains "real-world" utterances obtained from YouTube videos of celebrity interviews using a fully automated data collection pipeline. The data is challenging for ASV due to large intra-speaker variability caused by large differences in environments, speaking styles, and technical conditions. In addition to VoxCeleb, we validate our findings with the telephone condition of NIST 2010 Speaker Recognition Evaluation corpus.

Our contributions can be summarized as follows. First, we present all the selected methods in an unified notation highlighting the important formulas regarding their implementation. Second, we aim at validating the results claimed in [17] regarding the recently proposed PPLS method on different corpora, and we extend the study by introducing the weighting scheme proposed in [18]. Third, we implement and test SPPCA in the supervector compression domain. Fourth, we compare all the methods in terms of ASV performances and training times of total variability models. Lastly, we propose a slight simplification to the maximization principle of TVM training.

2. I-Vector Extraction by Front-End Factor Analysis

Front-end factor analysis (FEFA) [1] is the current standard method for extracting utterance level features known as *i-vectors*. In FEFA, a supervector $\boldsymbol{m}(u)$ of an utterance u is modeled as

$$\widehat{\boldsymbol{m}}(u) = \boldsymbol{\mu} + V\boldsymbol{w}(u),$$

where $\boldsymbol{\mu} \in \mathbb{R}^{h\times 1}$ is an utterance-independent bias supervector, $\boldsymbol{w}(u) \in \mathbb{R}^{d\times 1}$ is a low dimensional representation of an utterance supervector, *i.e.*, an i-vector, and $V \in \mathbb{R}^{h\times d}$ is a mapping between low and high dimensional spaces known as *total variability matrix*. The mathematical foundation of FEFA is presented in detail in [20].

The traditional way of TVM estimation and i-vector extraction begins with computing frame posterior probabilities for short-term spectral features (*e.g.* MFCCs) of an utterance with each Gaussian component of UBM. These posteriors are then used to compute zeroth and first order sufficient statistics,

$$n_c = \sum_{t=1}^{T} p_t(c),$$

$$\boldsymbol{f}_c = \sum_{t=1}^{T} p_t(c)\boldsymbol{x_t},$$

where $\boldsymbol{x_t}$ is the t^{th} feature vector of the utterance and $p_t(c)$ is the posterior probability of t^{th} vector belonging to c^{th} component of UBM, computed with the aid of Bayes' rule.

Assuming that the prior distribution $p(\boldsymbol{w}(u))$ is stardard normal, it can be shown [20] that

$$p(\boldsymbol{w}(u)|X(u), V) = \mathcal{N}(\boldsymbol{\mu}(u), \Sigma(u)),$$

where $X(u) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ is a sequence of all feature vectors in the utterance u and where

I-Vector Extraction

$$\boxed{\begin{aligned}\Sigma(u) &= \left(I + \sum_{c=1}^{C} n_c(u) V_c^{\intercal}\Sigma_c^{-1}V_c\right)^{-1}, \\ \boldsymbol{\mu}(u) &= \Sigma(u)\sum_{c=1}^{C} V_c^{\intercal}\Sigma_c^{-1}(\boldsymbol{f}_c(u) - n_c(u)\boldsymbol{\mu}_c).\end{aligned}}$$

In the above equations, Σ_c is the covariance matrix of the c^{th} UBM component, and V_c and $\boldsymbol{\mu}_c$ are component-wise representations of V and $\boldsymbol{\mu}$ so that

$$V = \begin{bmatrix} V_1 \\ \vdots \\ V_C \end{bmatrix} \quad \text{and} \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \vdots \\ \boldsymbol{\mu}_C \end{bmatrix},$$

where the vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_C$ are the means of the UBM components. Mean $\boldsymbol{\mu}(u)$ of the posterior i-vector distribution is the i-vector of the utterance u.

The matrix V is estimated using an offline training set of U utterances by maximizing

$$\sum_{u=1}^{U} \mathbb{E}[\ln p(X(u)|\boldsymbol{w}(u), V)], \tag{1}$$

where the expectations are taken with respect to posterior i-vector distributions [20]. This leads to an update formula

Update Formula for V

$$\boxed{\begin{aligned}&V_c = \left(\sum_{u=1}^{U} \boldsymbol{f}_c(u)\boldsymbol{\mu}(u)^{\intercal}\right)\left(\sum_{u=1}^{U} n_c(u)\mathbb{E}_{\mu\mu}(u)\right)^{-1}, \\ &\mathbb{E}_{\mu\mu}(u) = \Sigma(u) + \boldsymbol{\mu}(u)\boldsymbol{\mu}(u)^{\intercal}.\end{aligned}}$$

Training of V is iterative; one iteration consists of computing $\Sigma(u)$, $\boldsymbol{\mu}(u)$, and $\mathbb{E}_{\mu\mu}(u)$ for all training utterances by keeping V fixed and then updating V using the computed values. During the first iteration, parameters of posterior distributions are computed using a randomly initialized matrix V.

3. I-Vector Extraction by Supervector Compression

In this section, we present multiple approaches to compressing MAP adapted GMM supervectors to low-dimensional representations, which we will also refer as "i-vectors". Unlike FEFA, these approaches do not assume the underlying speaker model to be GMM.

In relevance MAP, the adapted mean vectors $\hat{\boldsymbol{\mu}}_c$, $c = 1, \ldots, C$, for utterance's GMM are obtained from UBM by computing

$$\hat{\boldsymbol{\mu}}_c = \alpha_c\boldsymbol{f}_c + (1 - \alpha_c)\boldsymbol{\mu}_c,$$

where

$$\alpha_c = \frac{n_c}{n_c + r}$$

and $r \geq 0$ is the *relevance factor* to be optimized [3]. When $r \to 0$, then $\alpha_c \to 1$, and when $r = 0$, the mean vectors are solely determined by the sufficient statistics $\boldsymbol{f}_c$. If r is large, then the adapted mean vectors remain close to UBM's mean vectors. Adapted mean vectors $\hat{\boldsymbol{\mu}}_c$ are concatenated together to form a *supervector* for the utterance.

3.1. Principal Component Analysis

Being one of the most commonly used dimension reduction techniques, we include the conventional *principal component analysis* (PCA) [23] as a baseline method for supervector compression. The PCA transformation matrix, consisting of eigenvectors of data covariance matrix, can be used to transform high dimensional supervectors to low dimensional i-vectors. In this study, we use the standard *singular value decomposition* (SVD) approach for PCA computation. However, for high dimensional data, PCA could be computed more efficiently by adopting iterative PCA algorithms [12].

3.2. Probabilistic Principal Component Analysis

Probabilistic principal component analysis (PPCA) [12, 13] models observations using a linear-Gaussian framework. In this section, we present a compact self-contained formulation of PPCA in the context of supervectors. The rest of the methods discussed in Section 3 are formulated similarly and their theory can be easily formulated using PPCA as a starting point.

In PPCA, supervectors are modeled as

$$\boldsymbol{m}(u) = V\boldsymbol{w}(u) + \boldsymbol{\epsilon}, \tag{2}$$

where $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, I)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 I)$. For brevity, we have omitted the bias term $\boldsymbol{\mu}$ from the right-hand side of the equation by assuming that supervectors have been centered using the data mean computed from the training data.

By using properties of normally distributed random variables and by assuming that V is given, from (2) it follows that

$$p(\boldsymbol{m}(u)|\boldsymbol{w}(u)) = \mathcal{N}(V\boldsymbol{w}(u), \sigma^2 I).$$

Further, in Appendix A, we show that

$$p(\boldsymbol{w}(u)|\boldsymbol{m}(u)) = \mathcal{N}(\boldsymbol{\mu}(u), \Sigma), \tag{3}$$

where

I-Vector Extraction

$$\boxed{\begin{aligned} \Sigma &= \left(I + \frac{1}{\sigma^2}V^\intercal V\right)^{-1}, \\ \boldsymbol{\mu}(u) &= \frac{1}{\sigma^2}\Sigma V^\intercal \boldsymbol{m}(u). \end{aligned}} \tag{4}$$

Unlike with FEFA, covariance Σ of the posterior i-vector distribution does not depend on the utterance. Hence, by adopting PPCA instead of FEFA, the time complexity of computing the parameters of posterior distributions drops from $O(U(CFd + Cd^2 + d^3))$ to $O(UCFd)$, where F is the dimension of acoustic feature vectors [14].

In the current work, we study two different ways of obtaining V for all the presented methods. The traditional approach (**max. principle 1**) maximizes

$$\sum_{u=1}^{U} \mathbb{E}[\ln p(\boldsymbol{m}(u)|\boldsymbol{w}(u), V)], \tag{5}$$

where expectations are taken with respect to posterior i-vector distributions (similar to (1)). We propose maximizing the sum of log-likelihoods directly (**max. principle 2**) without computing expectations by setting $\boldsymbol{w}(u) = \boldsymbol{\mu}(u)$. That is, we maximize

$$\begin{aligned} &\sum_{u=1}^{U} \ln p(\boldsymbol{m}(u)|\boldsymbol{w}(u), V) \qquad (6) \\ &= \sum_{u=1}^{U} \Big(-\frac{h}{2}\ln(2\pi\sigma^2) \\ &\quad - \frac{1}{2\sigma^2}\big(\boldsymbol{m}(u) - V\boldsymbol{\mu}(u)\big)^\intercal\big(\boldsymbol{m}(u) - V\boldsymbol{\mu}(u)\big)\Big) \\ &= -\frac{hU}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{u=1}^{U}\Big(\boldsymbol{m}(u)^\intercal \boldsymbol{m}(u) \\ &\quad - 2\boldsymbol{m}(u)^\intercal V\boldsymbol{\mu}(u) + \boldsymbol{\mu}(u)^\intercal V^\intercal V\boldsymbol{\mu}(u)\Big), \end{aligned}$$

where h is the dimension of supervectors.

By taking derivatives with respect to each variable in the matrix V and by setting them to zero, we obtain

Update Formulas for V and σ^2

$$\boxed{\begin{aligned} V &= \left(\sum_{u=1}^{U}\boldsymbol{m}(u)\boldsymbol{\mu}(u)^\intercal\right)\left(\sum_{u=1}^{U}\mathbb{E}_{\mu\mu}(u)\right)^{-1}, \\ \sigma^2 &= \frac{1}{hU}\sum_{u=1}^{U}\Big(\boldsymbol{m}(u)^\intercal\boldsymbol{m}(u) - \mathrm{tr}\big(\mathbb{E}_{\mu\mu}(u)V^\intercal V\big)\Big), \\ \mathbb{E}_{\mu\mu}(u) &= \boldsymbol{\mu}(u)\boldsymbol{\mu}(u)^\intercal \quad (\textbf{max principle 2}). \end{aligned}}$$

The traditional approach (**max principle 1**) of solving V results in the exact same formulas but with

$$\mathbb{E}_{\mu\mu}(u) = \Sigma + \boldsymbol{\mu}(u)\boldsymbol{\mu}(u)^\intercal \quad [13].$$

Similarly to FEFA, training of V is iterative. As initial values, we use random V and $\sigma^2 = 1$.

3.3. Factor Analysis

Factor analysis (FA) agrees with the model (2) of PPCA, except that the noise term $\boldsymbol{\epsilon}$ has more freedom by letting $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \Psi)$, where $\Psi \in \mathbb{R}^{h\times h}$ is diagonal instead of isotropic [13]. The training procedure is analogous to PPCA [21, pp. 585–586]. First, the parameters of posterior distributions (3) are computed as

I-Vector Extraction

$$\boxed{\begin{aligned} \Sigma &= \left(I + V^\intercal\Psi^{-1}V\right)^{-1}, \\ \boldsymbol{\mu}(u) &= \Sigma V^\intercal \Psi^{-1}\boldsymbol{m}(u). \end{aligned}}$$

Then, the model parameters are updated as follows:

Update Formulas for V and Ψ

$$\boxed{\begin{aligned} V &= \left(\sum_{u=1}^{U}\boldsymbol{m}(u)\boldsymbol{\mu}(u)^\intercal\right)\left(\sum_{u=1}^{U}\mathbb{E}_{\mu\mu}(u)\right)^{-1}, \\ \Psi &= \frac{1}{U}\sum_{u=1}^{U}\Big(\boldsymbol{m}(u)\boldsymbol{m}(u)^\intercal - V\mathbb{E}_{\mu\mu}(u)V^\intercal\Big) \odot I, \end{aligned}}$$

where $\odot$ denotes the Hadamard (element-wise) product. The update formula for matrix V is the same as with PPCA.

3.4. Supervised Approaches

The methods presented so far capture the variability between individual utterances regardless of the speaker's identity. That is, their training is unsupervised in the sense that no speaker labels are needed. This makes it convenient to apply any of the above methods to different classification tasks, but leaves an open question whether "better" i-vectors could be extracted by utilizing speaker labels. Thus, we explore two methods that include identity information to the training process of the total variability matrix V to discriminate speakers better. The explored methods, recently proposed *probabilistic partial least squares* (PPLS) [17] and *supervised PPCA* (SPPCA) [18], can both be thought as extensions of the regular PPCA. The underlying models of PPLS and SPPCA are essentially the same, where the difference is in the data used to discriminate speakers: PPLS adds discrimination by using *speaker labels* while SPPCA utilizes *speaker-dependent sufficient statistics* within the FEFA framework. In the current work, however, we apply SPPCA in the supervector context in contrast to [18], where the FEFA context is used.

In PPLS, supervector model is bundled together with a speaker label model. Speaker labels are encoded as *one-hot vectors*, $\boldsymbol{y}(u) \in \mathbb{R}^s$, where s is the number of speakers in the training set. For example, if the utterance u originates from the second speaker of the set, then $\boldsymbol{y}(u) = (0, 1, 0, \ldots, 0)^\intercal$. The speaker label model and the supervector model share the same i-vector $\boldsymbol{w}(u)$ as follows:

$$\begin{cases} \boldsymbol{m}(u) = V\boldsymbol{w}(u) + \boldsymbol{\epsilon}, & \text{(supervector model)} \\ \boldsymbol{y}(u) = Q\boldsymbol{w}(u) + \boldsymbol{\zeta}, & \text{(speaker label model)} \end{cases} \tag{7}$$

where (7) is defined in the same way as before, $Q \in \mathbb{R}^{s \times d}$ is a mapping between the i-vector space and the one-hot vector space, and $\boldsymbol{\zeta} \sim \mathcal{N}(\mathbf{0}, \rho^2 I)$.

As presented in [17] and [18], the PPLS model leads to

I-Vector Extraction

$$\boxed{\begin{aligned} \Sigma &= \left(I + \frac{1}{\sigma^2}V^\intercal V + \frac{1}{\rho^2}Q^\intercal Q\right)^{-1}, \\ \boldsymbol{\mu}(u) &= \Sigma\Big(\frac{1}{\sigma^2}V^\intercal \boldsymbol{m}(u) + \frac{1}{\rho^2}Q^\intercal \boldsymbol{y}(u)\Big) \end{aligned}} \tag{8}$$

and

Update Formulas for V, Q, σ^2, and ρ^2

$$\boxed{\begin{aligned} V &= \left(\sum_{u=1}^{U} \boldsymbol{m}(u)\boldsymbol{\mu}(u)^\intercal\right)\left(\sum_{u=1}^{U} \mathbb{E}_{\mu\mu}(u)\right)^{-1}, \\ Q &= \left(\sum_{u=1}^{U} \boldsymbol{y}(u)\boldsymbol{\mu}(u)^\intercal\right)\left(\sum_{u=1}^{U} \mathbb{E}_{\mu\mu}(u)\right)^{-1}, \\ \sigma^2 &= \frac{1}{hU}\sum_{u=1}^{U}\Big(\boldsymbol{m}(u)^\intercal\boldsymbol{m}(u) - \operatorname{tr}\big(\mathbb{E}_{\mu\mu}(u)V^\intercal V\big)\Big), \\ \rho^2 &= \frac{1}{sU}\sum_{u=1}^{U}\Big(\boldsymbol{y}(u)^\intercal\boldsymbol{y}(u) - \operatorname{tr}\big(\mathbb{E}_{\mu\mu}(u)Q^\intercal Q\big)\Big), \end{aligned}} \tag{9}$$

where, as before, we assume that all the supervectors and one-hot vectors are centered using the mean vectors calculated from the training data.

To extract i-vectors as in (8), we are required to have speaker information of the utterance stored in $\boldsymbol{y}(u)$. In the testing phase, however, we have no information about the speaker. To this end, we might simply extract the test i-vector using extraction formulas (4) for PPCA [18]. The result is not the same as with PPCA, since the training of V is influenced by the supervised approach. Another solution to deal with the lacking speaker information is to predict speaker labels as a mean of posterior distribution $p(\boldsymbol{y}(u)|\boldsymbol{m}(u))$; see the details in [17]. We found experimentally that both approaches extract exactly the same i-vectors.

The described formulations apply also for SPPCA with a difference that instead of using one-hot encoded speaker labels as vectors $\boldsymbol{y}(s)$, we use speaker dependent supervectors. A speaker dependent supervector is formed by using the acoustic features from all of the speaker's training utterances. Note that with SPPCA, h should be used instead of s in (9) and that Q's dimensionality is the same as V's.

In [18], a *weighted* SPPCA is proposed. In weighted SPPCA, (8) becomes

I-Vector Extraction

$$\boxed{\begin{aligned} \Sigma &= \left(I + \frac{1}{\sigma^2}V^\intercal V + \frac{\beta}{\rho^2}Q^\intercal Q\right)^{-1}, \\ \boldsymbol{\mu}(u) &= \Sigma\Big(\frac{1}{\sigma^2}V^\intercal \boldsymbol{m}(u) + \frac{\beta}{\rho^2}Q^\intercal \boldsymbol{y}(u)\Big), \end{aligned}}$$

where β is a weight parameter. The weight parameter can be used to adjust the amount of supervision added on top of the conventional PPCA model. With $\beta = 0$, the model equals PPCA and when $\beta = 1$, we have the ordinary SPPCA.

4. Experimental Setup

4.1. VoxCeleb Speaker Recognition Corpus

We performed the speaker verification experiments on the recently published *VoxCeleb* dataset [22]. VoxCeleb contains over 150,000 real-world utterances from 1251 celebrities, of which 561 are females and 690 are males. A key difference to the widely used NIST corpora is that, on average, VoxCeleb has more than 100 utterances per speaker, typically obtained from multiple sessions with highly variable environments and recording conditions providing a large intra-speaker variability. The average utterance length is about 8 seconds. Although most of the utterances are short, the utterance length varies considerably, the longest ones being longer than one minute. Utterances have a sampling rate of 16 kHz.

The dataset was collected using fully automated pipeline that extracts and annotates utterances from YouTube videos. To ensure correct speaker annotation, the pipeline contains automatic face verification verifying that mouth movement in the video corresponds to the audio track and that the speaker's identity is the correct one. The utterances are mostly extracted from interview situations ranging from quiet studios to public speeches in front of large audiences. Differing environments and speaking styles are not the only source of variability, since differences in recording devices and audio processing practices are present in YouTube videos. As the acoustic and technical conditions of the utterances vary considerably, the dataset turns out to be challenging for an automatic speaker verification task as we will see in the experiments.

We used the same standard trial list as in the baseline system of [22]. It contains 40 speakers, whose name starts with the letter 'E'. The list has 37720 trials, half of them (18860) being same speaker trials, which differs substantially from the typical NIST setups with about 10 to 1 ratio between non-target and target trials.

The rest of the speakers were used for developing our speaker verification systems, *i.e*., to train the UBM, TVM and classifier back-end. To speed up experimentation, we utilized only one-fourth of the available training data as we did not see large decrease in system performance by decreasing the amount of training data. Our training set contains total of 37160 utterances from 1211 speakers.

We report recognition accuracy using *equal error rates* (EERs) that are the rates where *false alarm* and *miss* rates are equal. With the current trial list setup, 95% confidence intervals around EERs that are below 8% (the case with most experiments) are at widest $\pm 0.27\%$ absolute. Confidence intervals are computed using *z-test* based methodology presented in [24].

4.2. NIST Data

Even if our primary interest is in the VoxCeleb data, for the sake of reference, we also study different i-vector systems using common condition 5 of NIST 2010 Speaker Recognition Evaluation (SRE10)[1]. Trial segments in condition 5 contain conversational 8 kHz telephone speech spoken with normal vocal effort. The trial list consists of 30373 trials, of which 708 are same speaker trials.

Speaker verification systems were trained using 43308 utterances obtained from SRE04, SRE05, SRE06, Switchboard, and Fisher corpora.

The performances are reported as EERs and *minimum values* of *detection cost function* (minDCF) used in SRE10. The SRE10 detection cost function is given as

$$\text{DCF} = 0.001\, P_{\text{miss}} + 0.999\, P_{\text{fa}},$$

where P_{miss} and P_{fa} are probabilities of miss and false alarm, respectively [25].

4.3. Description of Speaker Verification System

We extracted 38 dimensional acoustic feature vectors containing 19 *Mel-frequency cepstral coefficients* (MFCCs) and their delta coefficients. After discarding features of non-speech segments, we subjected features to utterance-level mean and variance normalization.

The *universal background model* (UBM), 1024 component *Gaussian mixture model* (GMM) with diagonal covariance matrices, was trained using the development data. UBM was used to extract sufficient statistics, which were used in FEFA or in supervector extraction. Supervectors were extracted by first creating utterance specific GMMs using *maximum a posteriori* (MAP) adaptation [3] and then by concatenating mean vectors of adapted GMMs into supervectors.

We extracted 400 dimensional i-vectors, which were then centered, length-normalized, and whitened. Finally, we used simplified *probabilistic linear discriminant analysis* (PLDA) [26] to perform supervised dimensionality reduction of i-vectors into 200 dimensions and to score verification trials.

5. Speaker Verification Experiments

The results presented in Sections 5.1 to 5.6 are given for the VoxCeleb ASV protocol. Section 5.7 presents results for SRE10.

[1]https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2010

5.1. Relevance Factor in MAP Adaptation

We studied how the choice of relevance factor affects speaker verification performance with PPCA and FA methods. The results for VoxCeleb protocol are presented in Figure 1. Based on the results, we fix the relevance factor to $r = 1$ for the remaining experiments with this data. The choice of relevance factor is data-dependent, and therefore, the same value might not work well with other datasets.

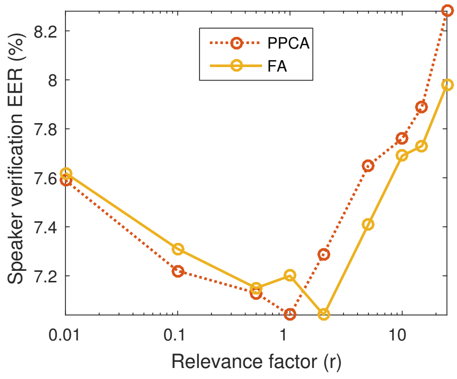


Figure 1: *The effect of relevance factor used in MAP adaptation on speaker verification performance. For VoxCeleb data, relevance factor close to 1 leads to the best results.*

5.2. Number of Training Iterations

To find out the sufficient number of iterations in TVM training, we evaluated verification performances with varying number of iterations. The results of the experiment, presented in Figure 2, reveal that 5 iterations are enough to obtain near to optimal performance. Hence, we fix the number of iterations to 5 for the remaining experiments.

All the methods, except SPPCA, behave similarly. With SPPCA, the training does not proceed in a desirable way during the first 5 iterations.

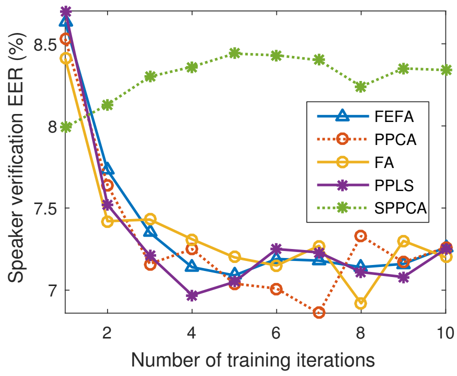


Figure 2: *Speaker verification performances with different numbers of training iterations in total variability model training. About 5 iterations are enough to obtain satisfying performance. The trend is similar for all the methods except for SPPCA, for which the iterative training does not improve the results.*

5.3. Maximization Principles in System Training

In Section 3.2, we presented two different maximization principles that can be used with all the discussed iterative TVM training methods. The comparison of the maximization principles in terms of resulting speaker verification performances is presented in Table 1. There are no clear differences between the principles.

Table 1: *Speaker verification equal error rates (%) for different methods and maximization principles used in TVM training. With conventional PCA approach we obtained EER of 7.39%.*

	max1 [Eq. (5)]	max2 [Eq. (6)]
FEFA	7.09	7.11
PPCA	**7.04**	7.18
FA	7.20	7.26
PPLS	7.05	7.42
SPPCA	8.44	8.26

5.4. Training Times

Figure 3 shows the elapsed times for 5 TVM training iterations with different methods. The measured times do not include the times needed to compute sufficient statistics or supervectors or to load them into memory using I/O operations. Measurements were conducted by running MATLAB implementations of all the methods in a 16-core 2.60 GHz machine with an ample amount of RAM (>300GB). To obtain reasonable training time with FEFA, we trained the system with 8 CPU cores, whereas other methods were trained using a single core.

Before the TVM training phase, different methods have only small differences in terms of computational requirements. Even though FEFA differs from other methods in that it uses sufficient statistics as inputs, the difference is minuscule, as most of the time in the extraction of MAP adapted supervectors is spent in the computation of sufficient statistics.

From the perspective of system optimization, note that FEFA does not require optimization of the relevance factor. But, the extra cost of relevance factor optimization in PPCA-PLDA system does not outweigh the training time of FEFA, as MAP adaptation using precomputed sufficient statistics and training of PPCA and PLDA are much less expensive operations than FEFA training.

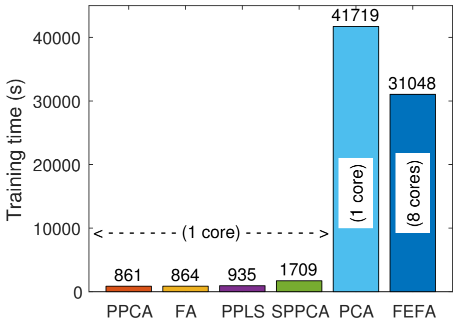


Figure 3: *Training times of TVMs with 5 iterations (PCA is non-iterative). Iterative supervector compression methods are the fastest to train, while FEFA requires the most amount of computation. FEFA was trained with 8 CPU cores to reduce the training time.*

5.5. Weight Parameter in Supervised Approaches

Next, we apply weighting to the supervised methods, PPLS and SPPCA, as discussed in Section 3.4. The results obtained with different weight parameter values are presented in Figure 4. We find that weighting does not affect PPLS and that the weighted SPPCA model functions better when it approaches PPCA ($\beta \to 0$). This suggests that the studied supervised methods do not provide any noticeable benefits over the unsupervised i-vector extractors.

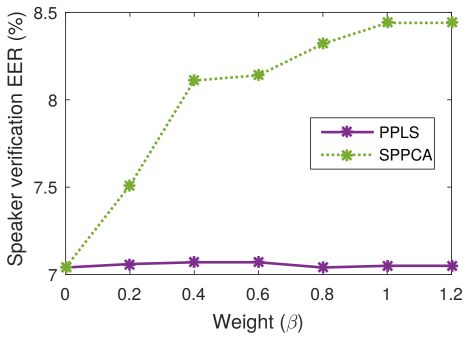


Figure 4: *Effect of the weight parameter (β) value in the supervised models. When $\beta = 0$, both models equal to the conventional PPCA. Adding supervision by increasing β does not improve the speaker verification performance.*

5.6. Dimensionality of I-Vectors

To improve the speaker verification performance, we jointly optimized dimensions of i-vectors and their PLDA-compressed versions. We varied the i-vector dimensionality between 200 and 1000 and the PLDA subspace dimensionality between 100 and 500. The results for all combinations using PPCA method are presented in Figure 5. The results indicate that, our initial parameters, 400 and 200 for i-vectors and PLDA, respectively, give a relatively good performance. We also see that a slight increase in performance might be obtained by using i-vector dimensions between 600 and 1000 with 350 to 400 dimensional PLDA subspaces.

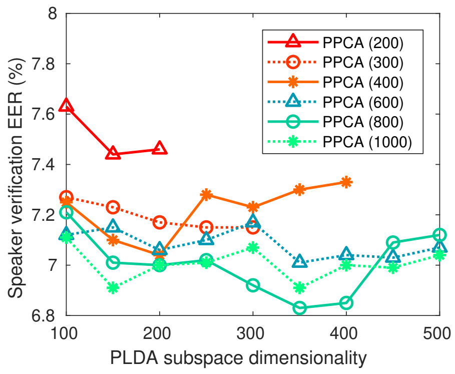


Figure 5: *Optimization of i-vector and PLDA subspace dimensions for PPCA method. Different lines represent different i-vector dimensionalities. The lowest error rate is obtained with 800-dimensional i-vectors and 350-dimensional PLDA space.*

5.7. Evaluation With NIST SRE10 Data

To increase our confidence of the generality of the results, we tested different i-vector systems on common condition 5 of SRE10. We ran the second protocol using mostly the same system configuration as with VoxCeleb corpus. We changed the filterbank spacing in MFCCs to match the 8 kHz sampling rate and we also increased the relevance factor to 6. Otherwise, the system was not optimized for the new data.

The results for SRE10, shown in Table 2, indicate that there are no clear differences between the two maximization principles. Further, we observe that the results for EER and minDCF are somewhat different as FEFA performs the best in terms of EER, while FA obtains the lowest minDCF. To gain better understanding on the performances of the systems, in Figure 6, we present *detection error trade-off* (DET) curves for all the methods using the maximization principle 1. The DET curves reveal that there are no clear differences between FEFA and FA.

6. Discussion and Conclusions

The development and optimization of i-vector systems tends to be time consuming. In particular, any change in the acoustic front-end or the UBM configuration requires retraining the TVM. If TVM training is slow, the parameter optimization can become unfeasible, possibly leading to suboptimal system configuration. In this work, we studied fast GMM supervector compression methods to streamline ASV system development. By focusing on compression of MAP adapted supervectors, we managed to cut the system training time down to a fraction of traditional approach (FEFA). Our results indicate that the alternative approaches work as well as the standard FEFA in terms of recognition accuracy. The less-optimistic performance reported in [14] and [15] (for the PPCA system) could be due to absence of MAP adaptation: we found that increasing the relevance factor from 0 (no MAP adaptation) towards some higher (optimized) value results in considerably higher verification accuracy.

We did not find benefit with either of the studied supervised models, PPLS or SPPCA. We were not, therefore, able to reproduce positive findings claimed in [17] for PPLS. This might be due to differing datasets or feature configurations. On a positive side, we found that PPLS attains similar training speeds to PPCA and FA.

The proposed modification to the maximization principle in TVM training did not affect verification results negatively. This modification makes the theory and the system implementation slightly simpler as it avoids computing expectations over i-vector posterior distributions.

We recognize that the findings of the current study can not be generalized to all existing system configurations without further studies. In this study, we only experimented with a specific set of acoustic features together with a specific UBM and backend configurations (PLDA).

In summary, from the various compared variants, we recommend to use PPCA and FA to compress supervectors. Both are easy to implement on top of the GMM framework and lead to considerably faster TVM training times. For optimal verification accuracy, supervectors should be created using the MAP adaptation with an optimized relevance factor. We have made our MATLAB implementations of PPCA, FA, PPLS, and SPPCA available at `http://cs.uef.fi/~vvestman/research_codes/supervector_compression.zip`.

Table 2: *Speaker verification performances for different methods and maximization principles (max1, max2) on common condition 5 of SRE10. With conventional PCA we obtained EER of 4.69% and minDCF of 6.55%.*

	EER (%)		minDCF (%)	
	max1	max2	max1	max2
FEFA	4.24	**3.86**	6.24	6.85
PPCA	4.66	4.66	6.55	6.72
FA	4.24	4.24	6.00	**5.50**
PPLS	4.66	4.79	6.73	6.53
SPPCA	4.46	4.38	6.46	6.26

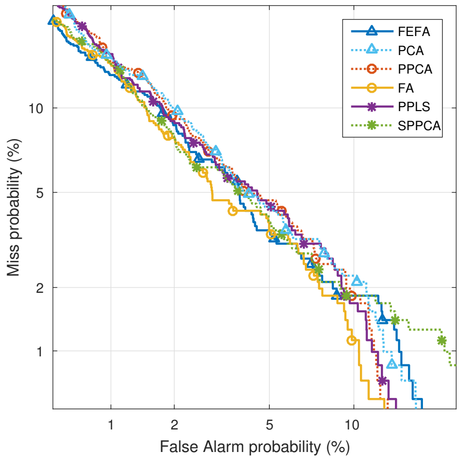


Figure 6: *Detection error trade-off curves for different methods using max. principle 1 on common condition 5 of SRE10.*

7. References

[1] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[2] Patrick Kenny, "A small footprint i-vector extractor," in *Odyssey*, 2012, vol. 2012, pp. 1–6.

[3] Douglas Reynolds, Thomas Quatieri, and Robert Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[4] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *ICASSP 2014*. IEEE, pp. 1695–1699.

[5] Ondřej Glembek, Lukáš Burget, Pavel Matějka, Martin Karafiát, and Patrick Kenny, "Simplification and optimization of i-vector extraction," in *ICASSP 2011*, pp. 4516–4519.

[6] Sandro Cumani and Pietro Laface, "Memory and computation trade-offs for efficient i-vector extraction," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 934–944, 2013.

[7] Ville Vestman, Dhananjaya Gowda, Md Sahidullah, Paavo Alku, and Tomi Kinnunen, "Time-varying autoregressions for speaker verification in reverberant conditions," in *Interspeech 2017*.

[8] Sandro Cumani and Pietro Laface, "Factorized sub-space estimation for fast and memory effective i-vector extraction," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 22, no. 1, pp. 248–259, 2014.

[9] Longting Xu, Kong Aik Lee, Haizhou Li, and Zhen Yang, "Generalizing i-vector estimation for rapid speaker recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2018.

[10] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[11] William Campbell, Douglas Sturim, and Douglas Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006.

[12] Sam Roweis, "EM algorithms for PCA and SPCA," in *Advances in neural information processing systems*, 1998, pp. 626–632.

[13] Michael Tipping and Christopher Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[14] Srikanth Madikeri, "A fast and scalable hybrid FA/PPCA-based framework for speaker recognition," *Digital Signal Processing*, vol. 32, pp. 137–145, 2014.

[15] Srikanth Madikeri, "A hybrid factor analysis and probabilistic pca-based system for dictionary learning and encoding for robust speaker recognition," in *Odyssey 2012*, pp. 14–20.

[16] Bavattichalil Haris and Rohit Sinha, "On exploring the similarity and fusion of i-vector and sparse representation based speaker verification systems," in *Odyssey 2012*, pp. 21–27.

[17] Chen Chen, Jiqing Han, and Yilin Pan, "Speaker verification via estimating total variability space using probabilistic partial least squares," in *Interspeech 2017*, pp. 1537–1541.

[18] Yun Lei and John Hansen, "Speaker recognition using supervised probabilistic principal component analysis," in *Interspeech 2010*, pp. 382–385.

[19] Shipeng Yu, Kai Yu, Volker Tresp, Hans-Peter Kriegel, and Mingrui Wu, "Supervised probabilistic principal component analysis," in *KDD '06*. ACM, 2006, pp. 464–473.

[20] Patrick Kenny, Gilles Boulianne, and Pierre Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE transactions on speech and audio processing*, vol. 13, no. 3, pp. 345–354, 2005.

[21] Christopher Bishop, "Machine learning and pattern recognition," *Springer*, 2006.

[22] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "VoxCeleb: A large-scale speaker identification dataset," in *Interspeech 2017*, pp. 2616–2620.

[23] Karl Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[24] Samy Bengio and Johnny Mariéthoz, "A statistical significance test for person authentication," in *Odyssey 2004: The Speaker and Language Recognition Workshop*, pp. 237–244.

[25] Alvin Martin and Craig Greenberg, "The NIST 2010 Speaker Recognition Evaluation," in *Interspeech 2010*, pp. 2726–2729.

[26] Daniel Garcia-Romero and Carol Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Interspeech 2011*.

A. Proof: I-Vector Posterior Distribution for PPCA Model

The following proof is similar in principle to the proof of Proposition 1 in [20].

It is enough to show that $p(\boldsymbol{w}(u)|\boldsymbol{m}(u)) \propto \mathcal{N}(\boldsymbol{\mu}(u), \Sigma)$, since $p(\boldsymbol{w}(u)|\boldsymbol{m}(u))$ is a probability density function and will hence be correctly scaled. For brevity, we drop u from the notation of the following chain of relations that proves the claim (*e.g.* $\boldsymbol{\mu}$ will refer to $\boldsymbol{\mu}(u)$):

$$
\begin{aligned}
&\mathcal{N}(\boldsymbol{\mu}, \Sigma) \\
&\propto \exp\Big(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu})^\intercal \Sigma^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\Big) \\
&= \exp\Big(-\frac{1}{2}\Big(\boldsymbol{w}-\frac{1}{\sigma^2}\Sigma V^\intercal \boldsymbol{m}\Big)^\intercal \Big(I+\frac{1}{\sigma^2}V^\intercal V\Big) \\
&\quad \Big(\boldsymbol{w}-\frac{1}{\sigma^2}\Sigma V^\intercal \boldsymbol{m}\Big)\Big) \\
&= \exp\Big(-\frac{1}{2}\Big[\Big(\boldsymbol{w}^\intercal+\frac{1}{\sigma^2}\boldsymbol{w}^\intercal V^\intercal V-\frac{1}{\sigma^2}(\Sigma V^\intercal \boldsymbol{m})^\intercal \Sigma^{-1}\Big) \\
&\quad \Big(\boldsymbol{w}-\frac{1}{\sigma^2}\Sigma V^\intercal \boldsymbol{m}\Big)\Big]\Big) \\
&= \exp\Big(-\frac{1}{2}\Big[\boldsymbol{w}^\intercal \boldsymbol{w}+\frac{1}{\sigma^2}\boldsymbol{w}^\intercal V^\intercal V \boldsymbol{w} \\
&\quad -\frac{1}{\sigma^2}\Big(\Sigma V^\intercal \boldsymbol{m}\Big)^\intercal \Sigma^{-1}\boldsymbol{w} \\
&\quad -\Big(\boldsymbol{w}^\intercal+\frac{1}{\sigma^2}\boldsymbol{w}^\intercal V^\intercal V\Big)\frac{1}{\sigma^2}\Sigma V^\intercal \boldsymbol{m}\Big]+\mathrm{const}(\boldsymbol{m})\Big) \\
&\propto \mathcal{N}(\boldsymbol{0}, I)\exp\Big(-\frac{1}{2\sigma^2}\Big[(V\boldsymbol{w})^\intercal V\boldsymbol{w}-\boldsymbol{m}^\intercal V\Sigma^\intercal \Sigma^{-1}\boldsymbol{w} \\
&\quad -\boldsymbol{w}^\intercal \Sigma^{-1}\Sigma V^\intercal \boldsymbol{m}\Big]\Big) \\
&\propto \mathcal{N}(\boldsymbol{0}, I)\exp\Big(-\frac{1}{2\sigma^2}\Big[(V\boldsymbol{w})^\intercal V\boldsymbol{w}-\boldsymbol{m}^\intercal V\boldsymbol{w} \\
&\quad -(V\boldsymbol{w})^\intercal \boldsymbol{m}+\boldsymbol{m}^\intercal \boldsymbol{m}\Big]\Big) \\
&= \mathcal{N}(\boldsymbol{0}, I)\exp\Big(-\frac{1}{2}(\boldsymbol{m}-V\boldsymbol{w})^\intercal \frac{1}{\sigma^2}(\boldsymbol{m}-V\boldsymbol{w})\Big) \\
&\propto \frac{p(\boldsymbol{w})p(\boldsymbol{m}|\boldsymbol{w})}{p(\boldsymbol{m})} \\
&= p(\boldsymbol{w}|\boldsymbol{m}).
\end{aligned}
$$

Note that in the above chain of proportional relations, we can drop $(\exp(\mathrm{const}(\boldsymbol{m})))$ and add $(\exp(\boldsymbol{m}^\intercal \boldsymbol{m})$; $p(\boldsymbol{m}))$ multipliers that only depend on $\boldsymbol{m}$ without breaking the chain.

Paper **IV**

WHO DO I SOUND LIKE?
SHOWCASING SPEAKER RECOGNITION TECHNOLOGY BY YOUTUBE VOICE SEARCH

Ville Vestman, Bilal Soomro, Anssi Kanervisto, Ville Hautamäki, Tomi Kinnunen

School of Computing, University of Eastern Finland

ABSTRACT

The popularization of science can often be disregarded by scientists as it may be challenging to put highly sophisticated research into words that general public can understand. This work aims to help presenting speaker recognition research to public by proposing a publicly appealing concept for showcasing recognition systems. We leverage data from YouTube and use it in a large-scale voice search web application that finds the celebrity voices that best match to the user's voice. The concept was tested in a public event as well as "in the wild" and the received feedback was mostly positive. The i-vector based speaker identification back end was found to be fast (665 ms per request) and had a high identification accuracy (93%) for the YouTube target speakers. To help other researchers to develop the idea further, we share the source codes of the web platform used for the demo at `https://github.com/bilalsoomro/speech-demo-platform`.

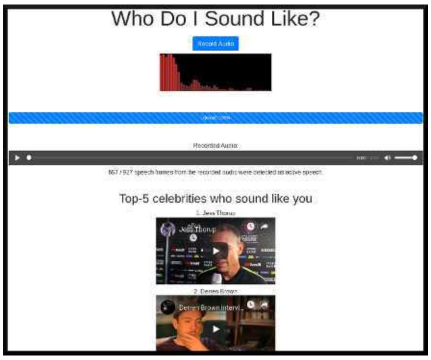
Index Terms— Large-scale speaker identification, speaker ranking, public demo, VoxCeleb, web service

1. INTRODUCTION

As methodology researchers, we often find it challenging to explain intuitively where and how our research advancements in speaker recognition can be used. To demonstrate speaker recognition technology in an appealing way to the public, many challenges need to be resolved. Besides the standard challenges of speaker recognition technology such as background noise [1], channel mismatch [2], and the requirement of fast response times in large-scale recognition tasks [3], there are challenges related to the demo design itself. First, the traditional speaker recognition setting requires at least two separate speech inputs from the user, one is for enrollment and the other one for test. The requirement of two separate recordings can be inconvenient for an user who wants to quickly test the system. The second challenge in showcasing is how to give an attractive feedback to the user. This could be implemented as a real-life application, for example, by using user's voice to open a physical lock, or in a less involved way by displaying recognition scores in a screen [4].

In this work, we present a concept for creating publicly appealing demos to showcase speaker recognition technology by leveraging public-domain target speaker data collected from YouTube. The core idea is to compare users' speech to the ones of celebrities on YouTube, who have been enrolled prior to the real-time demonstration. The results of the comparison are then displayed as a selection of YouTube videos from the best matching celebrities, which allows users to see and listen to the celebrity speakers who they most resemble to (Figure 1). Even if we focus on speaker recognition research, the same concept could also be applied for other things that can be inferred or estimated from speech such as age, emotion, or language.

This research was partially funded by the Academy of Finland (grants #313970 and #309629).



667 / 927 speech frames from the recorded audio were detected as active speech.

Fig. 1. A screenshot from our voice search web application displaying the basic elements of the UI: Recording button, audio visualization, playback option for the recorded speech, and the results.

For example, if the user records angry voice, the results could show YouTube videos of angry people. When the results include famous public figures, the user's interest and satisfaction of the demo tends to naturally rise. We saw this positive effect while presenting the demo in a locally organized sub-event of an European-wide "The European Researchers' Night 2018" [1] that aims to bring scientific research to public.

We run our demo on a web platform that can be used on PCs and mobile phones with an internet connection to ensure good accessibility of the demo. The web platform communicates with a computation server that runs the speaker recognition back end based on our recent work on computationally efficient i-vector extraction [5]. The back end provides the results to the web platform that displays them by using embedded YouTube video players.

Our extensive use of YouTube data has been made possible by recent automated speech data collection efforts in [6] and [7] resulting in VoxCeleb1 and VoxCeleb2 corpora, respectively. These corpora provide a large set of annotated YouTube speech data including metadata for obtaining web links to the original YouTube videos.

To best of our knowledge, prior existing speaker recognition demos have not utilized VoxCeleb data in the proposed way. We are aware of a website [2] with a similar idea, but unfortunately we have not been able to successfully run the demo to see how it functions. Based on the celebrity speaker names, that demo does not utilize VoxCeleb data, and likely does not display YouTube videos in the results.

In summary, the current work describes a novel concept that al-

[1] `https://ec.europa.eu/research/mariecurieactions/actions/european-researchers-night`
[2] `https://celebsoundalike.com`

lows speech technology research teams to demonstrate their research without requiring large amount of additional work. To help other researchers to apply the concept for their own research, we share the source code of our web platform allowing a quick start for prototyping possible demo applications. We tested the concept among the public using our voice comparison demo utilizing standard speaker recognition techniques, and the received feedback from the people was mainly positive.

2. WEB PLATFORM FOR SPEECH DEMOS

We designed the web platform with the sole purpose of demonstrating speech processing systems to the public, and in this work we used it to demonstrate speaker recognition using YouTube data. The platform is implemented as a web service in PHP and JavaScript, supporting different browser and devices. Users can select one of speech processing "methods" defined by the host of the platform. The methods could, for example, perform speaker recognition or age estimation from the recorded speech.

The back end of the platform is implemented in PHP, and thus only needs a web server capable of running PHP (*e.g.* Apache or Nginx). For simplicity and reliability, server-side code only receives WAV audio files from the clients and runs a specified method as a `system()` call, and finally returns the results to the client. For privacy reasons, the audio file is not stored on the server, and is immediately removed at the end of handling user's request. The platform supports including additional user inputs required for the analysis, *e.g.* the claimed identity for speaker verification demo.

The front end of the platform is implemented in JavaScript, which also handles recording of the audio in raw format at 16kHz. Features required by this code are supported by the most PCs and Android phones, making it easier to share the demo with others. The user interface records a sample of user's speech, queries what speech processing method should be applied on the recorded sample, sends the sample to server to be processed, and displays the results.

We share the source code of the platform in the hopes it will support other researchers in speech analysis to demonstrate their work to the public. The code includes instructions how to setup the server in couple of steps. New speech analysis systems for demonstration can be added by modifying a single JSON file.

3. SPEAKER RECOGNITION BACK END

The system comparing user's voice to voices in YouTube videos can be regarded as a *closed set speaker identification* system. As we only utilize a closed set of YouTube target speakers, we can include the data from the target speakers in the system development. In this section, we describe the data sets and the speaker identification system used for providing functionality to the web front end.

3.1. YouTube data: VoxCeleb1 & VoxCeleb2

The audio-visual VoxCeleb corpora [6, 7] have been adopted in many application areas including speaker recognition [6, 7], speech separation [8], and emotion recognition [9] to name a few. The VoxCeleb data has been automatically collected from YouTube by exploiting face verification and active speaker detection systems. An automated pipeline enabled collecting very large scale speaker recognition data sets: When combined, the VoxCeleb corpora consist of almost 1.3 million speech clips from over 170,000 YouTube videos from more than 7000 speakers and, in total, nearly 3000 hours of speech material. The average length of speech clips in VoxCeleb is about eight seconds.

The metadata provided with the VoxCeleb corpora includes, for example, speakers' names, IDs of the original YouTube videos, and the starting and ending times of the clips within the videos expressed as frames. This metadata is enough for setting up a demo where users can find best matching voices to theirs from YouTube. Although the metadata is automatically obtained, it is, in our experience, fairly accurate. Regarding to the correctness of the labels, the authors of VoxCeleb mention that the VoxCeleb2 corpus is mainly intended to be used as a training data set and that during the data collection thresholds for discarding false positives were not as strictly set as with VoxCeleb1 data collection [7]. We have witnessed a few labeling errors in VoxCeleb2, such as Finnish president Tarja Halonen being confused to talk-show host Conan O'Brien. However, the errors do not exist to an extent that would be a considerable problem for our application.

3.2. Speaker identification system description

The acoustic feature vectors of the speaker identification system consist of 20 MFCCs plus their delta and double-delta coefficients. The system discards non-speech frames using a energy based speech activity detector and normalizes obtained features to have zero mean and unit variance.

For training the system components and enrolling the celebrities, we used those speakers from VoxCeleb corpora who had more than five utterances of length of five seconds or more. There are 903,498 such utterances and 7,363 such speakers. In the training of some system components, only a fraction of this data was needed to reach close to optimal recognition accuracy. We trained an universal background model (UBM) using one-thirtieth of the selected 903,498 utterances. The UBM is a 1024-component Gaussian mixture model (GMM) [10], which is used to compute sufficient statistics for i-vector extraction. We compute 800-dimensional i-vectors by compressing mean supervectors of maximum a posteriori (MAP) adapted GMMs using probabilistic principal component analysis (PPCA) as described in [5]. This is a (speed-wise) high-performing alternative to the standard i-vector extraction that is traditionally done via front-end factor analysis [11, 12]. We trained the PPCA model using one-fifteenth of the selected data.

Prior to scoring, i-vectors are centered using the mean computed from the whole training data of 903,498 utterances and then normalized to unit length. Scoring is performed with a simplified Gaussian probabilistic linear discriminant analysis (G-PLDA) model [13], which has a 350-dimensional speaker subspace. The G-PLDA model was trained using the whole training data.

At the online stage, the i-vector extracted from user's recording is scored against all of the 903,498 i-vectors used in PLDA training. The speakers are sorted according to the scores of their highest scoring utterances, from highest score to lowest. Finally, the system sends the names of the top-5 speakers together with the links to the YouTube-videos that correspond to the highest scoring utterances to the client.

3.3. System runtime considerations at online stage

To ensure fast response times, we implemented the speaker recognition back end as a server that has all the necessary models preloaded in the memory. The server is implemented with Python using scientific computing libraries available to it (*e.g.* NumPy and SciPy). We pay special attention to the PLDA scoring and i-vector extraction as they are the most time consuming steps during the computation.

In [13], it is shown that the score for a trial using G-PLDA can be computed as

$$\text{score} = \tilde{\boldsymbol{\eta}}_1^{\intercal} \widetilde{Q} \tilde{\boldsymbol{\eta}}_1 + \tilde{\boldsymbol{\eta}}_2^{\intercal} \widetilde{Q} \tilde{\boldsymbol{\eta}}_2 + 2\tilde{\boldsymbol{\eta}}_1^{\intercal} \Lambda \tilde{\boldsymbol{\eta}}_2 + \text{const},$$

Table 1. Speaker rank testing for six public figures using 10 audio clips from each speaker. The speaker ranks range from 1 to 5 and 'x' is shown if the result list of top-5 speakers did not contain the correct speaker at all. The tests are performed with and without a replay channel. The replay experiment does not require direct access to the back end system, but can be done by using the web demo only.

	Without replay channel				With replay channel			
		Occurrences in				Occurrences in		
Speaker's name	list positions for 10 clips	top1	top3	top5	list positions for 10 clips	top1	top3	top5
Hillary Clinton	`1111111111`	10	10	10	`1111111111`	10	10	10
Ariana Grande	`1111111111`	10	10	10	`3111111111`	9	10	10
Oprah Winfrey	`1111111111`	10	10	10	`1311111112`	8	10	10
Johnny Depp	`1111112112`	8	10	10	`1111x11121`	8	9	9
Bruno Mars	`1411111112`	8	9	10	`1x21111211`	7	9	9
Conan O'Brien	`1111111111`	10	10	10	`1111111111`	10	10	10
Total (in % of max.)		93	98	100		87	97	97

where $\tilde{\boldsymbol{\eta}}_1$ and $\tilde{\boldsymbol{\eta}}_2$ are lower dimensional projections of enrollment and test i-vectors, respectively, and where $\tilde{\boldsymbol{\eta}}_1^{\intercal}\widetilde{Q}\tilde{\boldsymbol{\eta}}_1$ and $\tilde{\boldsymbol{\eta}}_1^{\intercal}\Lambda$ can be precomputed.

As we work with an identification system (one test segment vs. all enrollment segments), the second term $\tilde{\boldsymbol{\eta}}_2^{\intercal}\widetilde{Q}\tilde{\boldsymbol{\eta}}_2$ is a constant and thus can be neglected. Therefore, to get all the $n = 903{,}498$ scores at online stage, we only need to compute

$$\text{scores} = \boldsymbol{\nu} + 2DP\boldsymbol{\eta}_2,$$

where $\boldsymbol{\nu}$ is an n-dimensional vector containing precomputed values $\tilde{\boldsymbol{\eta}}_1^{\intercal}\widetilde{Q}\tilde{\boldsymbol{\eta}}_1$, matrix $D \in \mathbb{R}^{n\times 350}$ contains precomputed vectors $\tilde{\boldsymbol{\eta}}_1^{\intercal}\Lambda$, and P is a 350×800 projection matrix that projects test i-vector $\boldsymbol{\eta}_2$ to a lower dimensional space so that $\tilde{\boldsymbol{\eta}}_2 = P\boldsymbol{\eta}_2$. The product $D\tilde{\boldsymbol{\eta}}_2$ can be efficiently parallelized.

The i-vector extraction using PPCA is simply a matter of compressing 61440-dimensional GMM-supervector to 800-dimensional space using a precomputed projection matrix. Note that the traditional approach for i-vector extraction would, in addition, require inverting an 800×800 posterior covariance matrix [14, 5].

4. SYSTEM EVALUATION

We tested our voice search demo and the underlying speaker recognition back end in multiple ways using both objective and subjective measures in evaluation. On the objective side, we computed an equal error rate (EER) using VoxCeleb speaker verification protocol and further we tested the rankings that the system displays for newly downloaded and replayed YouTube data. On the subjective side, we gathered feedback from the users of the system, including their opinions on how close the displayed top five celebrities sound to the user.

4.1. Evaluation using VoxCeleb speaker verification protocol

The VoxCeleb1 speaker verification test protocol includes 37720 trials with a balanced number of same speaker trials and impostor trials. The trial list has been formed using 4715 utterances from 40 speakers. Using this protocol, we obtained EER of 6.69 %. This result is better than the baseline result for i-vectors in [7], but should not be directly compared as our system utilizes testing utterances also in system training.

4.2. Speaker rank testing on non-VoxCeleb YouTube data

To test the the final deployed demo, we studied the speaker rankings the system outputs. For this purpose, we collected a small set of new YouTube data. This set contains 10 new speech clips for six public figures in VoxCeleb corpora. The clips are about 15 seconds long each and they are extracted from videos that are not already present in VoxCeleb corpora. When the new clips are fed to the speaker recognition back end, the output lists of top-5 speakers should contain the correct speaker as they are present in VoxCeleb and hence are already enrolled to the system.

The new test data was used with the system in two ways: First, we downloaded the speech clips from YouTube and fed the data directly to the speaker recognition back end. Secondly, we played files directly from YouTube and at the same time recorded them with the web demo. Unlike the first approach, the second one includes the channel effects caused by replaying the data. In the replay experiment, the playback device was Sony SRS-XB10 portable Bluetooth speaker while the web demo was ran in Chrome browser in Nokia 8 smartphone running Android 8.1.0. The distance between the two devices was kept to 5 cm as the recording device was held by hand above the up facing speaker. The room in which the experiment took place was quiet and the only background noise that was present was the fan noise of the laptop which was connected to the speaker.

For both settings, with and without replay, the speaker rankings for all the test utterances are shown in Table 1. In addition, the table contains statistics of the number of occurrences in the top-1, top-3, and top-5 rankings. Without the replay, the system was always able to include the correct speaker to the top-5 list and 93% of the times the speaker was identified correctly (*i.e.*, in top-1). Replaying the audio clips decreased the system performance only slightly as the correct speaker was left outside the top-5 list only twice out of the 60 trials.

To get insight of how long of an utterance is required for getting good results in our celebrity matching demo, we studied the effect of length of the test utterance on system accuracy. We ran the previous experiment without the replay effect using utterances clipped to lengths ranging from 1 second to 15 seconds. We found that the test segment needs to be at least 9 seconds to obtain close to optimal performance and at least 5 seconds to obtain identification accuracies greater than 70% (Figure 2).

4.3. Feedback and impressions from public testing

The first public test for our voice search demo took place in the event "The European Researchers' Night 2018" (September 28, 2018), where researcher's from many fields were displaying their research to the public. The event was funded by EU and it was organized in many countries across the Europe. In our local event, we were

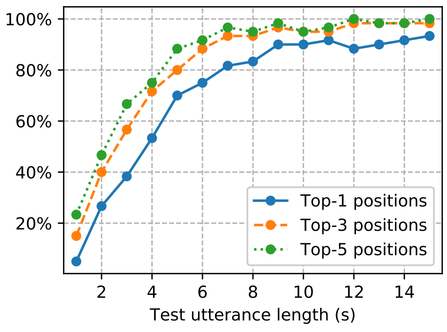


Fig. 2. The effect of utterance length on speaker ranking performance. Specifically, the graph shows how often the target speakers are displayed in the top-lists when tested with different lengths of test utterances from the target. An utterance of length 9s is required to reach close to optimal performance.

Table 2. Computation times for the different steps in the voice comparison pipeline. The steps marked with * are parallelized to 16 CPU cores while others steps utilize only 1 core. The total response time is the time it takes to upload the speech and compute and display the results. The data was collected from 402 requests, except for the total response time which was collected together with the feedback questionnaire (n=27).

	Times in milliseconds (ms)		
	median	mean	SD
Audio loading, MFCC extraction	47.1	86.2	142.1
Sufficient statistics computation	20.9	46.3	98.1
MAP adaptation	0.8	0.9	0.6
Supervector compression (PPCA)*	42.6	56.5	28.3
I-vector centering & length norm.	0.1	0.1	< 0.1
I-vector compression (PLDA)	0.4	0.5	0.3
PLDA scoring*	336.4	423.8	195.8
Sorting speakers	39.6	44.6	13.8
Total time in computing server	521.5	661.0	331.6
Total response time	1791.1	2503.5	1975.1

showcasing our demo for five hours and for the most of the time there was a long queue of people waiting for their turn to test our demo. In total, approximately 150 people tried the demo. The feedback was mostly positive, although not everyone was satisfied with their results. As the event was targeted for families, many of the testers were children. This was a slight problem as only a small minority of the speakers in VoxCeleb corpora are children, causing it to be difficult to find a good voice match for everyone.

In the event, we were using our own high-quality microphone (Zoom H6 Handy Recorder, XY mic) and a laptop that was well tested with the demo. To see how the demo works "in the wild", we shared a web link to our demo in a multiple social media platforms. The shared demo application was equipped with a short feedback questionnaire for subjective evaluation. We also collected error reports containing system information of the devices on which the demo did not work.

The public testing revealed that the device and browser support is still quite limited due to some issues with the audio recording and

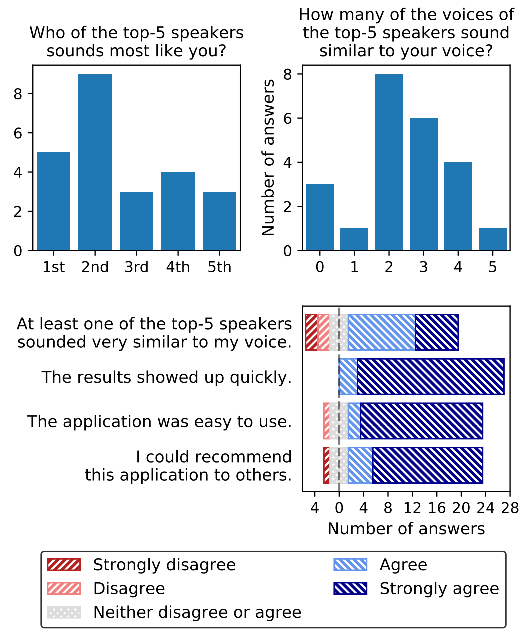


Fig. 3. Results from the feedback questionnaire, gathered from users using platform that finds matches for their speech from a set of over 7000 celebrities. Based on these subjective assessments, the system is able to find good matches for users' speech in most cases.

playback support. Based on the feedback, we estimate that demo ran on 50 to 70 percent of the device-browser configurations that our test users were using. We also got some good suggestions how to improve the user interface and we believe that together with improved browser support the user experience can be very good as the received answers (n=27) to the questionnaire were already fairly positive as can be seen from Figure 3.

4.4. Response and computation times

During the test in the wild, we collected computation times of the different steps in the voice comparison. The statistics are summarized in Table 2. The average time to compute one voice comparison request was 661 milliseconds, which means that our computation server could, theoretically, respond to 5000 requests in an hour without processing multiple requests in parallel. The total response time, on average, was about 2.5 seconds. As seen from Figure 3, this level of responding speed was considered to be fast.

5. CONCLUSIONS

We successfully capitalized the appeal to public figures with our YouTube voice search demo application. The objective and the subjective evaluations of the demo showed that the platform was mostly successful in providing good results and also being convenient to use. The feedback received from the users allows us to further develop our demo platform, which we have shared for open source development at https://github.com/bilalsoomro/speech-demo-platform. We would be happy to see the proposed concept to be applied in the future with other speech related recognition systems as well.

6. REFERENCES

[1] Xiaojia Zhao, Yuxuan Wang, and DeLiang Wang, "Robust speaker identification in noisy and reverberant conditions," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 4, pp. 836–845, 2014.

[2] A. Solomonoff, W. M. Campbell, and I. Boardman, "Advances in channel compensation for SVM speaker recognition," in *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, March 2005, vol. 1, pp. I/629–I/632 Vol. 1.

[3] L. Schmidt, M. Sharifi, and I. L. Moreno, "Large-scale speaker identification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1650–1654.

[4] Kong Aik Lee, Anthony Larcher, Helen Thai, Bin Ma, and Haizhou Li, "Joint application of speech and speaker recognition for automation and security in smart home," in *Proc. Interspeech*, 2011.

[5] Ville Vestman and Tomi Kinnunen, "Supervector compression strategies to speed up i-vector system development," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 357–364.

[6] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "VoxCeleb: A large-scale speaker identification dataset," in *Proc. Interspeech*, 2017, pp. 2616–2620.

[7] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. Interspeech*, 2018, pp. 1086–1090.

[8] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman, "The conversation: Deep audio-visual speech enhancement," in *Proc. Interspeech*, 2018, pp. 3244–3248.

[9] S. Albanie, A. Nagrani, A. Vedaldi, and A. Zisserman, "Emotion recognition in speech using cross-modal transfer in the wild," in *ACM Multimedia*, 2018.

[10] Douglas Reynolds, Thomas Quatieri, and Robert Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[11] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[12] Patrick Kenny, "A small footprint i-vector extractor," in *Odyssey*, 2012, vol. 2012, pp. 1–6.

[13] Daniel Garcia-Romero and Carol Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech*, 2011, pp. 249–252.

[14] Srikanth Madikeri, "A fast and scalable hybrid FA/PPCA-based framework for speaker recognition," *Digital Signal Processing*, vol. 32, pp. 137–145, 2014.

Paper **V**

INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Unleashing the Unused Potential of I-Vectors Enabled by GPU Acceleration

Ville Vestman[1,2], *Kong Aik Lee*[1], *Tomi H. Kinnunen*[2], *Takafumi Koshinaka*[1]

[1]Biometrics Research Laboratories, NEC Corporation, Japan
[2]Computational Speech Group, University of Eastern Finland, Finland

vvestman@cs.uef.fi, k-lee@ax.jp.nec.com, tkinnu@cs.uef.fi, koshinak@ap.jp.nec.com

Abstract

Speaker embeddings are continuous-value vector representations that allow easy comparison between voices of speakers with simple geometric operations. Among others, i-vector and x-vector have emerged as the mainstream methods for speaker embedding. In this paper, we illustrate the use of modern computation platform to harness the benefit of GPU acceleration for i-vector extraction. In particular, we achieve an acceleration of 3000 times in frame posterior computation compared to real time and 25 times in training the i-vector extractor compared to the CPU baseline from Kaldi toolkit. This significant speed-up allows the exploration of ideas that were hitherto impossible. In particular, we show that it is beneficial to update the universal background model (UBM) and re-compute frame alignments while training the i-vector extractor. Additionally, we are able to study different variations of i-vector extractors more rigorously than before. In this process, we reveal some undocumented details of Kaldi's i-vector extractor and show that it outperforms the standard formulation by a margin of 1 to 2% when tested with VoxCeleb speaker verification protocol. All of our findings are asserted by ensemble averaging the results from multiple runs with random start.

Index Terms: speaker recognition, PyTorch, factor analysis, total variability model

1. Introduction

A decade ago, the *i-vector* speaker embedding was introduced [1]. Since its introduction, it has remained as a standard solution for speaker recognition until recent years when it was excelled in many tasks by the deep neural network based embeddings [2, 3]. The recent developments are a result of the widespread interest among researchers to adopt deep learning techniques in their research. The most recent rise of deep learning has been partially made possible by the year-by-year increasing computation resources [4], and especially the use of *graphics processing units* (GPUs) to harness the benefits of massive parallelism even with consumer level devices.

While GPUs are heavily adopted in deep learning, they can also be conveniently utilized for the traditional learning of generative models such as the *total variability model* [5] underlying i-vector extraction. So far, this has been a largely unexploited possibility despite the fact that full-fledged i-vector extractors tend to be slow to train. The slowness of training has often forced many researchers to limit their experimental validation, for example by limiting the number of training iterations, or by relaying on the results from a single run with random initialization. In addition, simplifications and approximations of the model have been proposed to reduce the computational load [6, 7, 8].

For the current work, we utilize GPU to accelerate i-vector extraction and the total variability model training to alleviate the above limitations. The obtained speed-up allows us to study i-vector extractors in a more detailed manner than what has been possible previously. For example, we can train i-vector extractors *without* any approximations for hundreds of iterations to study the optimal number of iterations to maximize the speaker recognition performance. In addition, we are able to obtain more reliable comparisons between different variations of extractors by averaging the results of multiple runs with different random initializations of the model. For instance, the extractor training can differ in terms of whether model parameters are *re-estimated* using *minimum divergence criterion* [9] and whether the residual covariance matrix of the model is updated.

Further, we re-explore the idea of updating frame alignments during the training of i-vector extractor, which could potentially enhance the model fit and the resulting speaker recognition performance. The idea of updating the alignments was originally presented in the context of *eigenvoice* modeling for automatic speech recognition [10], but has received limited attention in the context of i-vectors for speaker recognition. In eigenvoice modeling, the alignment update is performed using speaker-dependent supervectors, which is not suitable approach for speaker recognition as it would tend to model out the speaker information from the i-vectors. Instead, we update the global UBM mean supervector to realign the training data.

In the experiments, we extensively utilize our GPU re-implementation of Kaldi speech recognition toolkit's [11] i-vector extractor. The implementation in Kaldi has some special traits, which, to the best of our knowledge, have not been extensively documented. Most notably, in Kaldi's implementation, the bias term is augmented to the *total variability matrix* [5], which causes some changes to the minimum divergence re-estimation step and which also eliminates the need of centralizing Baum-Welch statistics [12]. As Kaldi is one of the most popular tools used for the speaker recognition research, we consider it worthwhile to document the main differences of the two formulations in the following sections.

2. I-vector speaker embeddings

We compare two different formulations of the *total variability* approach [5] of *joint factor analysis* [13] to extract i-vectors. In the total variability model, all of the variability in utterances is modeled using a single subspace only, without having separate subspaces to model speaker and channel effects.

The first of the formulations is the original formulation [10, 14], which is commonly adopted in many available speaker recognition toolkits [15, 16, 17]. The second formulation, implemented in the Kaldi speech recognition toolkit, is inspired by the *subspace Gaussian mixture model* [18]. This formulation differs from the standard one as it augments the bias term of the model to the *factor loading matrix*, which allows estimating the bias term and the factor loading matrix jointly.

Common to both formulations is the use of *Baum-Welch statistics* as defined in [14]. In this work, we denote the occupancy statistics, first order statistics, and the second order statistics for the Gaussian component c ($c = 1, 2, \dots, C$) as n_c, $\mathbf{f}_c$, and $\mathbf{S}_c$, respectively. To obtain unified presentation for the two formulations, we hereafter assume that the first and second order statistics are centered [19] for the standard formulation and *not* centered for the augmented formulation.

2.1. Standard formulation

Following the standard formulation, we model the mean vector of the cth Gaussian component of utterance u as

$$\boldsymbol{\mu}_c(u) = \mathbf{m}_c + \mathbf{T}_c\boldsymbol{\omega}(u), \tag{1}$$

where $\mathbf{m}_c$ is a bias term, matrix $\mathbf{T}_c$ is a projection matrix, and $\boldsymbol{\omega}(u)$ is a latent vector. The latent vector is shared among all the components and we assume that the prior over latent vectors is standard normal. Further, the covariance matrix of the cth Gaussian is modeled as

$$\mathbf{D}_c(u) = \mathbf{T}_c\boldsymbol{\Phi}(u)\mathbf{T}_c^{\mathsf{T}} + \boldsymbol{\Sigma}_c, \tag{2}$$

where $\boldsymbol{\Phi}(u)$ is the posterior covariance matrix of the latent vector, and $\boldsymbol{\Sigma}_c$ is the residual covariance matrix for component c [20].

The posterior covariance matrix $\boldsymbol{\Phi}(u)$ and the mean vector $\boldsymbol{\phi}(u)$ for the latent vector are obtained as

$$\boldsymbol{\Phi}(u) = \left(\mathbf{I} + \sum_{c=1}^{C} n_c(u)\mathbf{T}_c^{\mathsf{T}}\boldsymbol{\Sigma}_c^{-1}\mathbf{T}_c\right)^{-1}, \tag{3}$$

$$\boldsymbol{\phi}(u) = \boldsymbol{\Phi}(u)\left(\mathbf{p} + \sum_{c=1}^{C}\mathbf{T}_c^{\mathsf{T}}\boldsymbol{\Sigma}_c^{-1}\mathbf{f}_c(u)\right), \tag{4}$$

where $\mathbf{p}$ is the *prior offset*, which is $\mathbf{0}$ in the standard formulation.

The model is trained iteratively using an EM-algorithm, for which the update formulas for matrices $\mathbf{T}_c$ and $\boldsymbol{\Sigma}_c$ are given in [10]. In the beginning of training, the matrices $\mathbf{T}_c$ are initialized with random values drawn from the standard normal distribution. The initial bias terms $\mathbf{m}_c$ and the residual covariance matrices $\boldsymbol{\Sigma}_c$ are obtained as the means and covariances from *universal background model* (UBM) [12]. As the training progresses, the residual covariances get smaller as the first term of right-hand side of (2) starts to explain parts of the covariance structures of training utterances.

2.2. Augmented formulation

In the second formulation, we augment the bias terms $\mathbf{m}_c$ into the matrices $\mathbf{T}_c$. This is done by assuming non-zero mean for the prior over the first elements of the latent vectors. Then, equation (1) becomes

$$\boldsymbol{\mu}_c(u) = \mathbf{T}_c\boldsymbol{\omega}(u), \tag{5}$$

where $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{p}, \mathbf{I})$ with $\mathbf{p} = \begin{bmatrix} p & 0 & \cdots & 0\end{bmatrix}^{\mathsf{T}}, p \in \mathbb{R}$.

Assuming that the Baum-Welch statistics are not centralized, the equations (3) and (4) hold also for the augmented formulation. The EM update equations presented in [10] remain the same as well[1]. It is worth to note that because of the augmentation, the update of matrices $\mathbf{T}_c$ also updates the bias terms, which reside in the first columns of matrices $\mathbf{T}_c$.

The model initialization differs slightly from the standard formulation. First, we set $p = 100$ (same as in the Kaldi implementation) and then we fill the first columns of the randomly initialized matrices $\mathbf{T}_c$ with the values from the mean vectors of the UBM divided by p.

[1]Although the residual covariance update implemented in Kaldi might seem different than in [10], they can be shown to be equivalent.

3. Training enhancements

The update step of the model training can have many variations. The most basic one is to only update matrices $\mathbf{T}_c$, while also updating residual covariances $\boldsymbol{\Sigma}_c$ gives a slight improvement to the performance as we will demonstrate later. Another way to improve the model is to apply *minimum divergence re-estimation* to make the empirical distribution of i-vectors close to standard normal [9, 14]. The minimum divergence re-estimation is not quite as straightforward for the augmented formulation as for the standard one. To the best of our knowledge, the procedure for the augmented formulation is not documented elsewhere than in the source code comments of Kaldi, hence we will provide the key details in the following. Finally, further improvements can be obtained by realigning the training data during the training using the updated models.

3.1. Minimum divergence re-estimation

For the minimum divergence re-estimation, we accumulate the sums

$$\mathbf{h} = \frac{1}{U}\sum_{u=1}^{U}\boldsymbol{\phi}(u), \tag{6}$$

$$\mathbf{H} = \frac{1}{U}\sum_{u=1}^{U}\left[\boldsymbol{\Phi}(u) + \boldsymbol{\phi}(u)\boldsymbol{\phi}(u)^{\mathsf{T}}\right], \tag{7}$$

during the E-step. Then, a whitening matrix can be computed via *eigendecomposition* (alternatively, via *Cholesky decomposition*) of the covariance matrix $\mathbf{G} = \mathbf{H} - \mathbf{h}\mathbf{h}^T$. That is, if $\mathbf{G} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\mathsf{T}}$ is an eigendecomposition of $\mathbf{G}$, then the whitening transform is obtained as $\mathbf{P}_1 = \boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^{\mathsf{T}}$. Now, the update $\mathbf{T}_c^{\text{upd}} = \mathbf{T}_c\mathbf{P}_1^{-1}$, has an effect of whitening the training i-vectors.

In the standard formulation, the above update is sufficient for the minimum divergence estimation. In the augmented formulation, however, we need to apply another transform $\mathbf{P}_2$ to the matrices $\mathbf{T}_c^{\text{upd}}$ to conform to the prior offset assumption. In specific, after transforming i-vectors with $\mathbf{P}_1$ and $\mathbf{P}_2$, they should remain whitened and only the first element (prior offset) of the projected mean vector $\mathbf{P}_2\mathbf{P}_1\mathbf{h}$ should be non-zero.

One option for a transform that can satisfy the requirements set for $\mathbf{P}_2$ is a reflection about a hyperplane that goes through the origin. This type of transform is known as the *Householder transform* [21]. The Householder transform with a reflection hyperplane that is orthogonal to an unit length vector $\mathbf{a}$ is defined as

$$\mathbf{P}_2 = \mathbf{I} - 2\mathbf{a}\mathbf{a}^{\mathsf{T}}. \tag{8}$$

Now, the problem is to find $\mathbf{a}$ so that the projected mean vector is a scalar multiple of a unit vector $\mathbf{e}_1 = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}$. That is,

$$\mathbf{P}_2\mathbf{P}_1\mathbf{h} = b\mathbf{e}_1, \quad b \in \mathbb{R}. \tag{9}$$

It can be shown that one solution is

$$\mathbf{a} = \alpha\tilde{\mathbf{h}} + \beta\mathbf{e}_1, \tag{10}$$

where $\tilde{\mathbf{h}}$ is $\mathbf{P}_1\mathbf{h}$ normalized to unit length ($\tilde{\mathbf{h}} = \mathbf{P}_1\mathbf{h}/||\mathbf{P}_1\mathbf{h}||$) and

$$\begin{cases}\alpha = \frac{1}{\sqrt{2(1-\tilde{\mathbf{h}}[1])}} \\ \beta = -\alpha,\end{cases} \tag{11}$$

where $\tilde{\mathbf{h}}[1]$ is the first element of $\tilde{\mathbf{h}}$.

Now, the update $\mathbf{T}_c^{\text{upd}} = \mathbf{T}_c\mathbf{P}_1^{-1}\mathbf{P}_2^{-1}$ whitens and centers the training i-vectors with respect to the prior offset. Finally, the prior offset $\mathbf{p}$ is updated as follows:

$$\mathbf{p} = \mathbf{P}_2\mathbf{P}_1\mathbf{h}. \tag{12}$$

3.2. Realignment of training data

To compute the Baum-Welch statistics used in training, the frames of training utterances are first aligned to the components of the UBM by computing frame posterior probabilities. The posteriors and the Baum-Welch statistics are typically held constant during the training of i-vector extractor.

In [10], the frame alignments of the training utterances are updated during the training of factor analysis model for *automatic speech recognition* (ASR). The realignment is done per speaker basis using adapted GMM means and covariances. In the application of speaker recognition, however, this would be counterproductive as it would reduce the amount of speaker information in the latent vectors. What we propose instead, is updating the UBM means with the updated bias terms $\mathbf{m}_c$ and then using the updated UBM to realign the data, which can potentially lead to a better model fit. To obtain the updated bias terms from the augmented formulation, we simply take the first columns of matrices $\mathbf{T}_c$ and multiply them with p.

In summary, the augmented model with posterior updates is trained by iterating over the following five steps:

1. The computation of frame alignments and Baum-Welch statistics using the current UBM [12, 14].
2. **E-step:** The computation of posterior means and covariances for the latent vectors using (3) and (4) to accumulate the required terms for the M-step.
3. **M-step:** The update of matrices $\mathbf{T}_c$ followed by the update of residual covariances $\mathbf{\Sigma}_c$ [10].
4. **Minimum divergence re-estimation:** The update of matrices $\mathbf{T}_c$ using the transforms $\mathbf{P}_1$ and $\mathbf{P}_2$ followed by the update of the prior offset $\mathbf{p}$ using (12).
5. If not the last iteration, the update of the mean vectors of the UBM with the first columns of matrices $\mathbf{T}_c^{\text{upd}}$ multiplied by p.

After the model has been trained, the updated UBM is used in the testing phase to compute the frame posteriors.

4. Experiments

4.1. Experimentation setup

We built the acoustic front-end of our systems on the basis of Kaldi [11] i-vector recipe for VoxCeleb [22, 23]. That is, we relied on Kaldi to extract MFCCs, to perform voice activity detection (VAD), and to train the UBM. We used the same settings as in the Kaldi recipe: The MFCC vectors are 72-dimensional including delta and double-delta coefficients, and the UBM consists of 2048 components with full covariance matrices.

Following the Kaldi recipe, the UBM was trained using all of the data from the training parts of VoxCeleb1 and Voxceleb2 consisting of 1 277 344 utterances from 7325 speakers. The i-vector extractors were trained using the 100 000 longest utterances. To train the scoring back-end, the Kaldi recipe uses the whole training data, while we utilized only the VoxCeleb1 proportion to speed up the experimentation. Although this reduced the number of training speakers from 7325 to 1211, we did not observe degradation in speaker verification performance[2].

After the i-vector extraction, we centered and length normalized the i-vectors. In addition, if minimum divergence reestimation was not used, we also whitened the i-vectors before length normalization. Then, we reduced the dimensionality of i-vectors from 400 to 200 using *linear discriminant analysis* (LDA) before subjecting them to *probabilistic linear discriminant analysis* (PLDA) scoring [24]. For testing, we used adopted the VoxCeleb1 speaker verification protocol, which consists of 37 720 trials with an equal number of target and non-target trials.

[2]This might be explained by the fact that VoxCeleb1 has more reliable speaker labels than VoxCeleb2 [23].

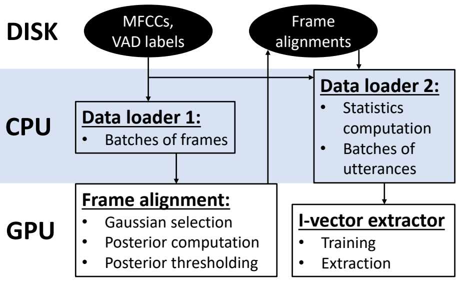


Figure 1: *An overview of computational flow of frame alignment, i-vector extraction, and model training using a GPU. To keep the GPU memory requirements constant, fixed size batches of frames and utterances are used for frame alignment and i-vector extraction, respectively.*

We ran the experiments on a server having Intel Xeon Gold 6152 CPU with 22 physical cores and NVIDIA Titan V GPU with 12 GB of memory. The file I/O operations were performed on a solid-state drive (SSD).

4.2. GPU implementation

In our implementations of frame alignment and i-vector extraction, we utilized PyTorch [25] for GPU computations, SciPy ecosystem [26] for computations in CPU, and PyKaldi [27] for reading files stored in Kaldi format. The implementations use multiple CPU cores in parallel as data loaders, which load, preprocess, and feed the data to the GPU (Figure 1). The data loaders function in parallel with respect to the GPU to keep the GPU utilized all the time.

For frame alignment, we use the same strategy as in Kaldi: First, to reduce the computational load, we use a UBM with diagonal covariance matrices to select the top-20 Gaussian components with the highest frame posteriors for each frame. Second, we compute the posteriors with only the selected components using a full covariance UBM. Finally, we discard the posteriors that are less than 0.025 and we linearly scale the remaining posteriors so that their sum equals to one. As a result, on average, only four Gaussian indices and the corresponding posteriors are stored to disk per frame.

The Baum-Welch statistics used in i-vector extractor training are computed in CPU, while the rest of the computation is done in GPU. The reason to compute statistics in CPU is as follows: For i-vector extraction implementation, it is natural to feed data in batches of utterances, and statistics provide a fixed size representation of utterances unlike the acoustic features. We opted not to compute statistics beforehand as the disk usage would be excessive; instead we recompute them during each iteration of i-vector extractor training.

With the settings laid out in Section 4.1, the GPU memory usage for alignment computation is about 2.5 GB and for i-vector extractor training about 4 GB. The frame alignment can be done about 3000 times faster than real time (including I/O operations), and assuming that the alignments are ready in the disk, i-vectors can be extracted 10 000 times faster than real

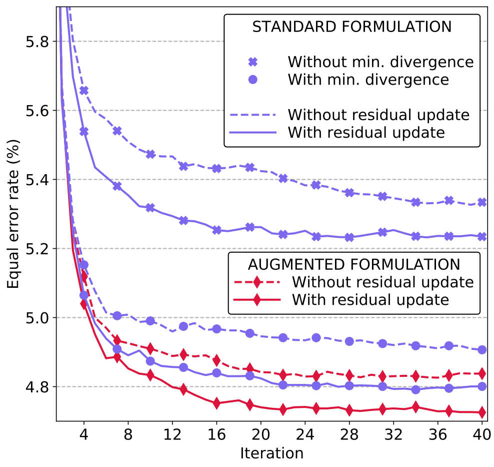


Figure 2: *System performance as function of number of iterations in i-vector extractor training. The frame alignments are not updated during the training. The standard formulation has four training variations, which are obtained by either performing or not performing minimum divergence re-estimation and either updating or not updating residual covariance matrices. In the augmented formulation, the minimum divergence re-estimation is always applied. Each curve is obtained as an average of five runs with different random initial values of* $\mathbf{T}_c$.

time. By using the GPU re-implementation of Kaldi's i-vector extractor training, we were able to obtain 25-fold reduction in the training times. This number was obtained by training both our GPU implementation and Kaldi's CPU implementation for five iterations and measuring the elapsed times. The training using Kaldi utilized all the available CPU cores in the server.

4.3. Speaker verification results

We began the experiments by comparing different variations of i-vector extractors to select the best one for further experiments with frame alignment updates. The results of the comparison are shown in Figure 2. We observe the following: First, the minimum divergence re-estimation to update the model hyper-parameters results in $7.5 - 9\%$ relative reduction in terms of equal error rate (EER). Second, the update of residual covariance matrices leads to $1.5 - 3\%$ relative reduction of error rates. Third, the augmented formulations obtain $1 - 2\%$ lower error rates (relative) than the standard formulations. Finally, we assert that 22 iterations are enough to reach the optimal speaker verification performance with the best performing extractors. As our results are averages of five runs, individual runs may converge faster than that. In addition, we confirmed that our assertion is correct by training the augmented model once for 200 iterations.

Based on the first experiment, we continued to experiment with the realignment of training data using the augmented formulation with residual covariance matrix updates. We varied the interval between the frame posterior updates ranging from updating on every iteration to updating only on every seventh iteration. We display the results in Figure 3. The findings are two-fold: First, the more frequently the frame posteriors are updated, the faster the performance improves. Second, updating the posteriors, no matter how frequently, leads to about

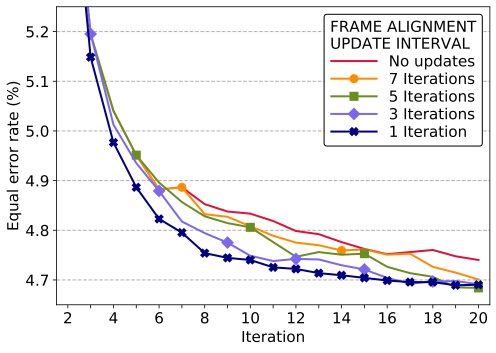


Figure 3: *Performance of the augmented formulation for varying intervals of frame alignment updates. The more often the alignments are updated, the faster the system performance improves. Each curve is obtained as an average of five runs with different random initializations.*

1% lower error rates (relative) compared to training without updates.

At best, we obtained an EER of 4.6%, which could be possibly made closer to 4.0% by carefully optimizing configurations in various parts of the system. For comparison, the state-of-the-art system, using x-vectors, obtains EER of 3.1% (reported in the Kaldi recipe). This is an expected performance difference between the i-vector and x-vector systems [28].

5. Discussion and conclusions

We have a couple of remarks from the practical aspect of the study. First, we found that by using the modern deep learning platforms, such as PyTorch, the implementation of GPU accelerated algorithms for generative models is almost as straightforward as it is with their non-GPU counterparts (*e.g.* NumPy). The only concern is the limited amount of memory in GPUs. This limitation can be often circumvented by relying on the computational power of GPUs to recompute values that do not fit into the memory.

The second remark concerns the update of the UBM means using the bias terms $\mathbf{m}_c$ of the model. For this purpose, we only used the augmented formulation, but it can be done also with the standard formulation by updating the means in the minimum divergence step using a formula $\mathbf{m}_c^{\text{upd}} = \mathbf{m}_c + \mathbf{T}_c\mathbf{h}$ [19]. However, we found that updating the means in this way did not work well together with residual covariance updates.

In summary, the results of the study showed that the choice of the training algorithm for i-vector extractor matters as the relative change in equal error rate between the worst and the best variations was 11.4%. For the optimal performance, our recommendation is to use the augmented formulation including the residual covariance updates and the updates of frame alignments. Additionally we found that the extractors reach their maximum performance after 22 training iterations.

6. Acknowledgements

This work was partially supported by Academy of Finland (proj. #309629) and by the Doctoral Programme in Science, Technology and Computing (SCITECO) of the UEF. The authors at UEF were also supported by NVIDIA Corporation with the donation of Titan V GPU.

7. References

[1] N. Dehak, "Discriminative and generative approaches for long- and short-term speaker characteristics modeling: application to speaker verification," Ph.D. dissertation, École de technologie supérieure, 2009.

[2] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification." in *Interspeech*, 2017, pp. 999–1003.

[3] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors," in *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, 2018, pp. 105–111. [Online]. Available: http://dx.doi.org/10.21437/Odyssey.2018-15

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

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

[6] O. Glembek, L. Burget, P. Matějka, M. Karafiát, and P. Kenny, "Simplification and optimization of i-vector extraction," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4516–4519.

[7] S. Cumani and P. Laface, "Factorized sub-space estimation for fast and memory effective i-vector extraction," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 248–259, 2014.

[8] L. Xu, K. A. Lee, H. Li, and Z. Yang, "Generalizing i-vector estimation for rapid speaker recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 4, pp. 749–759, 2018.

[9] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, vol. 14, pp. 28–29, 2005.

[10] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice modeling with sparse training data," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 3, pp. 345–354, 2005.

[11] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," IEEE Signal Processing Society, Tech. Rep., 2011.

[12] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital signal processing*, vol. 10, no. 1-3, pp. 19–41, 2000.

[13] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[14] P. Kenny, "A small footprint i-vector extractor," in *Odyssey*, vol. 2012, 2012, pp. 1–6.

[15] S. Madikeri, S. Dey, P. Motlicek, and M. Ferras, "Implementation of the standard i-vector system for the kaldi speech recognition toolkit," Idiap, Tech. Rep., 2016.

[16] A. Larcher, K. A. Lee, and S. Meignier, "An extensible speaker identification SIDEKIT in Python," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5095–5099.

[17] S. O. Sadjadi, M. Slaney, and L. P. Heck, "MSR identity toolbox v1.0: A MATLAB toolbox for speaker recognition research," 2013.

[18] D. Povey, L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow *et al.*, "The subspace Gaussian mixture model—A structured model for speech recognition," *Computer Speech & Language*, vol. 25, no. 2, pp. 404–439, 2011.

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

[20] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Speaker adaptation using an eigenphone basis," *IEEE transactions on speech and audio processing*, vol. 12, no. 6, pp. 579–589, 2004.

[21] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the ACM (JACM)*, vol. 5, no. 4, pp. 339–342, 1958.

[22] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[23] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[24] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[26] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[27] D. Can, V. R. Martinez, P. Papadopoulos, and S. S. Narayanan, "Pykaldi: A Python wrapper for Kaldi," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[28] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

Paper **VI**

INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection

Massimiliano Todisco[1], Xin Wang[2], Ville Vestman[3,6], Md Sahidullah[4], Héctor Delgado[1], Andreas Nautsch[1], Junichi Yamagishi[2,5], Nicholas Evans[1], Tomi Kinnunen[3], Kong Aik Lee[6]

[1]EURECOM, France
[2]National Institute of Informatics, Japan
[3]University of Eastern Finland, Finland
[4]Inria, France
[5]University of Edinburgh, UK
[6]NEC Corp., Japan

`info@asvspoof.org`

Abstract

ASVspoof, now in its third edition, is a series of community-led challenges which promote the development of countermeasures to protect automatic speaker verification (ASV) from the threat of spoofing. Advances in the 2019 edition include: (i) a consideration of both logical access (LA) and physical access (PA) scenarios and the three major forms of spoofing attack, namely synthetic, converted and replayed speech; (ii) spoofing attacks generated with state-of-the-art neural acoustic and waveform models; (iii) an improved, controlled simulation of replay attacks; (iv) use of the tandem detection cost function (t-DCF) that reflects the impact of both spoofing and countermeasures upon ASV reliability. Even if ASV remains the core focus, in retaining the equal error rate (EER) as a secondary metric, ASVspoof also embraces the growing importance of *fake audio detection*. ASVspoof 2019 attracted the participation of 63 research teams, with more than half of these reporting systems that improve upon the performance of two baseline spoofing countermeasures. This paper describes the 2019 database, protocols and challenge results. It also outlines major findings which demonstrate the real progress made in protecting against the threat of spoofing and fake audio.

Index Terms: spoofing, automatic speaker verification, ASVspoof, presentation attack detection, fake audio.

1. Introduction

The ASVspoof initiative[1] [1, 2, 3] spearheads research in anti-spoofing for automatic speaker verification (ASV). Previous ASVspoof editions focused on the design of spoofing countermeasures for synthetic and converted speech (2015) and replayed speech (2017). ASVspoof 2019 [4], the first edition to focus on all three major spoofing attack types, extends previous challenges in several directions, not least in terms of adressing two different use case scenarios: logical access (LA) and physical access (PA).

The LA scenario involves spoofing attacks that are injected directly into the ASV system. Attacks in the LA scenario are generated using the latest text-to-speech synthesis (TTS) and voice conversion (VC) technologies. The best of these algorithms produce speech which is perceptually indistinguishable from bona fide speech. ASVspoof 2019 thus aims to determine whether the advances in TTS and VC technology pose a greater threat to the reliability of ASV systems, as well as spoofing countermeasures. For the PA scenario, speech data is assumed to be captured by a microphone in a physical, reverberant space. Replay spoofing attacks are recordings of bona fide speech which are assumed to be captured, possibly surreptitiously, and then re-presented to the microphone of an ASV system using a replay device. In contrast to the 2017 edition of ASVspoof, the 2019 edition PA database is constructed from a far more controlled simulation of replay spoofing attacks that is also relevant to the study of fake audio detection in the case of, *e.g.* smart home devices.

While the *equal error rate* (EER) metric of previous editions is retained as a secondary metric, ASVspoof 2019 migrates to a new primary metric in the form of the ASV-centric tandem decision cost function (t-DCF) [5]. While the challenge is still a stand-alone spoofing detection task which does not require expertise in ASV, adoption of the t-DCF ensures that scoring and ranking reflects the comparative impact of both spoofing and countermeasures *upon an ASV system*.

This paper describes the ASVspoof 2019 challenge, the LA and PA scenarios, the evaluation rules and protocols, the t-DCF metric, the common ASV system, baseline countermeasures and challenge results.

2. Database

The ASVspoof 2019 database[2] encompasses two partitions for the assessment of LA and PA scenarios. Both are derived from the VCTK base corpus[3] which includes speech data captured from 107 speakers (46 males, 61 females). Both LA and PA databases are themselves partitioned into three datasets, namely training, development and evaluation which comprise the speech from 20 (8 male, 12 female), 10 (4 male, 6 female) and 48 (21 male, 27 female) speakers respectively. The three partitions are disjoint in terms of speakers and the recording conditions for all source data are identical. While the training and development sets contain spoofing attacks generated with the same algorithms/conditions (designated as *known* attacks), the evaluation set also contains attacks generated with different algorithms/conditions (designated as *unknown* attacks). Reliable spoofing detection performance therefore calls for systems that generalise well to previously-unseen spoofing attacks. With full descriptions available in the ASVspoof 2019 evaluation plan [4], the following presents a summary of the specific characteristics of the LA and PA databases.

2.1. Logical access

The LA database contains bona fide speech and spoofed speech data generated using 17 different TTS and VC systems. Data used for the training of TTS and VC systems also comes from the VCTK database but there is no overlap with the data contained in the 2019 database. Six of these systems are designated as known attacks, with the other 11 being designated

[1]http://www.asvspoof.org

[2]http://dx.doi.org/10.7488/ds/2555

[3]http://dx.doi.org/10.7488/ds/1994

as unknown attacks. The training and development sets contain known attacks only whereas the evaluation set contains 2 known and 11 unknown spoofing attacks. Among the 6 known attacks there are 2 VC systems and 4 TTS systems. VC systems use a neural-network-based and spectral-filtering-based approaches [6]. TTS systems use either waveform concatenation or neural-network-based speech synthesis using a conventional source-filter vocoder [7] or a WaveNet-based vocoder [8]. The 11 unknown systems comprise 2 VC, 6 TTS and 3 hybrid TTS-VC systems and were implemented with various waveform generation methods including classical vocoding, Griffin-Lim [9], generative adversarial networks [10], neural waveform models [8, 11], waveform concatenation, waveform filtering [12], spectral filtering, and their combination.

2.2. Physical access

Inspired by work to analyse and improve ASV reliability in reverberant conditions [13, 14] and a similar approach used in the study of replay reported in [15], both bona fide data and spoofed data contained in the PA database are generated according to a simulation [16, 17, 18] of their presentation to the microphone of an ASV system within a reverberant acoustic environment. Replayed speech is assumed first to be captured with a recording device before being replayed using a non-linear replay device. Training and development data is created according to 27 different acoustic and 9 different replay configurations. Acoustic configurations comprise an exhaustive combination of 3 categories of room sizes, 3 categories of reverberation and 3 categories of speaker/talker[4]-to-ASV microphone distances. Replay configurations comprise 3 categories of attacker-to-talker recording distances, and 3 categories of loudspeaker quality. Replay attacks are simulated with a random attacker-to-talker recording distance and a random loudspeaker quality corresponding to the given configuration category. Both bona fide and replay spoofing access attempts are made with a random room size, reverberation level and talker-to-ASV microphone distance.

Evaluation data is generated in the same manner as training and development data, albeit with different, random acoustic and replay configurations. The set of room sizes, levels of reverberation, talker-to-ASV microphone distances, attacker-to-talker recording distances and loudspeaker qualities, while drawn from the same configuration categories, are different to those for the training and development set. Accordingly, while the categories are the same and *known* a priori, the specific impulse responses and replay devices used to simulate bona fide and replay spoofing access attempts are different or *unknown*. It is expected that reliable performance will only be obtained by countermeasures that generalise well to these conditions, i.e. countermeasures that are not over-fitted to the specific acoustic and replay configurations observed in training and development data.

3. Performance measures and baselines

ASVspoof 2019 focuses on assessment of *tandem* systems consisting of both a spoofing countermeasure (CM) (designed by the participant) and an ASV system (provided by the organisers). The performance of the two combined systems is evaluated via the minimum normalized **tandem detection cost function** (t-DCF, for the sake of easier tractability) [5] of the form:

$$\text{t-DCF}_{\text{norm}}^{\text{min}} = \min_{s} \left\{ \beta P_{\text{miss}}^{\text{cm}}(s) + P_{\text{fa}}^{\text{cm}}(s) \right\}, \qquad (1)$$

[4]From hereon we refer to *talkers* in order to avoid potential confusion with loud*speakers* used to mount replay spoofing attacks.

where β depends on application parameters (priors, costs) *and* ASV performance (miss, false alarm, and spoof miss rates), while $P_{\text{miss}}^{\text{cm}}(s)$ and $P_{\text{fa}}^{\text{cm}}(s)$ are the CM miss and false alarm rates at threshold s. The minimum in (1) is taken over all thresholds on given data (development or evaluation) with a known key, corresponding to oracle calibration. While the challenge rankings are based on *pooled* performance in either scenario (LA or PA), results are also presented when decomposed by attack. In this case, β depends on the effectiveness of each attack. In particular, with everything else being constant, β is *inversely proportional to the ASV false accept rate for a specific attack*: the penalty when a CM falsely rejects bona fide speech is higher in the case of less effective attacks. Likewise, the relative penalty when a CM falsely accepts spoofs is higher for more effective attacks. Thus, while (1) appears to be deceptively similar to the NIST DCF, β (hence, the cost function itself) is automatically adjusted according to the effectiveness of each attack. Full details of the t-DCF metric and specific configuration parameters as concerns ASVspoof 2019 are presented in [4]. The EER serves as a secondary metric. The EER corresponds to a CM operating point with equal miss and false alarm rates and was the primary metric for previous editions of ASVspoof. Without an explicit link to the impact of CMs upon the reliability of an ASV system, the EER may be more appropriate as a metric for fake audio detection, *i.e.* where there is no ASV system.

The common ASV system uses *x-vector* speaker embeddings [14] together with a *probabilistic linear discriminant analysis* (PLDA) [19] backend. The x-vector model used to compute ASV scores required to compute the t-DCF is pre-trained[5] with the Kaldi [20] VoxCeleb [21] recipe. The original recipe is modified to include PLDA adaptation using disjoint, bona fide, in-domain data. Adaptation was performed separately for LA and PA scenarios since bona fide recordings for the latter contain additional simulated acoustic and recording effects. The ASV operating point, needed in computing β in (1), is set to the EER point based on target and nontarget scores.

ASVspoof 2019 adopted two CM baseline systems. They use a common Gaussian mixture model (GMM) back-end classifier with either *constant Q cepstral coefficient* (CQCC) features [22, 23] (B01) or *linear frequency cepstral coefficient* (LFCC) features [24] (B02).

4. Challenge results

Table 1 shows results[6] in terms of the t-DCF and EER for primary systems, pooled over all attacks. For the LA scenario, 27 of the 48 participating teams produced systems that outperformed the best baseline B02. For the PA scenario, the performance of B01 was bettered by 32 of the 50 participating teams. There is substantial variation in minimum t-DCF and EER for both LA and PA scenarios. The top-performing system for the LA scenario, T05, achieved a t-DCF of 0.0069 and EER of 0.22%. The top-performing system for the PA scenario, T28, achieved a t-DCF of 0.0096 and EER of 0.39%. Confirming observations reported in [5], monotonic increases in the t-DCF that are not always mirrored by monotonic increases in the EER show the importance of considering the performance of the ASV and CM systems *in tandem*. Table 1 also shows that the top 7 (LA) and 6 (PA) systems used neural networks whereas 9 (LA) and 10 (PA) systems used an ensemble of classifiers.

[5]`http://kaldi-asr.org/models/m7`

[6]As for previous editions of ASVspoof, results are anonymised, with individual teams being able to identify their position in the evaluation rankings via an identifier communicated separately to each of them.

Table 1: *Primary system results. Results shown in terms of minimum t-DCF and the CM EER [%]. IDs highlighted in grey signify systems that used neural networks in either the front- or back-end. IDs highlighted in* ***bold font*** *signify systems that use an ensemble of classifiers.*

ASVspoof 2019 LA scenario							
#	ID	t-DCF	EER	#	ID	t-DCF	EER
1	**T05**	0.0069	0.22	26	T57	0.2059	10.65
2	**T45**	0.0510	1.86	27	**T42**	0.2080	8.01
3	**T60**	0.0755	2.64	28	*B02*	0.2116	8.09
4	**T24**	0.0953	3.45	29	**T17**	0.2129	7.63
5	**T50**	0.1118	3.56	30	**T23**	0.2180	8.27
6	**T41**	0.1131	4.50	31	**T53**	0.2252	8.20
7	**T39**	0.1203	7.42	32	**T59**	0.2298	7.95
8	**T32**	0.1239	4.92	33	*B01*	0.2366	9.57
9	**T58**	0.1333	6.14	34	T52	0.2366	9.25
10	T04	0.1404	5.74	35	**T40**	0.2417	8.82
11	**T01**	0.1409	6.01	36	T55	0.2681	10.88
12	**T22**	0.1545	6.20	37	**T43**	0.2720	13.35
13	T02	0.1552	6.34	38	T31	0.2788	15.11
14	**T44**	0.1554	6.70	39	**T25**	0.3025	23.21
15	**T16**	0.1569	6.02	40	**T26**	0.3036	15.09
16	T08	0.1583	6.38	41	T47	0.3049	18.34
17	**T62**	0.1628	6.74	42	T46	0.3214	12.59
18	**T27**	0.1648	6.84	43	T21	0.3393	19.01
19	**T29**	0.1677	6.76	44	T61	0.3437	15.66
20	**T13**	0.1778	6.57	45	**T11**	0.3742	18.15
21	**T48**	0.1791	9.08	46	**T56**	0.3856	15.32
22	**T10**	0.1829	6.81	47	T12	0.4088	18.27
23	T54	0.1852	7.71	48	T14	0.4143	20.60
24	T38	0.1940	7.51	49	T20	1.0000	92.36
25	T33	0.1960	8.93	50	T30	1.0000	49.60

ASVspoof 2019 PA scenario							
#	ID	t-DCF	EER	#	ID	t-DCF	EER
1	**T28**	0.0096	0.39	27	**T29**	0.2129	8.48
2	**T45**	0.0122	0.54	28	**T01**	0.2129	9.07
3	**T44**	0.0161	0.59	29	T54	0.2130	11.93
4	**T10**	0.0168	0.66	30	T35	0.2286	7.77
5	**T24**	0.0215	0.77	31	T46	0.2372	8.82
6	**T53**	0.0219	0.88	32	**T34**	0.2402	10.35
7	**T17**	0.0266	0.96	33	*B01*	0.2454	11.04
8	**T50**	0.0350	1.16	34	**T38**	0.2460	9.12
9	**T42**	0.0372	1.51	35	**T59**	0.2490	10.53
10	**T07**	0.0570	2.45	36	T03	0.2593	11.26
11	T02	0.0614	2.23	37	**T51**	0.2617	11.92
12	**T05**	0.0672	2.66	38	T08	0.2635	10.97
13	**T25**	0.0749	3.01	39	**T58**	0.2767	11.28
14	**T48**	0.1133	4.48	40	T47	0.2785	10.60
15	T57	0.1297	4.57	41	**T09**	0.2793	12.09
16	**T31**	0.1299	5.20	42	T32	0.2810	12.20
17	**T56**	0.1309	4.87	43	T61	0.2958	12.53
18	T49	0.1351	5.74	44	*B02*	0.3017	13.54
19	**T40**	0.1381	5.95	45	**T62**	0.3641	13.85
20	**T60**	0.1492	6.11	46	T19	0.4269	21.25
21	T14	0.1712	6.50	47	T36	0.4537	18.99
22	**T23**	0.1728	7.19	48	**T41**	0.5452	28.98
23	**T13**	0.1765	7.61	49	T21	0.6368	27.50
24	**T27**	0.1819	7.98	50	T15	0.9948	42.28
25	**T22**	0.1859	7.44	51	T30	0.9998	50.19
26	**T55**	0.1979	8.19	52	T20	1.0000	92.64

4.1. CM analysis

Corresponding CM detection error trade-off (DET) plots (no combination with ASV) are illustrated for LA and PA scenarios in Fig. 1. Highlighted in both plots are profiles for the two baseline systems B01 and B02, the best performing primary systems for teams T05 and T28, and the same teams' single systems. Also shown are profiles for the overall best performing single system for the LA and PA scenarios submitted by teams T45 and, again, T28 respectively. For the LA scenario, very few systems deliver EERs below 5%. A dense concentration of systems deliver EERs between 5% and 10%. Of interest is the especially low EER delivered by the primary T05 system, which delivers a substantial improvement over the same team's best performing single system. Even the overall best performing single system of T45 is some way behind, suggesting that reliable performance for the LA scenario depends upon the fusion of complementary sub-systems. This is likely due to the diversity in attack families, namely TTS, VC and hybrid TTS-VC systems. Observations are different for the PA scenario. There is a greater spread in EERs and the difference between the best performing primary and single systems (both from T28) is much narrower. That a low EER can be obtained with a single system suggests that reliable performance is less dependent upon effective fusion strategies. This might be due to lesser variability (as compared to that for the LA) in replay spoofing attacks; there is only one family of attack which exhibits differences only in the level of convolutional channel noise.

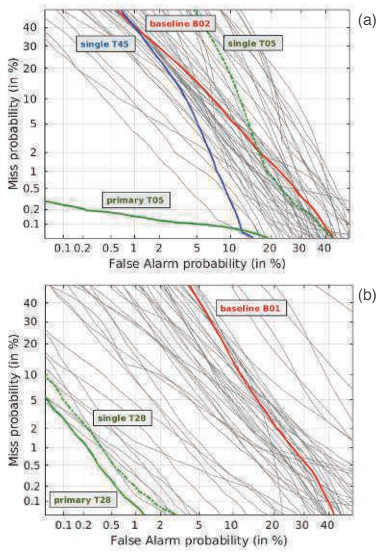


Figure 1: *CM DET profiles for (a) LA and (b) PA scenarios.*

4.2. Tandem analysis

Fig. 2 illustrates boxplots of the t-DCF when pooled (left-most) and when decomposed separately for each of the spoofing attacks in the evaluation set. Results are shown individually for the best performing baseline, primary and single systems whereas the boxes illustrate the variation in performance for the top-10 performing systems. Illustrated to the top of each boxplot are the EER of the common ASV system (when subjected to each attack) and the median CM EER across all primary systems. The ASV system delivers baseline EERs (without spoofing attacks) of 2.48% and 6.47% for LA and PA scenarios respectively.

As shown in Fig. 2(a) for the LA scenario, attacks A10, A13 and, to a lesser extent, A18, degrade ASV performance while being challenging to detect. They are end-to-end TTS with WaveRNN and a speaker encoder pretrained for ASV [25], VC using moment matching networks [26] and waveform filtering [12], and i-vector/PLDA based VC [27] using a DNN glottal vocoder [28], respectively. Although A08, A12, and A15 also use neural waveform models and threaten ASV, they are easier to detect than A10. One reason may be that A08, A12, A15 are pipeline TTS and VC systems while A10 is optimized in an end-to-end manner. Another reason may be that A10 transfers ASV knowledge into TTS, implying that advances in ASV also improve the LA attacks. A17, a VAE-based VC [29] with waveform filtering, poses little threat to the ASV system, but it is the

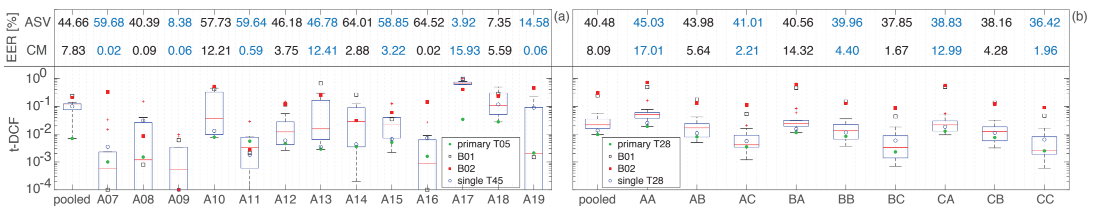


Figure 2: *Boxplots of the top-10 performing LA (a) and PA (b) ASVspoof 2019 submissions. Results illustrated in terms of t-DCF decomposed for the 13 (LA) and 9 (PA) attacks in the evaluation partion. ASV under attack and the median CM EER [%] of all the submitted systems are shown above the boxplots. A16 and A19 are known attacks.*

most difficult to detect and lead to the highest t-DCF. All the above attacks are new attacks not included in ASVspoof 2015.

More consistent trends can be observed for the PA scenario. Fig. 2(b) shows the t-DCF when pooled and decomposed for each of the 9 replay configurations. Each attack is a combination of different attacker-to-talker recording distances {A,B,C}X, and replay device qualities X{A,B,C} [4]. When subjected to replay attacks, the EER of the ASV system increases more when the attacker-to-talker distance is low (near-field effect) and when the attack is performed with higher quality replay devices (fewer channel effects). There are similar observations for CM performance and the t-DCF; lower quality replay attacks can be detected reliably whereas higher quality replay attacks present more of a challenge.

5. Discussion

Care must be exercised in order that t-DCF results are interpreted correctly. The reader may find it curious, for instance, that LA attack A17 corresponds to the *highest* t-DCF while, with an ASV EER of 3.92%, the attack is the *least effective*. Conversely, attack A16 provokes an ASV EER of almost 65%[7], yet the median t-DCF is among the lowest. So, does A17 — a weak attack — really pose a problem? The answer is affirmative: A17 *is* problematic, as far as the t-DCF is concerned. Further insight can be obtained from the attack-specific weights β of (1). For A17, a value of $\beta \approx 26$, indicates that the induced cost function provides 26 times higher penalty for rejecting bona fide users, than it does for missed spoofing attacks passed to the ASV system. The behavior of primary system T05 in Fig. 1(a), with an aggressively tilted slope towards the low false alarm region, may explain why the t-DCF is near an order of magnitude better than the second best system.

6. Conclusions

ASVspoof 2019 addressed two different spoofing scenarios, namely LA and PA, and also the three major forms of spoofing attack: synthetic, converted and replayed speech. The LA scenario aimed to determine whether advances in countermeasure design have kept pace with progress in TTS and VC technologies and whether, as result, today's state-of-the-art systems pose a threat to the reliability of ASV. While findings show that the most recent techniques, e.g. those using neural waveform models and waveform filtering, in addition to those resulting from transfer learning (TTS and VC systems borrowing ASV techniques) do indeed provoke greater degradations in ASV performance, there is potential for their detection using countermeasures that combine multiple classifiers. The PA scenario aimed to assess the spoofing threat and countermeasure performance via simulation with which factors influencing replay spoofing attacks could be carefully controlled and studied. Simulations consider variation in room size and reverberation time, replay device quality and the physical separation between both talkers and attackers (making surreptitious recordings) and talkers and the ASV system microphone. Irrespective of the replay configuration, all replay attacks degrade ASV performance, yet, reassuringly, there is promising potential for their detection.

Also new to ASVspoof 2019 and with the objective of assessing the impact of both spoofing and countermeasures upon ASV reliability, is adoption of the ASV-centric t-DCF metric. This strategy marks a departure from the independent assessment of countermeasure performance in isolation from ASV and a shift towards cost-based evaluation. Much of the spoofing attack research across different biometric modalities revolves around the premise that spoofing attacks are harmful and should be detected at any cost. That spoofing attacks have potential for harm is not in dispute. It does not necessarily follow, however, that *every* attack *must* be detected. Depending on the application, spoofing attempts could be extremely rare or, in some cases, ineffective. Preparing for a worst case scenario, when that worst case is unlikely in practice, incurs costs of its own, *i.e.* degraded user convenience. The t-DCF framework enables one to encode explicitly the relevant statistical assumptions in terms of a well-defined cost function that generalises the classic NIST DCF. A key benefit is that the t-DCF disentangles the roles of ASV and CM developers as the error rates of the two systems are still treated independently. As a result, ASVspoof 2019 followed the same, familiar format as previous editions, involving a low entry barrier — participation still requires no ASV expertise and participants need submit countermeasures scores only — the ASV system is provided by the organisers and is common to the assessment of all submissions. With the highest number of submissions in ASVspoof's history, this strategy appears to have been a resounding success.

7. Acknowledgements

The authors express their profound gratitude to the 27 persons from 14 organisations who contributed to creation of the LA database. The work was partially supported by: JST CREST Grant No. JPMJCR18A6 (VoicePersonae project), Japan; MEXT KAKENHI Grant Nos. (16H06302, 16K16096, 17H04687, 18H04120, 18H04112, 18KT0051), Japan; the VoicePersonae and RESPECT projects funded by the French Agence Nationale de la Recherche (ANR); the Academy of Finland (NOTCH project no. 309629); Region Grand Est, France. The authors at the University of Eastern Finland also gratefully acknowledge the use of computational infrastructures at CSC – IT Center for Science, and the support of NVIDIA Corporation with the donation of a Titan V GPU used in this research.

[7]Scores produced by spoofing attacks are higher than those of genuine trials.

7. References

[1] N. Evans, T. Kinnunen, and J. Yamagishi, "Spoofing and countermeasures for automatic speaker verification," in *Proc. Interspeech, Annual Conf. of the Int. Speech Comm. Assoc.*, Lyon, France, August 2013, pp. 925–929.

[2] Z. Wu, T. Kinnunen, N. Evans, J. Yamagishi, C. Hanilçi, M. Sahidullah, and A. Sizov, "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *Proc. Interspeech, Annual Conf. of the Int. Speech Comm. Assoc.*, Dresden, Germany, September 2015, pp. 2037–2041.

[3] T. Kinnunen, M. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. Lee, "The ASVspoof 2017 challenge: Assessing the limits of replay spoofing attack detection," in *Proc. Interspeech, Annual Conf. of the Int. Speech Comm. Assoc.*, 2017, pp. 2–6.

[4] ASVspoof 2019: the automatic speaker verification spoofing and countermeasures challenge evaluation plan. [Online]. Available: http://www.asvspoof.org/asvspoof2019/asvspoof2019_evaluation_plan.pdf

[5] T. Kinnunen, K. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "t-DCF: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification," in *Proc. Odyssey*, Les Sables d'Olonne, France, June 2018.

[6] D. Matrouf, J.-F. Bonastre, and C. Fredouille, "Effect of speech transformation on impostor acceptance," in *Proc. ICASSP*, vol. 1. IEEE, 2006, pp. 933–936.

[7] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[8] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[9] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. ASSP*, vol. 32, no. 2, pp. 236–243, 1984.

[10] K. Tanaka, T. Kaneko, N. Hojo, and H. Kameoka, "Synthetic-to-natural speech waveform conversion using cycle-consistent adversarial networks," in *Proc. SLT*. IEEE, 2018, pp. 632–639.

[11] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5916–5920.

[12] K. Kobayashi, T. Toda, G. Neubig, S. Sakti, and S. Nakamura, "Statistical singing voice conversion with direct waveform modification based on the spectrum differential," in *Proc. Interspeech*, 2014, pp. 2514–2518.

[13] T. Ko, V. Peddinti, D. Povey, M. L. Seltzer, and S. Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5220–5224.

[14] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.

[15] A. Janicki, F. Alegre, and N. Evans, "An assessment of automatic speaker verification vulnerabilities to replay spoofing attacks," *Security and Communication Networks*, vol. 9, no. 15, pp. 3030–3044, 2016.

[16] D. R. Campbell, K. J. Palomäki, and G. Brown, "A MATLAB simulation of "shoebox" room acoustics for use in research and teaching." *Computing and Information Systems Journal, ISSN 1352-9404*, vol. 9, no. 3, 2005.

[17] E. Vincent. (2008) Roomsimove. [Online]. Available: http://homepages.loria.fr/evincent/software/Roomsimove_1.4.zip

[18] A. Novak, P. Lotton, and L. Simon, "Synchronized swept-sine: Theory, application, and implementation," *J. Audio Eng. Soc*, vol. 63, no. 10, pp. 786–798, 2015. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=18042

[19] S. J. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[20] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," IEEE Signal Processing Society, Tech. Rep., 2011.

[21] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[22] M. Todisco, H. Delgado, and N. Evans, "A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients," in *Proc. Odyssey*, Bilbao, Spain, 6 2016.

[23] M. Todisco, H. Delgado, and N. Evans, "Constant Q cepstral coefficients: A spoofing countermeasure for automatic speaker verification," *Computer Speech & Language*, vol. 45, pp. 516 – 535, 2017.

[24] M. Sahidullah, T. Kinnunen, and C. Hanilçi, "A comparison of features for synthetic speech detection," in *Proc. Interspeech, Annual Conf. of the Int. Speech Comm. Assoc.*, Dresden, Germany, 2015, pp. 2087–2091.

[25] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *CoRR*, vol. abs/1806.04558, 2018.

[26] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *International Conference on Machine Learning*, 2015, pp. 1718–1727.

[27] T. Kinnunen, L. Juvela, P. Alku, and J. Yamagishi, "Non-parallel voice conversion using i-vector PLDA: Towards unifying speaker verification and transformation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5535–5539.

[28] L. Juvela, B. Bollepalli, X. Wang, H. Kameoka, M. Airaksinen, J. Yamagishi, and P. Alku, "Speech waveform synthesis from MFCC sequences with generative adversarial networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5679–5683.

[29] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding Wasserstein generative adversarial networks," in *Proc. Interspeech 2017*, 2017, pp. 3364–3368.

Paper **VII**

Available online at www.sciencedirect.com

ScienceDirect

Computer Speech & Language 59 (2020) 36–54

www.elsevier.com/locate/csl

Voice Mimicry Attacks Assisted by Automatic Speaker Verification[☆]

Ville Vestman[1,a], Tomi Kinnunen[*,a], Rosa González Hautamäki[a], Md Sahidullah[b]

[a] *School of Computing, University of Eastern Finland, Joensuu, FI-80101, Finland*
[b] *Université de Lorraine, CNRS, Inria, LORIA, Nancy F-54000, France*

Received 29 November 2018; received in revised form 26 February 2019; accepted 24 May 2019
Available online 12 June 2019

Abstract

In this work, we simulate a scenario, where a publicly available ASV system is used to enhance mimicry attacks against another closed source ASV system. In specific, ASV technology is used to perform a similarity search between the voices of recruited attackers (6) and potential target speakers (7,365) from VoxCeleb corpora to find the closest targets for each of the attackers. In addition, we consider 'median', 'furthest', and 'common' targets to serve as a reference points.

Our goal is to gain insights how well similarity rankings transfer from the attacker's ASV system to the attacked ASV system, whether the attackers are able to improve their attacks by mimicking, and how the properties of the voices of attackers change due to mimicking. We address these questions through ASV experiments, listening tests, and prosodic and formant analyses. For the ASV experiments, we use i-vector technology in the attacker side, and x-vectors in the attacked side. For the listening tests, we recruit listeners through crowdsourcing.

The results of the ASV experiments indicate that the speaker similarity scores transfer well from one ASV system to another. Both the ASV experiments and the listening tests reveal that the mimicry attempts do not, in general, help in bringing attacker's scores closer to the target's. A detailed analysis shows that mimicking does not improve attacks, when the natural voices of attackers and targets are similar to each other. The analysis of prosody and formants suggests that the attackers were able to considerably change their speaking rates when mimicking, but the changes in F0 and formants were modest. Overall, the results suggest that untrained impersonators do not pose a high threat towards ASV systems, but the use of ASV systems to attack other ASV systems is a potential threat.

Keywords: Speaker verification; Mimicry; Crowdsourcing; Spoofing; Automatic target speaker selection; Perceptual speaker similarity; Prosody

☆ This paper has been recommended for acceptance by Roger Moore.
* Corresponding author.
E-mail addresses: vvestman@cs.uef.fi (V. Vestman), tkinnu@cs.joensuu.fi (T. Kinnunen), rgonza@cs.uef.fi (R. González Hautamäki), md.sahidullah@inria.fr (M. Sahidullah).
[1] A part of the work of the first author was carried out during an intership at NEC.

https://doi.org/10.1016/j.csl.2019.05.005
0885-2308/ 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Security is of key importance in today's society where information processing gets increasingly digital, automated and lacks human-to-human communication. We need new ways to protect our data records from unauthorized access. Alongside with the traditional means of user authentication, biometric technology has emerged as one of the potential solutions. The use of human voice for strong user authentication is attractive especially under remote, unattended scenarios and due to the readily available infrastructure (namely, telephones) to scale it up easily.

Similar to the traditional means of user authentication, however, biometric systems are prone to malicious attacks by hackers. It is no longer news, neither to the research community nor to the general public, that biometric systems can be fooled through various *representation attacks* (Ratha et al., 2001; ISO/IEC 30107-1:2016, 2016), also known as *spoofing attacks*. A spoofing attack involves an adversary (attacker) who aims at masquerading oneself as another targeted user to gain illegitimate access to the targeted person's data. Unprotected *automatic speaker verification* (ASV) systems can be easily spoofed using replay, voice conversion (VC) and text-to-speech (TTS) attacks (Wu et al., 2015a). Since the attacks are typically not perfect but contain either processing artifacts or display degraded audio quality, they can be detected to a certain extent. To this end, community-driven challenges such as ASVspoof (Wu et al., 2015b) and AVspoof (Ergunay et al., 2015) were launched for an organized study of *spoofing countermeasures*. In the context of security, the continuous arms race between attacks and their defenses is well known (Biggio and Roli, 2018): so as to develop effective countermeasures, it is necessary to understand the attacks. The speech synthesis community has independently launched *voice conversion challenges* (Toda et al., 2016; Lorenzo-Trueba et al., 2018b) to advance VC methods (though targeted primarily for human listeners rather than for ASV spoofing). To sum up, within the past few years, active and dynamic communities both at the 'attack' and 'defense' sides of ASV have emerged. There is now a far better understanding of the technology-based attacks and their defenses against ASV systems than half a decade ago — see (Sahidullah et al., 2018) for an up-to-date review.

In this study we focus on a nearly-forgotten ASV attack − *mimicry* (impersonation). Unlike the technology-induced attacks, mimicry involves *human*-based modification of one's voice production. The question of recognizer vulnerability against mimicry was addressed at least around half a century ago (Luck, 1969; Endres et al., 1971) and has remained a cursory topic within the ASV field (Lau et al., 2004; 2005; Mariéthoz and Bengio, 2005; Eriksson, 2010; González Hautamäki et al., 2015; Farrús, 2018). While ASV vulnerability caused by technical attacks is widely reported, less (reliable) information is available on effectivess of mimicry, primarily due to adoption of small and proprietary datasets. The only conclusions that one can possibly extrapolate from the prior studies on mimicry effect against ASV is that the results depend on a specific study. This suggests that mimicry is less consistent attack compared to replay, VC and TTS that are repeatable reported to be successful in spoofing ASV systems.

The authors are aware of the difficulties in collecting mimicry data from professional artists (González Hautamäki et al., 2015), whose prevalence in the general population is arguably very low. Nonetheless, *if* mimicry attacks could be shown to be a threat to ASV, it would be conceivably challenging to devise countermeasures: natural human speech lacks processing artifacts that enable detection of technical attacks. Thus, we argue that it is important to keep mimicry also in the list of potential attacks against ASV. Besides the security aspect, mimicry could potentially help us in the design of better ASV methods for voice comparison.

Of particular interest in this work are mimicry attacks against persons whose voice data is exposed in a public domain in large quantities — such as celebrities or anyone streaming or uploading massive amounts of his/her videos to the Internet. In line with the recent EU's *General Data Protection Regulation* (GDPR) GDPR, intended to protect the privacy of individuals, it is important to assess potential risks associated with multimedia data in the public domain; we elaborate on this emerging problem further in Section 2. Differently from most prior studies, we focus on *technology-assisted* mimicry attacks. In specific, we use the ASV technology itself to identify potential target speakers to be subjected to mimicry attacks. The idea is to identify targets whose voice is *a priori* similar to that of the attacker's voice in terms of acoustic parameters. The assumption is that nearby target speakers might be easier to mimic due to potentially fewer articulatory or voice source modifications required. Two related prior studies are (Lau et al., 2004) and (Panjwani and Prakash, 2014) which involve search of either targets (Lau et al., 2004) or attackers (Panjwani and Prakash, 2014) from a pool of candidates. The authors of (Lau et al., 2004) used a Gaussian mixture model (GMM) system to find closest, intermediate and furthest target speakers from YOHO corpus for two naive impersonators, leading to substantially increased false acceptance rate for the closest targets. In (Panjwani and

Prakash, 2014), the authors selected impersonators (rather than targets) through a commercial crowd-sourcing platform based on self-judgment and further refinement using ASV.

Our study can be seen as an attempt to reproduce the findings of (Lau et al., 2004) using up-to-date ASV technology and a far larger target candidate set (7,365 celebrities pooled from VoxCeleb1 (Nagrani et al., 2017) and VoxCeleb2 (Chung et al., 2018)). Besides the order of magnitude larger target speaker pool and adoption of state-of-the-art ASV systems, there is a key difference in the research methodology as well: unlike (Lau et al., 2004) that used a *single* GMM recognizer, we include two *different* ASV systems as illustrated in Fig. 1. We argue that it is unrealistic for the attacker to interact many times with the targeted ASV, as done in that past work. In our attack model, therefore, the attacker uses an offline, publicly available *substitute* ASV system to first identify which speakers to attack; ideally, the substitute system would behave similar to the attacked ASV system. This idea bears some resemblance to *black box attacks* (Papernot et al., 2017) in adversarial machine learning (Biggio and Roli, 2018), though our adversary is not a machine learning algorithm but a human. Further, those methods use either classifier output score or decision to optimize the attacks, while we assume that the attacker receives no feedback from the attacked system in any form. Thus, we expect that our attacks are not strong, but we argue that they are *realistic* given the abundance of both voice data and ASV implementations in the public domain. We seek to answer the question whether the use of ASV technology itself could increase the risk of an attacker being falsely accepted by (another) ASV system.

A preliminary version of this work appears in Kinnunen et al. (2019). Our preliminary findings in that work suggested a *negative* result — *i.e.* that mimicry attempts, even when the target speakers were selected with automatic speaker identification, would not have left the attacked ASV systems vulnerable. We are not entirely content with just this finding, however — we are interested to understand the reasons. To this end, the present work substantially extends Kinnunen et al. (2019) by contrastive automatic, perceptual, prosody, and formant analyses. In particular, we include (i) **analysis of domain mismatch in ASV score domain** (presented in Section 6), (ii) a **human benchmark** of speaker similarity (presented in Section 7), and (iii) **prosody and formant** analysis (presented in Section 8). Additionally, (iv) Section 2 provides a broad background context to our work. None of the above were provided in Kinnunen et al. (2019). The score domain analysis seeks to answer whether the negative finding might have been due to condition differences across our attacker and celebrity corpora. The human benchmark, implemented via crowdsourcing, serves for a reference point to the automatic methods. Finally, the prosody and formant analyses serve to study changes in the speaking rate, fundamental frequency (F0), and formants induced by mimicry. Our hypothesis is that some of these 'broad' speech parameters might be among the prominent cues that a naive mimic attempts to primarily modify towards the target speaker. While this article is intended to be as self-contained as possible, the interested reader may consult additional online material Vestman et al. (2019) for further details about our text prompts and target speakers.

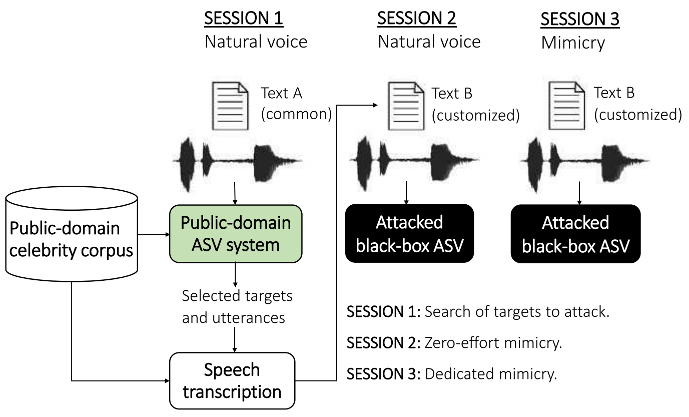


Fig. 1. Automatic speaker verification (ASV) assisted mimicry attack: attacker uses a public-domain ASV system to select target speakers matched with his/her voice from a public celebrity database. The attacker then practices target speaker mimicry, intended to attack another independently developed ASV system.

2. Attacks on speaker verification systems with found data

The amount of personal data that people upload to the Internet increases year by year. Enabled by popular social media platforms and other picture/video sharing services, people upload (or stream) their self-portraits (selfies), voice samples and video clips much more easily — perhaps more carelessly — than in the past. The general public may be unaware that their face photos, videos and voice samples contain biometric traits and form potentially their ‘unique’ identifiers[2]. Somewhat paradoxically, of a specific concern is the rapidly advancing biometric technology *itself*. The aim of biometric technology, similar to the traditional ways of user authentication, is to regulate access to a restricted domain. The basic premise is that a biometric database administrator (such as the police, a border control officer, or a bank) has sufficient security countermeasures to protect their biometric database and systems from being hacked or tampered. But what if the user decides to voluntarily expose his or her biometric data to the public? Very few of us would purposefully upload our credit card number or a photo-copy of our passport to a public website, but uploading our face and voice data does not seem to concern many. It is important to address the potential risk scenarios of misuse of personal data, and to make the general public aware of the potential risks of uploading their data to a public domain. Awareness on the potential risks among the professional community has increased due to initiatives such as EU’s IC1206 COST action[3] that focused on de-identification and privacy protection of multimedia data (see (Ribaric et al., 2016) for a review). The overall picture is not yet complete, however, and human voice has received far less attention than image-based biometric traits in this context.

One potential risk is that biometric data that is not searchable or indexable using today’s technology might become so tomorrow. Imagine a search engine that uses face or speaker recognition to cross-link someone’s sensitive personal multimedia data — such as sexually explicit photographs shared confidently with one’s partner but leaked to a porn website; or a video portraying someone under the influence of drugs — with his or her personal website or social media profile. Other risks could include fabricating a ‘digital clone’ of someone using machine learning — recent warning examples are provided by the so-called *deepfakes* Chung et al. (2017); Liu et al. (2017); Suwajanakorn et al. (2017), realistic-appearing but fabricated or tampered videos portraying a targeted person created with the aid of deep learning (the interested reader is pointed to to Chesney and Citron (2018) for a detailed review of potential societal, ethical and legal implications of deepfakes). In the context of speaker verification in specific, Lorenzo-Trueba et al. (2018a) addressed voice cloning of a well-known celebrity (the former US president Barack Obama). Even if the result was essentially negative (the cloned voice samples were detectable as artificial ones using a spoofing countermeasure), machine learning, including voice cloning techniques, do not stand still.

As current machine learning models require large training sets, one may argue that persons who have more (and of technically higher-quality) data in the Internet might become more easily exposed to novel, yet unforeseen, types of attacks and misuse in the future. Our present study is framed in the context of *celebrity* voices (due to the adoption of the VoxCeleb corpus) but we intend it as a proxy to address a specific risk associated with anyone having large quantities of biometric data in a public domain, often referred to as *found data*. In specific, we carry out empirical assessment of attacks on voice biometric system with the help of found audio data. This type of attacks have received surprisingly little attention in the literature. Unlike the use of publicly available tools for voice cloning of a specific target, we look for a speaker with the most similar voice and use him/her as an imposter. We use target speaker’s publicly available voice *data* and publicly available ASV tool for the voice similarity search.

The potential threat of natural impersonated voice, also known as mimicry González Hautamäki et al. (2015), has been studied in a limited number of target speakers and mimickers Luck (1969); Lau et al. (2004); González Hautamäki et al. (2015); Zetterholm et al. (2004). The present work is related to the study on the impact of the voice impersonation in ASV where the impersonator and potential target speakers are selected from large set of speakers. This enables us to choose the those impersonator-target pairs who are already similar in their natural voice. Surprisingly, the studies involving the search of potential attackers and the assessment of their ability to break the biometric security system are very limited. For other behavioral biometric traits (than voice), perhaps the only related study is done with *shoulder surfing* attack in the context of touch input implicit authentication Khan et al. (2016). This

[2] The authors argue that ‘unique’ is a misleading term in the context of biometrics where decisions are not based on exact pattern matching but probabilistic reasoning.

[3] https://www.cost.eu/actions/IC1206

demonstrated that when potential attackers are selected and trained to perform targeted mimicry, this authentication method is highly prone to such attacks.

The closest prior work in spirit to our study is Lau et al. (2005) where the authors studied the effect of mimicry in ASV with two professional imitators and four non-professional imitators. The closest speaker for each imitator was chosen from YOHO corpus of 138 speakers using Gaussian mixture model (GMM) based likelihood. The study indicated that, when mimicking the most similar speaker, the professionals did not achieve better mimicry performance than non-professional imitators. On the other hand, the professional imitators were more successful at mimicry when the target speaker is different from the most similar speaker. In another study crowdsourcing is used to select the best imitator for a set of 53 target speakers Panjwani and Prakash (2014). The authors used GMM-based ASV system for finding the imitators from a set of 176 participants. As a first step, the participants were asked to speak in natural and mimicked voices. Then an ASV system was used to filter the candidates by assessing the closeness of their voice samples to the target speakers. Finally, a set of good imitators were confirmed based on the performance of filtered candidates on multiple imitation tasks.

In contrast to the studies in Lau et al. (2004, 2005); Panjwani and Prakash (2014) with limited number of target speakers (and use of a single ASV system only), the current work uses two large publicly available datasets, VoxCeleb1 and VoxCeleb2, consisting of more than 7,000 speakers to search the targets corresponding to the six recruited participants who are native Finnish speakers. In addition to the impersonator-specific closest, median, and furthest targets, we also consider a common celebrity target. This is to evaluate the impersonator's natural ability to mimic a known person. Further, the target speakers are chosen from both Finnish and non-Finnish speakers to assess impersonator's success rate for native and non-native targets.

3. ASV-assisted mimicry attacks

3.1. Attack implementation

Let $\mathcal{T} = \{T_j\}_{j=1}^{J}$ denote a set of unique, publicly known **target speaker** identities and let $\mathcal{A} = \{A_k\}_{k=1}^{K}$ denote a set of **attacker** identities. The aim of an attacker $A \in \mathcal{A}$ is to masquerade him/herself as a specific target $T \in \mathcal{T}$ that he/she pre-selects using automatic speaker recognition technology. We assume that $J \gg K$ — that is, an attacker is relatively infrequent, but there are many natural persons who have their voice samples available in a public domain. Celebrities and anyone actively uploading or streaming their video or voice data to social media platforms are representative examples.

Given a pair of speech utterances (or a pair of *collections* of multiple utterances), (U_i, U_j), an **automatic speaker verification** (ASV) system (speaker detector), $\mathcal{D}(U_i, U_j)$ computes a *detection score*, $s_{ij} \in \mathbb{R}$, typically a *log-likelihood ratio* (LLR),

$$s_{ij} = \log \frac{p(U_i, U_j | H_0)}{p(U_i, U_j | H_1)}, \tag{1}$$

where the null hypothesis H_0 states that U_i and U_j originate from the same speaker and its complement H_1 states they originate from two different speakers. In this work, utterances are represented as fixed-sized *embeddings* using either *identity vectors* (i-vectors) (Dehak et al., 2011) or *x-vectors* (Snyder et al., 2018). If either U_i or U_j consist of multiple utterances, their embeddings are averaged. The LLR computation uses *probabilistic linear discriminant analysis* (PLDA) Prince and Elder (2007) scoring. The higher the LLR score, the stronger the support for the null hypothesis. We consider two different types of ASV systems. The first one, **attacker's ASV** ($\mathcal{D}_{\text{pub}}$), is a public-domain ASV implementation while the latter, **black-box ASV** ($\mathcal{D}_{\text{black}} \neq \mathcal{D}_{\text{pub}}$), is the system which the attacker attempts to hack into as a specific target. The attacker does not have access to the internal workings of $\mathcal{D}_{\text{black}}$ or its outputs to optimize mimicry attacks. The attack proceeds as follows:

ASV-assisted target speaker selection for mimicry attack

1. Attacker $A \in \mathcal{A}$ records his/her natural voice sample, $\mathcal{U}_{\text{nat}}$ (one or several utterances).
2. A uses $\mathcal{D}_{\text{pub}}$ to compute scores $\{s_j\}_{j=1}^{J}$ between $\mathcal{U}_{\text{nat}}$ and all the targets in a public domain. A picks the **closest target**, $j^* = \arg\max_{j=1}^{J} \mathcal{D}_{\text{pub}}(\mathcal{U}_{\text{nat}}, U_j)$, where U_j contains all the public recordings of target T_j.

Table 1
Details of the speaker verification systems used to simulate targeted impersonation attack against automatic speaker verification. The attacker is assumed to not have information about the attacked system, and hence the attacker's system differs from the attacked system.

	Attacker's ASV system ($\mathcal{D}_{\text{pub}}$)	Attacked ASV system ($\mathcal{D}_{\text{black}}$)
Type	Text-independent	Text-independent
Implementation	MSR Identity Toolkit (MATLAB)	Kaldi (c++)
Sampling rate	16 kHz	16 kHz
Acoustic features	60 MFCCs (20 static+20-Δ+20-ΔΔ), RASTA, SAD, CMVN	30 MFCCs (no deltas), Sliding CMN normalization, SAD
Embedding type	i-vector (400-D)	x-vector (512-D)
Back-end / scoring	LDA (250-D)+PLDA (simplified, 200-D)	LDA (200-D)+PLDA (2-cov)
Development data	Librispeech (train-clean-360 and train-clean-100 subsets), WSJ0 and WSJ1	VoxCeleb2, training part of VoxCeleb1
Data augmentation	None	Reverberation, noise, music, babble
EER*	12.84 (%)	3.11 (%)

* EER for VoxCeleb1 test protocol

3. A further uses $\mathcal{D}_{\text{pub}}$ to pick the top-scoring utterances of T_{j^*} similarly.
4. A listens to the selected utterance(s) and tries to adjust his/her voice towards the target. Once completed practicing, A submits a mimicked test utterance U_{mimic} to $\mathcal{D}_{\text{black}}(U_{\text{mimic}}, U_{j^*})$ with identity claim T_{j^*} (aiming to be accepted as T_{j^*}).

Note that in our model, the attacker uses the public-domain ASV system only to select the target speakers. In some prior work, such as (Zetterholm et al., 2004), ASV score was provided as feedback for the impersonators to improve their mimicry skills. We do not provide ASV (or other) feedback signals to our attackers. The main reason is that the ASV score is not necessarily intuitive to humans. For instance, a low attacker-to-target ASV score does not suggest *how* to modify one's voice production so as to improve the score. Providing intuitive feedback, for instance in terms of suggested articulatory or voice source modifications, would require a different system (and user interface) design. In our model, the attacker uses a readily-available public-domain ASV system to rank and select potential target speakers, but *without* any further numerical feedback or system optimization. Such 'passive' ASV system could be, for instance, a voice search service that finds most similar speakers to the user's voice from a public video archive — see Vestman et al. (2019); Intelligent Voice (2019) as examples.

Both the attacker's and the attacked ASV systems are *text-independent, i.e.* none assumes the spoken contents of the compared enrollment and test utterances to match. Even if properly-optimized text-dependent ASV systems can provide higher recognition accuracy, text-independent ASV systems provide more flexibility and are justifiable in certain authentication applications, such as secure teleconferencing and telephone banking. The use of text-independent ASV systems in this study was, in fact, *necessary* as we have no control over the text content in the celebrity corpus (VoxCeleb).

3.2. Public-domain (attacker's) ASV system

The attacker's ASV system uses i-vector front-end (Dehak et al., 2011) and probabilistic discriminant analysis (PLDA) (Prince and Elder, 2007) back-end to compute speaker similarity scores. The system's acoustic front-end[4] extracts 20 mel-frequency cepstral coefficients (MFCCs) per frame using 20 filters, leading to 60 features per frame after including deltas and double-deltas. The chosen MFCC configuration is commonly used in speaker recognition experiments Alam et al. (2013); Dehak et al. (2011). The features are processed with RASTA filtering (Hermansky and Morgan, 1994) and cepstral mean and variance normalization (CMVN). Non-speech frames are omitted using energy-based speech activity detector (SAD) (described in Section 5.1 of (Kinnunen and Li, 2010)).

The universal background model (UBM), i-vector extractor, linear discriminant analyzer (LDA), and PLDA, are trained using Wall Street Journal (WSJ) and Librispeech corpora. LDA is used to reduce 400-dimensional i-vectors to 250 dimensions before centering, whitening, and length normalization. Simplified PLDA with 200-dimensional

[4] http://cs.joensuu.fi/~sahid/codes/AntiSpoofing_Features.zip

speaker subspace is used for scoring. For further details, refer to Table 1 of the current work and Section 2.2 of Kinnunen et al. (2019).

3.3. Attacked ASV system

In our experiments, we regard the x-vector system (Snyder et al., 2018), based on pre-trained Kaldi (Povey et al., 2011) recipe, as the ASV system to be attacked. To emulate the scenario of attacker's limited knowledge of this system, the attacker's ASV is made intentionally different from the attacked ASV system in terms of feature extractor set-up, embedding type, and development corpora (Table 1). The attacked system is the Kaldi x-vector recipe for VoxCeleb, while the attacker's system uses i-vectors. Unlike the i-vector extractor, the x-vector extractor is trained discriminatively using speaker labels.

4. Corpus of target speakers: VoxCeleb

The attacker's ASV is used as a voice search tool to find the closest speakers from the combination of VoxCeleb1 (Nagrani et al., 2017) and Voxceleb2 (Chung et al., 2018) to each of the locally recruited subjects (described in Section 5). The combined VoxCeleb corpus contains about 1.3 million speech excerpts extracted from more than 170,000 YouTube videos from $J = 7,365$ unique speakers. This totals to about 2,800 hours of audio material, most of which is active speech. Both VoxCeleb corpora were collected using automated pipeline exploiting face verification and active speaker verification technologies (Chung et al., 2018).

VoxCeleb1 contains mostly English speech, while VoxCeleb2 is more diverse in nationalities and languages. The nationality information of the target speakers was of our interest, as the recruited local speakers are Finnish and we wanted to see if Finnish people do better job at imitating Finnish rather than non-Finnish targets. According to the VoxCeleb1 metadata, there are no Finnish speakers in VoxCeleb1. VoxCeleb2 did not include nationality metadata but we extracted the nationalities automatically using Google's *Knowledge Graph* API[5]. This way we identified a total of 44 Finnish speakers from VoxCeleb2.

5. Locally recruited attackers

5.1. Speakers and recording gear

We recruited $K = 6$ voluntary local speakers (4M + 2F) to serve as 'attackers'. The selected terminology, 'attacker', is made for convenience to reflect the focus of ASV vulnerability study; it should be understood that all speakers took part voluntarily and were not asked to 'hack' any computer systems in the sense understood in the security field. In fact, most of our speakers are considered *naive* to the study aims: two of the male subjects knew the specific goals of the study but the remaining four subjects were not informed that the text and target speakers were tailored for them, nor where the target voices were obtained from. The speakers were not informed that the study relates to ASV vulnerability, but were asked to mimic the target speakers as accurately as they could. All the subjects signed an informed consent form to use their speech data for research, and were rewarded with movie and coffee tickets.

All six attackers are native Finnish speakers with an age range between 24 to 44 years old. They are *naive* impersonators who lack formal training in mimicry. We adopt the same recording setup from (González Hautamäki et al., 2017) and text prompts are described in detail in Vestman et al. (2019). As illustrated in Fig. 1, the subjects took part to three recording sessions. The first session, produced in the subject's natural voice, is used for VoxCeleb target speaker selection, while the remaining two sessions serve for vulnerability analysis of the attacked systems. The tasks in the recording sessions differed, while the recording set-up was the same: recordings took place in a silent laboratory room with a portable Zoom H6 Handy Recorder using an omnidirectional headset mic (Glottal Enterprises M80) with 44.1 kHz sampling and 16-bit quantization. Three other channels (two smartphones and electroglottograph) were also collected, but are not used in this study.

[5] https://developers.google.com/knowledge-graph/

Table 2
Target speakers (closest, median and furthest) per attacker. Selection of potential targets from 44 Finnish celebrities in VoxCeleb2.

Attacker ID	Celebrity	Profession	Spoken language
M1	Samuli Edelmann	Actor, singer	Finnish, English
	Paavo Väyrynen	Politician	Finnish
	Antti Tuisku	Pop singer	Finnish
M2	Samuli Edelmann	Actor, singer	Finnish, English
	Paavo Väyrynen	Politician	Finnish
	Mika Kojonkoski	Ski jumper, politician	Finnish, English
M3	Joni Ortio	Ice hockey player	Finnish, English
	Elastinen	Rap musician	Finnish
	Perttu Kivilaakso	Musician	English
M4	Samuli Edelmann	Actor, singer	Finnish, English
	Tuomas Holopainen	Musician	Finnish, English
	Jyrki Katainen	Politician	Finnish, English
F1	Anna Puu	Pop singer	Finnish
	Karita Mattila	Opera singer	Finnish, English
	Tarja Halonen	Politician	Finnish, English
F2	Sofi Oksanen	Writer	Finnish, English
	Kaisa Mäkäräinen	Biathlete	Finnish, English
	Tarja Halonen	Politician	Finnish, English

5.2. *The first recording session (data for target search)*

The first session, used for the targeted VoxCeleb speaker search, consists of four tasks in the speaker’s natural voice. The tasks consisted of spontaneous speech and read text (13 sentences) in both Finnish and English. The read texts in Finnish are the same used in (González Hautamäki et al., 2017). Their corresponding English versions were added for this study. We have approximately six minutes of speech (before speech activity detection) per speaker from Session 1. Detailed description of the material used in data collection can be found in the online supplementary material Vestman et al. (2019).

5.3. *Attacked target speaker search and utterance selection*

For the purpose of targeted speaker search, we compute a single averaged i-vector for each of the six speakers resulting from 28 individual utterances from Session 1. Similar to (Lau et al., 2004), we use the ASV system to pick for each attacker the **closest, median**, and **furthest** speakers among the VoxCeleb speakers. The closest one is most relevant for vulnerability analysis while the other two serve for reference purposes. We do this ASV-assisted search separately for *all* the VoxCeleb speakers (unconstrained search from 7,365 speakers) and for the subset of 44 Finnish speakers. We pool all the speech data of the VoxCeleb speakers to compute average i-vector per target. The selected target speakers per attacker are presented in Tables 2 and 3.

In addition to the three ASV-selected targets, we include **common target** matched with the speaker’s gender, in both Finnish and English. The common Finnish speaking targets are Päivi Räsänen (female, politician) and Ilkka Kanerva (male, politician), and the common English speaking targets are Hillary R. Clinton (female, politician) and Leonardo DiCaprio (male, actor). The choice of the common targets is arbitrary but based on a loose, subjective criterion *as famous as possible*. We first identified a short-list of VoxCeleb celebrities that we thought are well-known. We then ran an e-mail survey among our friends and colleagues (23 responded), asking each one to indicate the three most famous persons (in their opinion). We combined their votes to select the common targets. Even if the selected targets are well-known, from the viewpoint of ASV they are *random* target speakers with no strong presuppositions how similar their voices are to our attackers.

In summary, for each of our four male and two female subjects, we select six customized targets (three ASV-ranks $\times$ two languages) and two common gender-matched ones (one Finnish, one English). This gives a theoretical total of $3 \times 2 \times 4$ male $+ 2$ common male $+ 3 \times 2 \times 2$ female $+ 2$ common female $= 40$ target speakers. But as the reader can see from Table 2, not all of the ASV-selected targets are unique: one Finnish male celebrity (Edelmann) was the closest target for three attackers, one Finnish male celebrity repeated as the median speaker for two male attackers (Väyrynen), and one Finnish female celebrity (Halonen) is the furthest speaker for both female

Table 3
English speaking celebrities (closest, median and furthest) per attacker. Selection from 7321 potential targets in VoxCeleb1 and VoxCeleb2. * indicates speakers from VoxCeleb1.

Attacker ID	Celebrity	Profession	Spoken language
M1	Valentin Inzko	Politician	English (Austrian)
	Elijah Cummings	Politician	American English
	Chris Colfer *	Actor	American English
M2	Jeremy Irons *	Actor	British English
	Karan Tacker	Actor	Indian English
	Ryan Ochoa *	Actor	American English
M3	Éric Boullier	F1 manager	English (French)
	Guillaume Canet *	Actor, director	English (French)
	Bill Gilman	Singer	American English
M4	Ciarán Hinds	Actor	Irish English
	Ian Kinsler	Baseball player	American English
	Phil Mickelson	Golf player	American English
F1	Jessie J *	Singer	British English
	Candace Cameron *	Actress	American English
	Lin Shaye *	Actress	American English
F2	Fay Ripley	Actress, author	American English
	Belcim Bilgin	Actress	English (Turkish)
	Anne Hathaway *	Actress	American English

attackers. These collisions might be explained by the the limited number of Finnish celebrities (30M, 14F) in Vox-Celeb. The total number of unique celebrity targets is 36.

For each of the 36 target speakers, we selected multiple short utterances so that, when combined, each target would have at minimum 30 seconds of active speech. The selected utterances were used to evaluate the ASV system attacks. We selected only short utterances for two reasons. First, the duration of most of the VoxCeleb excerpts varies between five to ten seconds. Second, we deemed shorter utterances to be easier for our attackers to imitate. Detailed description of these utterances is provided in an online supplementary material Vestman et al. (2019).

The selection of the VoxCeleb excerpts was done by utilizing attacker's ASV system. For the closest and furthest targets we selected, respectively, the highest and lowest scoring utterances. For the median speakers, we selected the utterances closest to the mean. This was further accompanied by manual inspection: if the audio quality (determined subjectively by listening) in a given utterance was not deemed high enough, we discarded it and moved on to the next ones in the ranked list.

5.4. Speech transcription and the mimicry recordings

Unlike the first recording session (common to all subjects), the second and third sessions were tailored for each subject. This process involved the use of speech transcripts of the selected target utterances. To this end, we used Amazon's Mechanical Turk[6] (MTurk), a commercial crowdsourcing service, to transcribe the English language audio. The Finnish transcripts were produced by two native Finnish speakers. The 35 MTurk crowdworkers and the two Finnish transcribers were asked to transcribe all the nuances of conversational speech, including repetitions, hesitations, filler words *etc*. Finally, two reviewers audited the quality of all the transcripts. All the final transcriptions are provided in the supplementary material Vestman et al. (2019).

In Session 2, which took place five to six weeks after Session 1, the subject was provided with the transcripts of the selected target utterance(s) and was asked to read the sentences twice in his or her natural voice. The speaker was not informed whose speech the transcripts corresponded to. The rationale of including this session was to familiarize each attacker with the target speaker sentences. We adopted the general idea to include a session with reference text only and another one with audio from the design used in (Mariéthoz and Bengio, 2005). In that study, the target speakers were public personalities that each impersonator knew. Each impersonator completed three scenarios with an increasing level of detail about the target speakers. The impersonator was first asked to produce prototypical target speech without knowledge of text (other than common category, *e.g.* everyday sentences). The impersonator

[6] https://www.mturk.com/

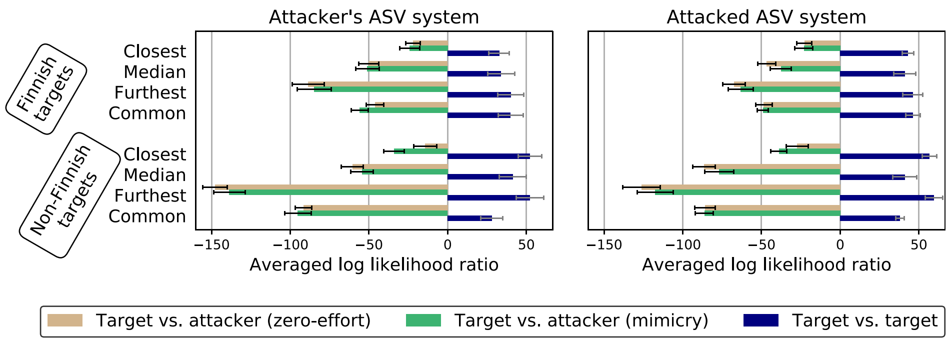


Fig. 2. Comparison of attackers' ASV scores (log likelihood ratios) to the targets' scores for both of the ASV systems involved in the study. The scores are averaged over all attackers and all speech segments. The error bars represent 95 % confidence intervals for the means.

was then revealed the target speaker texts to be impersonated and, finally, he would be provided audio reference of target.

In the last session, which took place two to six days after Session 2, the subjects were provided with the same transcript as in Session 2. Additionally, they were now provided access to the actual target speaker audio excerpts. The transcripts were provided on a printed paper and the audio was presented through headphones connected to a tablet computer with an interactive webpage. The subject was allowed to interact with the audio samples and could listen to the target utterance(s) as many times as needed, and he/she then tried to mimic the voice according to their best skills. Again, the subject was asked to mimic each sentence twice. In the experiments, we use only the second recording of each sentence.

Following standard convention in the context of spoofing and countermeasure studies Wu et al. (2015a), we refer to the speech recordings of the second session as *zero-effort*. This is to signify that the attackers were instructed to produce target speaker texts in their own modal voice, *i.e.* without dedicated effort to sound like the target. The recordings from the last session, in turn, are simply referred to as *mimicry* utterances.

6. Results: mimicry attacks against automatic verification system

In the following, we evaluate the effectiveness of mimicry attacks against ASV systems. The target speaker models used in the experiments were enrolled using all available segments except those selected for testing as described in Section 5.3.

Figure 2 displays how the PLDA scores of genuine and attack trials compare to each other. The general findings are as expected. First, the order of the closest, the median, and the furthest speakers transfers from the attacker's ASV system to the attacked ASV system, implying that the ASV-assisted speaker selection *can* help in ASV attacks. Second, in general, the attackers' natural and mimicry scores are significantly (by a wide margin) below the target scores. Additionally, we find no significant difference between the zero-effort and mimicry attacks (except for the closest category). Finally, as the recruited attackers are Finnish, attackers' scores against the Finnish targets are higher than for the non-Finnish targets (within each rank category).

We further display the difference of mimicked and natural speech scores in Table 4. Interestingly, and contradictory to what we assumed, if the target speaker's voice is already close to the attacker's voice, the impersonation attempts *degrade* the score. The same finding was noted in situations where the target is a well known public figure (as the targets in the common category are). We suspect that the effect might be due to people having higher tendency to *overact* someone they already know well. However, if the targets are not close to the attackers (*i.e.*, median and furthest categories) or are less well known, impersonation is potentially helpful (though, not by a statistically significant margin).

Our attackers are native Finnish speakers recorded with a specific set-up which may differ from the target domain (VoxCeleb) conditions. This raises a question whether our mimicry attacks might have been unsuccessful due to *domain mismatch*. To address this question, we studied *target-domain, attacker-domain*, and *cross-domain* non-target score distributions as well as target-domain and attacker-domain target score distributions. It was not possible to

Table 4
Score differences between attacks with impersonated voices and attacks with natural voices. Differences are averaged over attackers, target nationalities, and utterances. ± indicates 95 % confidence intervals. In the case of the closest target speakers, impersonation attempts are counterproductive.

ASV system	Closest	Median	Furthest	Common
Attacker's ASV	−9.7 ± 5.2	2.2 ± 4.3	5.9 ± 7.1	−7.2 ± 4.3
Attacked ASV	−5.2 ± 3.9	9.2 ± 3.3	6.1 ± 4.3	−0.5 ± 3.8

construct cross-domain target trials as we do not have speakers common to both domains. The main interest in this specific study is to compare target-domain non-target scores to cross-domain non-target scores. If the cross-domain scores (the case of attacks) do not fall below the target-domain scores, it suggests that the attacker does not get penalized by the domain mismatch. The scores for the study were obtained from the attacked x-vector based ASV system.

Figure 3 indicates that when the nationality mismatch is present (non-Finnish target-domain speakers), the cross-domain non-target scores are, on average, slightly lower than the the target-domain non-target scores. If, however, the target-domain speakers are Finnish, like our recruited attackers are, the non-target speaker distributions overlap almost perfectly. This suggests that the Finnish attackers attacking the Finnish VoxCeleb targets did not seem to get penalized by the domain mismatch. The domain mismatch can be observed by comparing target and non-target scores of attacker-domain and target-domain. As the attacker-domain is has much less variability in the conditions, the scores in attacker-domain tend to be higher.

7. Perceptual evaluation of mimicry attacks

Next, we evaluated how ASV assisted mimicry attacks perform against human listeners. Further, we compared the findings of perceptual test to those obtained from the attacks against the ASV system. To avoid nationality mismatch between targets and attackers, we restricted our experiments to Finnish targets only.

7.1. Listening test setup

In total, we had 625 pairs of speech samples (trials) to be evaluated by the listeners. These trials can be divided into five groups of 125 trials (4 to 7 trials for each of the 24 attacker-target combinations). The first three groups are related to the mimicry attacks: 1) target vs. target (reference point), 2) target vs. attacker (zero-effort mimicry), and 3) target vs. attacker (mimicry). For each set of three trials, the same target enrollment utterance is used. The speech content of the test utterances is the same in all three cases, but different from that of the enrollment utterance (*i.e.* text-independent speaker comparison). The two last types of trials focus on the attacker. They are 4) attacker (zero-effort) vs. attacker (zero-effort) and 5) attacker (zero-effort) vs. attacker (mimicry). These two cases are included, respectively, to study the listeners' performance for the same-speaker trials with fixed recording conditions, and to study how much the attackers modify their voices relative to their natural voices when mimicking. In the cases 4) and 5), the enrollment utterances are selected from the English part of the data described in Section 5.2. Similarly as

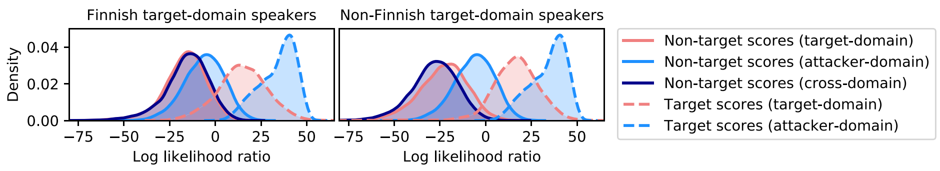


Fig. 3. Distributions of target and non-target scores in different domains. *Cross-domain* non-target scores are obtained by scoring speakers from the attacker domain against the speakers from the target (VoxCeleb) domain. The simulated mimicry attacks in this work fall under the category of cross-domain trials. As the cross-domain score distributions overlap almost perfectly with the target-domain non-target distributions, the domain mismatch does not seem to make attacking more difficult, at least when the targets are Finnish.

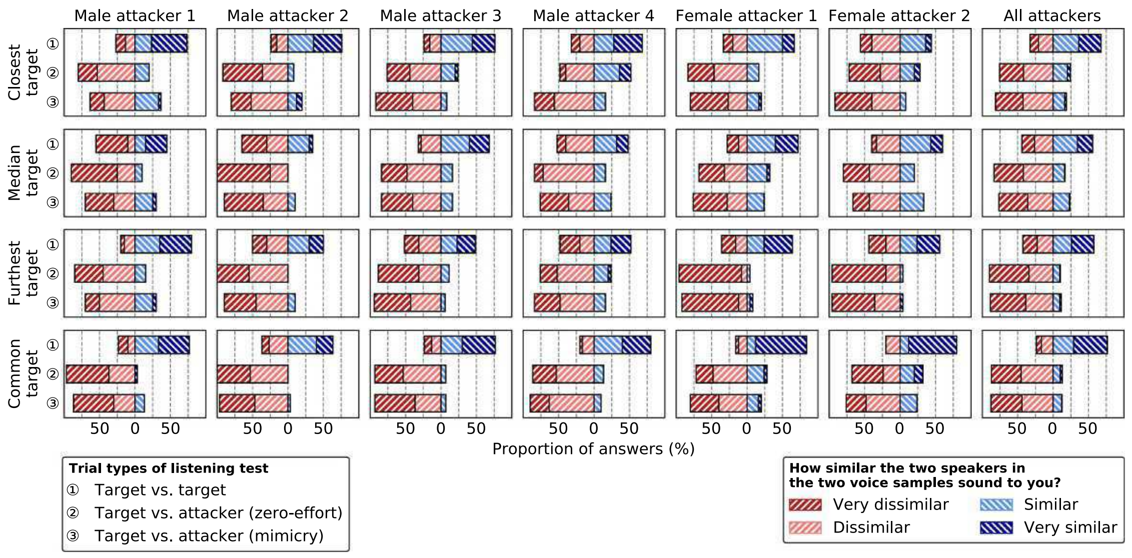


Fig. 4. Results from the listening test (target speaker enrollment vs. test segment). Each attacker (in columns) has 4 targets speakers (in rows: closest, median, furthest, common). For each attacker-target combination, there are three different trial types (denoted by circled digits) as described in the left-hand side legend. The last column shows the results when trials from all the attackers are combined.

above, for each set of two trials, the enrollment utterance is fixed and the two test utterances have the same content. In all of the cases, the enrollment utterance was selected from the available utterances so that its duration is close to the duration of the test utterances.

The listening trials were accompanied with a question *"How similar the two speakers in the two voice samples sound to you?"*, to which the listeners answered using a 4-point scale with options *Very dissimilar, Dissimilar, Similar*, and *Very similar*. The 4-point scale was selected to enforce the listeners to make up their mind regarding speaker similarity. When presenting the trials, the order of the two voice samples in a trial was randomized so that the enrollment utterance was not always played the first. Each trial was presented individually and their order was randomized as well. For each of the 625 trials, we asked opinions from five different listeners, so in total we collected 3125 responses from the listeners.

We recruited the listeners using the Amazon's MTurk service. All the listeners were either native English speakers or had advanced English skills. In total, 225 crowdworkers participated the listening trials. Five workers rated more than 100 trials, whereas 130 completed less than five. On average, a crowdworker completed 3125/225 ≈ 14 trials. Out of the 225 listeners, 40 provided information about their mother tongue: 26 English, 4 Italian, 4 Portuguese, 2 German, 2 Spanish, 1 Estonian, 1 Tamil.

7.2. Listening test results

We present the main results of the listening test in Figure 4, which presents the listener judgements of speaker similarity for all the studied attacker-target combinations. First, the listeners regard the two samples from the same target speaker (target vs. target cases) similar or very similar to each other, as expected. However, there are individual cases that turned out to be difficult for the listeners. For example, the median target of the male attacker 1 was considered dissimilar or very dissimilar sounding to himself in most of the answers. Informal listening of the utterances of this target revealed that the target's voice sounded different each time mostly due to differences in speaking style, recording conditions, and audio processing. For example, in one sample, the target speaker (Finnish politician) is being interviewed in a talk show, whereas in another sample he is giving a public speech in very different conditions.

How are the listeners opinions affected by mimicry? On average (see the last column of Figure 4), mimicry does not seem to help to make the attackers sound more like the targets. At the individual level, we find, however, that

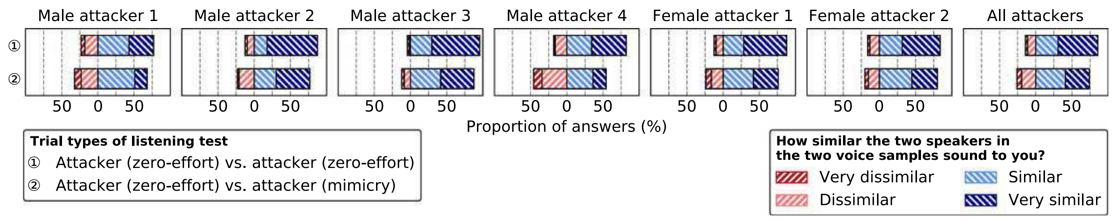


Fig. 5. Results from the listening test (attacker enrollment vs. attacker test segment). Listeners evaluate each attacker's enrollment samples against attacker's zero-effort and mimicry-effort attack samples. The voice modification induced by mimicry attempt makes the attackers sound less like themselves.

male attackers 1 and 2 got higher ratings for their mimicked speech. Further, we find that ASV assisted target speaker selection can help in choosing attacker-target pairs that sound similar to each other. That is, the furthest targets get lower similarity ratings than the closest targets. Even if automatic systems and humans based their speaker similarity judgments differently, the broad rank categories seem consistent.

Figure 5 displays listening test results for those trial types where attacker's enrollment utterances are compared to attacker's test segments with and without mimicry effort. The same-speaker trials have higher similarity ratings in comparison to those in Figure 4). This is expected since our attacker corpus is practically free from channel variation and background noise unlike the VoxCeleb collections. In addition, we find that when the attackers are trying to mimic the voices of the target speakers, they sound a little bit less like themselves.

7.3. *Comparison of human listeners and automatic speaker verification system*

To compare human opinions to ASV system scores, we scored the same trials using both the attacker's ASV system and the attacked ASV system. All the individual scores for three different trial types are displayed in Figure 6. The scores for the content matching test utterances are connected with lines and thus form score-triplets. This allows us to see how close the attacker's scores are to the target's scores and how successful were the mimicry attempts in individual cases. The results agree with the results of Figure 2, as expected — the only difference with the earlier ASV protocol is the number of target speaker enrollment utterances, which is now only one[7].

In general, the findings from the listening test are similar to what the ASV system scores imply. The ASV-assisted target speaker selection helps to bring attacker's scores closer to the target's scores, while the mimicry attempts do not seem to help much to bring the scores closer to the target's scores.

8. Prosody and formant analysis of mimicry attacks

To gain further insight how attackers' change their voices to mimic their targets, we carried out a study of the changes in fundamental frequency (F0), speaking rate, and formants. Our main motivation to study these qualities is to see whether attackers changed more their prosody than spectral cues. If this is the case, the changes might not be reflected by ASV scores as our systems are based on spectral features.

8.1. *Estimation of fundamental frequency and speech rate*

Speaking rate, in terms of syllable rate (the number of syllables per second), was measured using a Praat (Boersma and Weenink, 2015) implementation (De Jong and Wempe, 2009) that automatically calculates the number of syllables per sample duration by detecting syllable nuclei (Wang and Narayanan, 2007) and pause duration. As for F0 extraction, we adopt an autocorrelation-based method (Boersma, 1993) implemented in Praat. We use gender-specific frequency ranges set to [75, 200] Hz for males and [100, 300] Hz for females. We initially tested

[7] In general, data processing capacity of ASV systems and listeners differ: ASV systems can process multiple enrollment utterances and large number of trials, but humans have limited attention span and memory and cannot process many trials (or excessively long utterances). For the maximum benefit of the ASV system, the earlier ASV protocol used in Fig. 2 used multiple enrollment utterances, while the scaled-down ASV protocol (single enrollment utterance) used in Fig. 6 was designed to facilitate perceptual speaker comparisons.

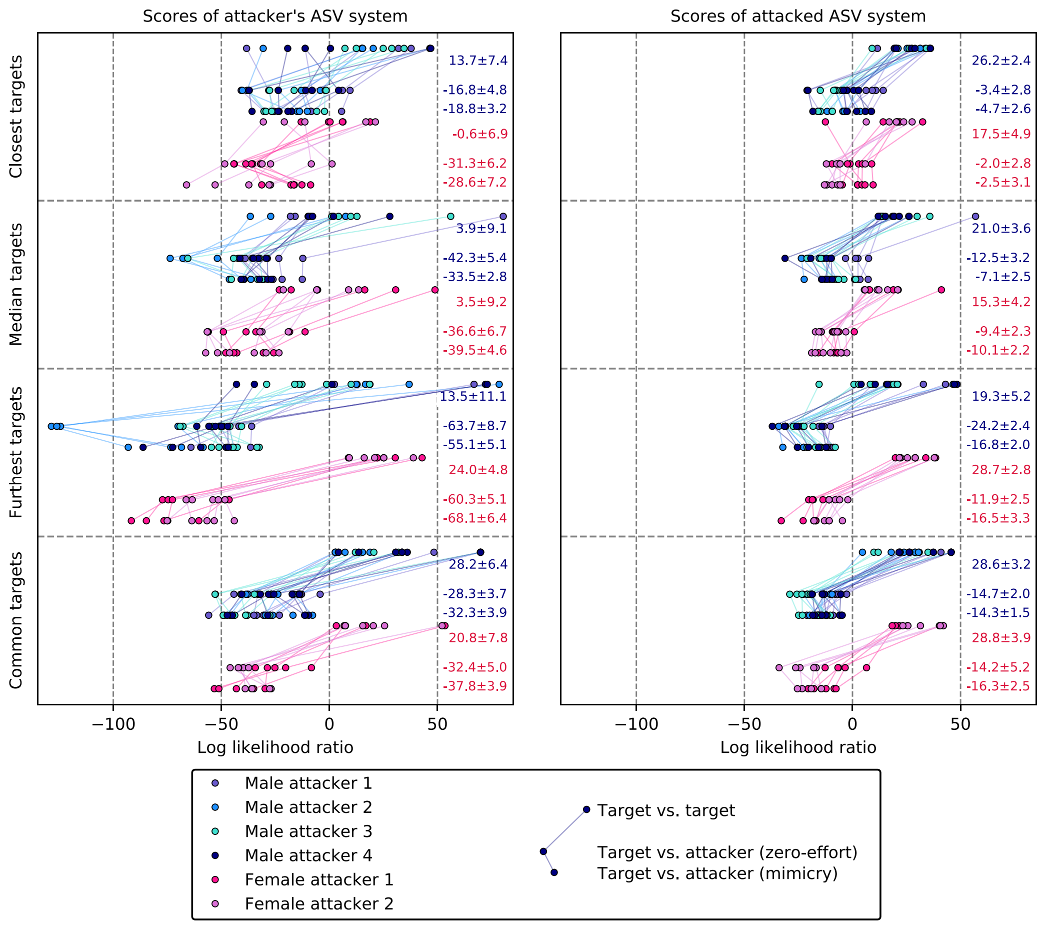


Fig. 6. The scores of the ASV systems for the trials used in the listening test. The scores in each score triplet (described in the legend) are from the trials that have the same target speaker enrollment utterance and the speech content is the same in all the three test segments. Scores for male and female attackers are shown in separate groups. The right side of each graph displays the mean values of the score groups together with standard error of the mean multiplied by 1.96.

F0 extraction with wider F0 ranges but it was observed that the selected ranges were appropriate to exclude possible tracking errors and outliers in the F0 contour. The parameters to select the F0 candidates at 10ms intervals were set at their default values in Praat: silence threshold 0.03, voicing threshold 0.45, octave cost per octave 0.01, octave-jump cost 0.35, and voiced-unvoiced transition cost 0.14.

We summarize F0 values of each utterance using two summary statistics, namely, median and standard deviation. They reflect, respectively, the average pitch range and pitch dynamics within a given utterance. We study changes in these summary statistics between the zero-effort and mimicry attempts, with the aim of studying whether or not our attackers attempt to match their broad prosody characteristics with those of their targets upon their mimicry attempts.

8.2. *Estimation and alignment of formant frequencies*

We performed formant analysis by comparing formant information of aligned utterances. First, we extracted formant center frequencies of the first three formants (F1, F2, and F3) using VoiceSauce Shue et al. (2011) with Praat backend. Next, we aligned attacker's utterances (natural & mimicry) with target's utterance using *dynamic time*

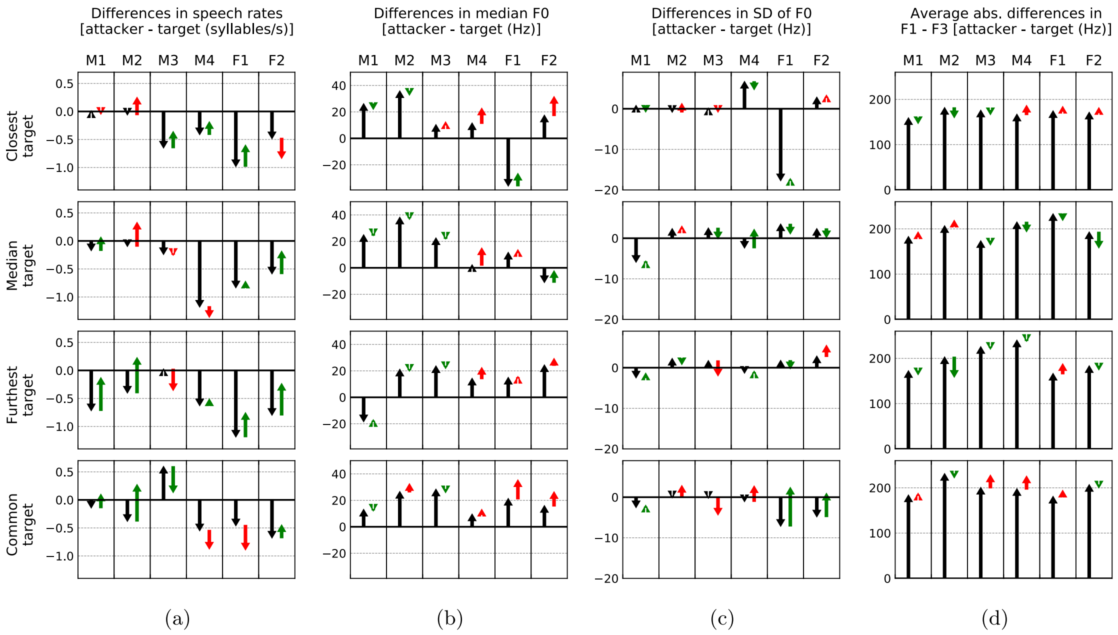


Fig. 7. Differences of attacker's (M1, M2, M3, M4, F1, F2) prosodic and formant parameters to target's parameters for all attacker-target combinations. Differences are shown for non-effort speech (**black arrow**) and for mimicked speech. The effect of mimicry is displayed with a **green arrow** if it made attacker's and target's parameters closer to each other and with a **red arrow** otherwise.

warping (DTW) Sakoe and Chiba (1978). The aligning process was done similarly as in Vestman et al. (2018). This process involves using automatic selection of active speech frames that are well aligned and have reliable formant information. The alignment of utterances turned out to be challenging due to differences in speaking styles, acoustic conditions, and small deviations in spoken texts caused by mumbling. Thus, in addition to the automatic frame selection, we listened the aligned utterances in order to discard the the badly misaligned ones. Finally, after getting the aligned formant data, we measured the formant difference d between utterances a and b as

$$d(a,b) = \frac{1}{3T}\sum_{t=1}^{T}\sum_{n=1}^{3}\left|f_a(t,n) - f_b(t,n)\right|, \tag{2}$$

where T is the number of aligned frames and $f_a(t, n)$ is the center frequency of formant n of utterance a at frame t.

8.3. Results of prosody and formant analysis

In Figure 7a, we show the results for the analysis of speech rate differences. For each attacker-target combination, the displayed speech rates are obtained by averaging the speech rates of the available utterances (4 to 7 utterances per combination). The results indicate that the speech rates of the attackers were, in general, slower than the targets' speech rates, when the attackers were not mimicking. This was anticipated, since the attackers were reading prompted text from a paper yielding slower speaking rates as opposed to those of the targets samples obtained from conversational situations. After listening to target's speech, the attackers were in most cases able to change their speech rates towards the targets' speech rates. At the individual level, we find that the male attacker 1 (M1) was good at adjusting his speech rate, while the male attacker 3 (M3) had naturally fast reading pace so that in some cases (common target) his speech rate was already too fast.

A similar comparison regarding F0 statistics is shown in Figures 7b and 7c. We find that the attackers M1, M2, and M3 did not change their F0 considerably while mimicking, whereas attackers M4, F1, and F2 had some mimicry attempts with clearly different F0 than what their natural F0 is. We do not observe clear differences between closest, median, and furthest target categories in terms of distances in F0 parameters between attackers and targets.

Finally, in Figure 7d, we depict the formant differences between targets and attackers as defined in (2). Again, we find that the mimicking did not have major impact to the similarity of the formant frequencies. In 14 out of 24 cases, mimickers managed to get slightly closer to their targets in terms of the given metric. We further find that the formant differences are larger in the furthest category than in the closest category, which is expected as the location of formants affect the spectral features used in the target speaker selection.

9. Conclusion

Biometric data uploaded to the Internet in large quantities, including human voice samples, opens up potential for misuse whenever the same biometric identifiers are adopted for strong user authentication to regulate access to personal data records, bank accounts and other services. Our study addressed a potential risk related to combination of public-domain automatic speaker verification (ASV) technology and public-domain voice data. The former is used as a search tool to identify potential target speakers to be mimicked.

Our results suggest that human mimicry is a rather special skill and less effective in spoofing modern ASV systems compared to voice conversion, text-to-speech, and replay. In specific, none of our six attackers received high detection scores for their attacks from our simulated[8] public-domain or attacked ASV systems. Similar negative findings have been reported in earlier studies and are often speculated to be due to difficulty of humans to mimic accurately low-level spectral cues employed by ASV systems. One of our motivations was to *re-assess* whether speech mimicry — one of the weakest known attacks against ASV — might be made substantially stronger (or more practical) when the target speakers are selected using ASV. We approached this question from two perspectives. On the one hand, we wanted to find out how the score ranges associated with broad target speaker rank (closest, median, further) transfer from the attacker's ASV to the attacked ASV. This is the *technology* dimension of our attack model. On the other hand, we wanted to isolate the effect of the mimicry effort by collecting attackers' voice samples both 'before' (zero-effort attack) and 'after' (mimicry attack) listening to the target speaker's voice. This allows us to analyze the changes in attacker-to-target log-likelihood ratio (LLR) scores due to mimicry effect alone. This is the *human* dimension of our attack model.

Concerning the broad target speaker rank, the score relations generalize well from the attacker's ASV system to the attacked ASV system: LLR(closest target) $>$ LLR(median target) $>$ LLR(furthest target) relationship was retained both for Finnish and non-Finnish targets. This suggests that one could, indeed, use one ASV system (here, i-vector PLDA) to emulate the broad speaker ranking of another, targeted ASV system (here, x-vector PLDA). We find this result interesting and worthwhile of future work. Even if the VoxCeleb corpora are among the largest (public) speaker corpora at this time, they are still tiny compared to the number of voice samples in the Internet. It would be interesting to repeat a similar study design to ours in a few years, perhaps with an order of magnitude larger target speaker corpus and, at this stage, unforeseen ASV technology. It would be important to uncover the conditions under which such emulation succeeds (or fails). With an increasing number of video and voice samples posted online, it is not only the security, but user privacy, that deserves attention.

Concerning the impact of mimicry effort, the attacker-to-target LLRs remained low, and substantially below the target-to-target LLRs in both zero-effort and mimicry scenarios. Curiously, while the LLR scores for the furthest target speakers indicated some increase between zero-effort and mimicry scenarios, for the closest targets the LLR scores *decreased* (but significantly only for the non-Finnish target speakers). To sum up, the broad target speaker rank generalized across the ASV systems, while the mimicry effect itself lead to negative (or no difference) effect. These findings reinforce the conjecture that voice mimicry by itself may not pose a strong attack against ASV; but ASV-based target speaker selection may.

We hypothesized that while our attackers' mimicry efforts did not have major impact on the ASV scores, they might have impact on human perception. Human listeners might, to some degree, focus on different cues of speaker identity than the ASV systems, which mostly focus on spectral characteristics of speech. However, the results of our

[8] The ASV implementations combine scripts/tools (*e.g.* MSR Identity Toolkit, Kaldi) that are all public-domain code. They should be considered as proxies of modern ASV technology, rather than end-user software.

listening test did not support the above hypothesis, as the results showed similar patterns to those we saw from the ASV scores.

So as to understand better the mimicry strategies implemented by the attackers, we also analyzed changes in formant frequencies and prosody statistics (F0, speaking rate). Even if some attackers were able to adjust their average formant frequencies towards those of their target speakers, the relative change in attacker-to-target formant distance (from zero-effort to mimicry) was minor. Adjustments in F0 statistics were minor as well. The most prominent adjustments towards the targets were seen in the speaking rate.

Our study has a number of limitations that one should take into account in future studies. First, the number of attackers (six) is admittedly small. This limitation, familiar to some of the authors González Hautamäki et al. (2015), is common to most speech mimicry studies and relates to difficulties in data collection. The number of attackers varies from 1 to half dozen (or so) Wu et al. (2015a). Here, additional complications were caused by tailored target speaker selection, involving tedious speech transcription and several stages of data quality auditing. In future work, it might be practical to drop the transcription step and ask the attackers to impersonate their targets based on audio only. Another way to scale up the study would be attacker recruitment through crowdsourcing Panjwani and Prakash (2014). This will, however, introduce new uncontrolled variations (such as attacker microphone differences). All our attacks were recorded using the same gear in the same room.

The second limitation relates to the cross-domain data conditions: our attackers are native Finnish speakers, while VoxCeleb consists of many different nationalities and accents. Further, VoxCeleb consists of conversational speech while our attackers read text passages in an office environment. These differences induce style differences and might make the impersonation task harder for the attackers. This limitation is primarily due to lack of large Finnish celebrity corpus at the authors' exposure, as well as our preference to interact with the attackers conveniently. It would be interesting to repeat selected experiments using a larger target speaker corpus with matched mother tongue. In VoxCeleb, we are limited to 44 Finnish target speakers. Future work could therefore either adopt a larger Finnish celebrity corpus, or to recruit native American English attackers. Given the nature of *found data*, controlling all the variations will be difficult.

Our attacks could also be made stronger in a number of ways. First, the attacker might use the public-domain ASV system in a more proactive way, such as optimizing its detection accuracy further in off-line experiments. Second, the attacker could potentially utilize more detailed feedback from a dedicated ASV system — in this work, attackers used ASV for speaker *ranking* while some prior work has used ASV score as a feedback signal Zetterholm et al. (2004). Third, assuming there would be an actual monetary (or other strong) motivator to seriously mimic someone — similar to practicing to forge someone's signature — the attacker might use substantially more effort to get familiar with the speaking style of his or her targets. He or she might perhaps use feedback from prosody measurements in addition to ASV score. In our study, given the extensive work required to prepare the tailored targets and collect the data, all the above had to be relaxed to complete recordings in a reasonable time. The mimicry attacks (with audio reference of the target) took place in a single session and our attackers completed their mimicry tasks relatively fast. Nonetheless, in future work it would be interesting to evaluate whether mimicry attacks could be improved with further, and more proactive, training. Another interesting target would be studying combination of automatic target speaker selection with voice conversion (or other technical) spoofing attacks.

It would be also interesting to address whether, and how, one may benefit from current (or suitably modified) ASV methods to provide intuitive feedback to improve one's mimicry skills. This would be potentially helpful in suggesting specific articulatory or voice source modifications required to increase the ASV score. The present study was framed to the context of ASV attacks but such methods could be potentially useful for mimicry artists, voice actors, and language learners as well.

Acknowledgement

The work has been supported by Academy of Finland (proj. no. 309629 entitled "NOTCH: NOn-cooperaTive speaker CHaracterization") and by the Doctoral Programme in Science, Technology and Computing (SCITECO) of the UEF. A part of the work of the first author was supported by NEC internship program. The work of Md Sahidullah was made with the support of Region Grand Est. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.csl.2019.05.005.

References

GDPR, 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (General Data Protection Regulation). OJ L 119 (1), 1–88. http://data.europa.eu/eli/reg/2016/679/oj.

Alam, M.J., Kinnunen, T., Kenny, P., Ouellet, P., O'Shaughnessy, D., 2013. Multitaper MFCC and PLP features for speaker verification using i-vectors. Speech Communication 55 (2), 237–251.

Biggio, B., Roli, F., 2018. Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition 84, 317–331.

Boersma, P., 1993. Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In: Proc. of the Institute of Phonetic Sciences, 17, pp. 97–110.

Boersma, P., Weenink, D., 2015. Praat: doing phonetics by computer [Computer program]. Version 5.4.09, retrieved 15 June 2015 from http://www.praat.org/.

Chesney, R., Citron, D.K., 2018. Deep fakes: A looming challenge for privacy, democracy, and national security. 07 California Law Review (2019, Forthcoming); U. of Texas Law, Public Law Research Paper No. 692; U. of Maryland Legal Studies Research Paper No. 2018−21.

Chung, J., Nagrani, A., Zisserman, A., 2018. VoxCeleb2: Deep speaker recognition. In: Proc. INTERSPEECH, pp. 1086–1090. doi: 10.21437/Interspeech.2018-1929.

Chung, J.S., Jamaludin, A., Zisserman, A., 2017. You said that? In: Proc. British Machine Vision Conference.

De Jong, N.H., Wempe, T., 2009. Praat script to detect syllable nuclei and measure speech rate automatically. Behavior Research Methods 41 (2), 385–390.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P., 2011. Front-end factor analysis for speaker verification. IEEE Trans. on Audio, Speech, and Language Processing 19 (4), 788–798. doi: 10.1109/TASL.2010.2064307.

Endres, W., Bambach, W., Flösser, G., 1971. Voice spectrograms as a function of age, voice disguise, and voice imitation. The Journal of the Acoustical Society of America 49 (6), 1842–1848.

Ergunay, S., Khoury, E., Lazaridis, A., Marcel, S., 2015. On the vulnerability of speaker verification to realistic voice spoofing. In: Proc. Seventh International Conference on Biometrics Theory, Applications and Systems (BTAS). IEEE, pp. 1–6.

Eriksson, A., 2010. The disguised voice: imitating accents or speech styles and impersonating individuals. In: Llamas, C., Watt, D. (Eds.), Language and Identities. 8, Edinburgh University Press, pp. 86–96.

Farrús, M., 2018. Voice disguise in automatic speaker recognition. ACM Comput. Surv. 51 (4), 68:1–68:22. doi: 10.1145/3195832.

González Hautamäki, R., Kinnunen, T., Hautamäki, V., Laukkanen, A.-M., 2015. Automatic versus human speaker verification: The case of voice mimicry. Speech Communication 72, 13–31. doi: 10.1016/j.specom.2015.05.002.

González Hautamäki, R., Sahidullah, M., Hautamäki, V., Kinnunen, T., 2017. Acoustical and perceptual study of voice disguise by age modification in speaker verification. Speech Communication 95, 1–15. doi: 10.1016/j.specom.2017.10.002.

Hermansky, H., Morgan, N., 1994. RASTA processing of speech. IEEE Trans. Speech and Audio Processing 2 (4), 578–589.

Intelligent Voice, 2019. Who do you sound like? https://celebsoundalike.com/

ISO/IEC 30107-1:2016, 2016. Information technology − Biometric presentation attack detection − part 1: Framework. https://www.iso.org/obp/ui/#iso:std:iso-iec:30107:-1:ed-1:v1:en, Online; accessed 22-February-2018.

Khan, H., Hengartner, U., Vogel, D., 2016. Targeted mimicry attacks on touch input based implicit authentication schemes. In: Proc. of the 14th Annual International Conference on Mobile Systems, Applications, and Services. ACM, pp. 387–398.

Kinnunen, T., González Hautamäki, R., Vestman, V., Sahidullah, M., 2019. Can we use speaker recognition technology to attack itself? enhancing mimicry attacks using automatic target speaker selection. In: Proc. ICASSP. IEEE, pp. 6146–6150.

Kinnunen, T., Li, H., 2010. An overview of text-independent speaker recognition: From features to supervectors. Speech Communication 52 (1), 12–40.

Lau, Y., D.T., Wagner, M., 2005. Testing voice mimicry with the YOHO speaker verification corpus. In: Proc. 9th Int. Conf. Knowledge-Based Intelligent Information and Engineering Systems KES, Part IV. Melbourne, Australia, pp. 15–21.

Lau, Y., Wagner, M., Tran, D., 2004. Vulnerability of speaker verification to voice mimicking. In: Proc. Int. Symp on Intelligent Multimedia, Video & Speech Processing (ISIMP'2004), Hong Kong, pp. 145–148.

Liu, M.-Y., Breuel, T., Kautz, J., 2017. Unsupervised image-to-image translation networks. In: Proc. Advances in Neural Information Processing Systems, pp. 700–708.

Lorenzo-Trueba, J., Fang, F., Wang, X., Echizen, I., Yamagishi, J., Kinnunen, T., 2018. Can we steal your vocal identity from the internet? initial investigation of cloning Obama's voice using GAN, WaveNet and low-quality found data. In: Proc. Odyssey 2018: The Speaker and Language Recognition Workshop, pp. 240–247.

Lorenzo-Trueba, J., Yamagishi, J., Toda, T., Saito, D., Villavicencio, F., Kinnunen, T., Ling, Z., 2018. The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods. In: Proc. Odyssey: the Speaker and Language Recognition Workshop, pp. 195–202. Les Sables d'Olonne, France.

Luck, J., 1969. Automatic speaker verification using cepstral measurements. The Journal of the Acoustic Society of America 46 (4), 1026–1032.

Mariéthoz, J., Bengio, S., 2005. Can a Professional Imitator Fool a GMM-Based Speaker Verification System? Idiap-RR. IDIAP.

Nagrani, A., Chung, J., Zisserman, A., 2017. VoxCeleb: A large-scale speaker identification dataset. In: Proc. INTERSPEECH, pp. 2616–2620. doi: 10.21437/Interspeech.2017-950.

Panjwani, S., Prakash, A., 2014. Crowdsourcing attacks on biometric systems. In: Proc. Tenth Symposium on Usable Privacy and Security, SOUPS 2014, pp. 257–269.

Papernot, N., McDaniel, P.D., Goodfellow, I.J., Jha, S., Celik, Z.B., Swami, A., 2017. Practical black-box attacks against machine learning. In: Proc. ACM on Asia Conf. on Computer and Comm. Security, AsiaCCS 2017. Abu Dhabi, UAE, pp. 506–519.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K., 2011. The kaldi speech recognition toolkit. In: Proc. IEEE ASRU, pp. 1–4.

Prince, S.J.D., Elder, J.H., 2007. Probabilistic linear discriminant analysis for inferences about identity. In: Proc. Eleventh IEEE International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007, pp. 1–8. doi: 10.1109/ICCV.2007.4409052.

Ratha, N., Connell, J., Bolle, R., 2001. Enhancing security and privacy in biometrics-based authentication systems. IBM Systems Journal 40 (3), 614–634. doi: 10.1147/sj.403.0614.

Ribaric, S., Ariyaeeinia, A.M., Pavesic, N., 2016. De-identification for privacy protection in multimedia content: A survey. Sig. Proc.: Image Comm. 47, 131–151.

Sahidullah, M., Delgado, H., Todisco, M., Kinnunen, T., Evans, N., Yamagishi, J., Lee, K., 2018. Handbook of Biometric Anti-Spoofing: Presentation Attack Detection, Second Springer. Introduction to voice presentation attack detection and recent advances. Advances in Computer Vision and Pattern Recognition.

Sakoe, H., Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. on Acoustics, Speech, and Signal Processing 26 (1), 43–49.

Shue, Y.-L., Keating, P., Vicenik, C., Yu, K., 2011. VoiceSauce: A program for voice analysis. In: Proc. Seventeenth International Congress of Phonetic Sciences, pp. 1846–1849.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S., 2018. X-vectors: Robust DNN embeddings for speaker recognition. In: Proc. ICASSP. IEEE, pp. 5329–5333.

Suwajanakorn, S., Seitz, S.M., Kemelmacher-Shlizerman, I., 2017. Synthesizing Obama: Learning lip sync from audio. ACM Transactions on Graphics (TOG) 36 (4), 95.

Toda, T., Chen, L.-H., Saito, D., Villavicencio, F., Wester, M., Wu, Z., Yamagishi, J., 2016. The voice conversion challenge 2016. In: Proc. INTERSPEECH, pp. 1632–1636. doi: 10.21437/Interspeech.2016-1066.

Vestman, V., Gowda, D., Sahidullah, M., Alku, P., Kinnunen, T., 2018. Speaker recognition from whispered speech: A tutorial survey and an application of time-varying linear prediction. Speech Communication 99, 62–79.

Vestman, V., Kinnunen, T., González Hautamäki, R., Sahidullah, M., 2019. Voice mimicry attacks assisted by automatic speaker verification. Additional material. Published online.

Vestman, V., Soomro, B., Kanervisto, A., Hautamäki, V., Kinnunen, T., 2019. Who do I sound like? showcasing speaker recognition technology by YouTube voice search. In: Proc. ICASSP. IEEE, pp. 5781–5785.

Wang, D., Narayanan, S.S., 2007. Robust speech rate estimation for spontaneous speech. IEEE Trans. on Audio, Speech, and Language Processing 15 (8), 2190–2201.

Wu, Z., Evans, N., Kinnunen, T., Yamagishi, J., Alegre, F., Li, H., 2015. Spoofing and countermeasures for speaker verification: A survey. Speech Communication 66, 130–153.

Wu, Z., Kinnunen, T., N.E., Yamagishi, J., Hanilçi, C., Sahidullah, M., Sizov, A., 2015. ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge. In: Proc. INTERSPEECH, pp. 2037–2041.

Zetterholm, E., Blomberg, M., Elenius, D., 2004. A comparison between human perception and a speaker verification system score of a voice imitation. In: Proc. Tenth Australian International Conference on Speech Science & Technology. Macquarie University, Sydney, Australia, pp. 393–397.

Paper **VIII**

Computer Speech & Language 60 (2020) 101024

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

Voice biometrics security: Extrapolating false alarm rate via hierarchical Bayesian modeling of speaker verification scores

Alexey Sholokhov[a,b,c], Tomi Kinnunen[*,c], Ville Vestman[1,c], Kong Aik Lee[d]

[a] *St. Petersburg Electrotechnical University, ul. P. Popova 5, St. Petersburg 197376, Russia*
[b] *ITMO University, Kronverkskiy pr. 49, St. Petersburg 197101, Russia*
[c] *School of Computing, University of Eastern Finland, Joensuu FI-80101, Finland*
[d] *Biometrics Research Laboratories, NEC Corp., Tokyo, Japan*

A R T I C L E I N F O

Article History:
Received 6 May 2019
Revised 9 July 2019
Accepted 25 September 2019
Available online 3 October 2019

Keywords:
Speaker verification
Population size
Security
False alarm rate
Random impostor
Closest impostor
Bayesian score modeling
VoxCeleb

A B S T R A C T

How secure automatic speaker verification (ASV) technology is? More concretely, given a specific target speaker, how likely is it to find another person who gets falsely accepted as that target? This question may be addressed empirically by studying naturally confusable pairs of speakers within a large enough corpus. To this end, one might expect to find at least some speaker pairs that are indistinguishable from each other in terms of ASV. To a certain extent, such aim is mirrored in the standardized ASV evaluation benchmarks, for instance, the series of speaker recognition evaluation (SRE) organized by the National Institute of Standards and Technology (NIST). Nonetheless, arguably the number of speakers in such evaluation benchmarks represents only a small fraction of all possible human voices, making it challenging to extrapolate performance beyond a given corpus. Furthermore, the impostors used in performance evaluation are usually selected *randomly*. A potentially more meaningful definition of an impostor — at least in the context of security-driven ASV applications — would be *closest* (most confusable) other speaker to a given target.

We put forward a novel performance assessment framework to address both the inadequacy of the random-impostor evaluation model and the size limitation of evaluation corpora by addressing ASV security against closest impostors on arbitrarily large datasets. The framework allows one to make a prediction of the safety of given ASV technology, in its current state, for arbitrarily large speaker database size consisting of virtual (sampled) speakers. As a proof-of-concept, we analyze the performance of two state-of-the-art ASV systems, based on i-vector and x-vector speaker embeddings (as implemented in the popular Kaldi toolkit), on the recent VoxCeleb 1, and 2 corpora, containing a total of 7365 speakers. We fix the number of target speakers to 1000, and generate up to $N = 100{,}000$ virtual impostors sampled from the generative model. The model-based false alarm rates are in a reasonable agreement with empirical false alarm rates and, as predicted, increase substantially (values up to 98%) with $N = 100{,}000$ impostors. Neither the i-vector or x-vector system is immune to increased false alarm rate at increased impostor database size, as predicted by the model.

*Corresponding author.
E-mail addresses: sholokhovalexey@gmail.com (A. Sholokhov), tkinnu@cs.joensuu.fi (T. Kinnunen), vvestman@cs.uef.fi (V. Vestman), k-lee@ax.jp.nec.com (K.A. Lee).
[1] A part of the work of the third author was carried out during an intership at NEC.

https://doi.org/10.1016/j.csl.2019.101024
0885-2308/© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

Some have predicted that voice-operated user interfaces will be the next paradigm of human-machine interaction. Given that the consumer market already provides various *virtual assistants* — Google Home, Apple Siri, and Amazon Alexa to name a few — it might be a reasonable prediction. Such services are intended to provide human-to-human like user experience leveraging from speech and speaker recognition technology, dialogue modeling and speech synthesis. An increasing number of smart services also enable users to log-in or authenticate payments using voice (or other biometric traits), for both increased security and user convenience — there is no need to consult, *e.g.*, printed key-lists (or other stealable or copiable accessories). The co-evolution of smart device technology and machine learning (Bishop, 2006; Goodfellow et al., 2016) has substantially broadened the landscape of *automatic speaker verification* (ASV) (Reynolds, 1995) use cases from its traditional, highly specialized applications — forensics and surveillance — to our living rooms and everyday mobile environments. For instance, nowadays, smart phones, virtual assistants and other devices with powerful processors and wireless connectivity enable efficient on-device or cloud-based voice data processing, including ASV-based user authentication with algorithms that would have been difficult to execute on portable devices of the past decades. Early ASV technology, such as Reynolds (1995), was developed with the aid of far less powerful computers and smaller datasets. The increase in dataset sizes and computing power has not only enabled the research community to address increasingly more challenging ASV tasks, but enabled running more powerful models in portable devices. Much of the progress in the underlying core ASV technology has been facilitated by coordinated technology benchmarks, pioneered by *National Institute of Standards and Technology* (NIST) in their evaluation campaigns (Doddington et al., 2000; Sadjadi et al., 2017; Wu et al., 2017).

Increased awareness of the possibilities of voice-based interaction also raises concern about the security of the technology. The possibility to invoke malicious voice commands from a distance in another user's phone Carlini et al. (2016) (potentially even using *inaudible* sounds Zhang et al., 2017), and the potential to masquerade oneself as another targeted speaker through various *spoofing attacks* (Ratha et al., 2001) is widely acknowledged. The latter includes *replay, text-to-speech*, and *voice conversion* attacks. Many of these *technology*-aided attacks can be combated through various *countermeasures* ranging from knowledge-based approaches to classification approaches, known within biometric technology standardization bodies as *presentation attack detection* (PAD) (ISO/IEC 30107-1:2016, 2016) methods. For instance, specialized binary detector could be used to verify liveness of a voice sample before being passed to a speaker verification system. Detection of attacks is possible since replayed speech, introduced through loudspeakers, has different frequency characteristics than live human; and since synthetic and converted voices contain processing artifacts due to training data limitations and modeling imperfections. More details of different attacks, their effectiveness, detection, and evaluation metrics are discussed elsewhere (Sahidullah et al., 2018) in more detail. In this study, we focus on core ASV technology.

While recent efforts have capitalized the importance of preparing ASV systems against spoofing attacks, another, more fundamental question remains: how *unique* the human voice is? Note that even the performance of an ASV system equipped with *perfect* PAD subsystem will be upper bounded by the performance of the underlying core technology (Kinnunen et al., 2018). This raises fundamental, yet thus far conclusively unanswered questions such as,

- Given a large-enough population of speakers (such as 7.6 billion), how likely is it to find two speakers that are confusable with each other? In other words, *how many unique voices there are?*
- Conversely, assuming that we wish to maintain a certain minimum level of non-confusability between speakers, is there some maximum population (speaker database) size for which it can be guaranteed?

Answers would enable both technology vendors and users of ASV technology to have increased confidence to the expected reliability of such systems. By drawing analogy from the security of passwords, some studies (Nautsch et al., 2015) based on biometric information measures (Lim and Yuen, 2016) have assessed the strength of speech representations in terms of their speaker information, though the viewpoint is rarely neither on the population size nor attacks.

Before proceeding further, it is necessary to constrain the scope. First, the question of voice uniqueness is, clearly, ill-posed. *In theory*, the number of different human voices is, if not infinite, some very large number: both the organic (physiological) and learnt traits vary greatly across individuals thanks to differences in the anatomy and kinematics of our articulatory systems — it would be extremely unlikely to find another *voice clone* with perfectly-matched voice production systems and learned traits. *In practice*, when working with real-world acoustic speech waveforms, we are bounded both by *extrinsic* and *intrinsic* signal variations. Extrinsic variation refers to the inability to accurately measure 'pure' speaker characteristics from imperfect acoustic observations (for instance, due to imperfect transducer, lossy communication channel, background noise, or reverberant environment). Intrinsic variation, in turn, refers to linguistic and non-linguistic variation induced by the speaker him/herself, some of which can be substantial (Hansen et al., 2017; González Hautamäki et al., 2017). The main focus of the ASV research community for the past several decades (Reynolds, 1995; Kenny, 2010) has been on improving ASV technology to handle extrinsic variations of increased complexity, though specific intrinsic factors, such as *vocal effort*, have also been addressed in the context of NIST SREs (Greenberg et al., 2011).

Neither the extrinsic nor intrinsic variations are deterministic, fixed operations. Therefore, there are practical limits as to how accurately one can discriminate two voices from each other. As these limits are clearly a function of the specific types of variations and distortions (as the ASV community is well aware of), it would be meaningless to attempt to answer the unconditional question of voice uniqueness. The answer depends on both data conditions and the employed hypothesis tester (e.g., a specific human listener or a specific ASV system). We might even say that uniqueness of voices is a *subjective* matter; a pair of speakers that is confusable for one hypothesis tester *A* (for instance, a human) may not be so for another hypothesis tester *B* (for instance, a machine).

We, therefore, constrain the focus on *statistical methods* to address questions such as the above *empirically* for given data. In particular, we are interested in the relation of corpus size (number of speakers) and the probability of a false alarm (P_{FA}) for a given ASV system, under a specific model detailed in Section 2. The input data to our proposed model consists of detection scores (log-likelihood ratios or uncalibrated raw scores) of *any* ASV system on a specific corpus. This makes the method widely applicable for the analysis of any ASV system, treated as a black-box.

The reader familiar with performance assessment of ASV systems may wonder if there is anything new to say about detection scores of a given system on a given corpus. Indeed, measuring detection errors (including P_{FA}) and calibrating speaker recognition systems is a fairly standardized activity (Doddington et al., 2000; Brümmer and du Preez, 2006). So, what is new here? The answer, in brief, is that in the NIST-style ASV evaluations, the *non-target* speaker trials (pairwise comparisons of test utterances against a hypothesized speaker model with disjoint speaker identities) are, essentially, random pairs of speakers. We use more effort to model situation of more confusable (closest) pairs of speakers; one could argue a recognizer that handles the 'worst cases' (closest competing) speakers well may exhibit improved generalization.

In our model, 'closest' speakers are in fact *none* of the non-target speakers in the training set, but *virtual speakers* sampled from the distribution that models random sampling of speakers. Specifically, speakers are represented implicitly by distributions of scores corresponding to pairs of speakers. This allows us to *extrapolate* beyond the given evaluation corpus to arbitrarily large virtual speaker populations. Assuming that the observed speaker pairs are sampled from a same underlying generative process, we can get an idea of how the ASV system scales up with corpus size, *without collecting new speech data*.

While the technical voice conversion spoofing attacks have received a lot of attention in the recent years, it might be appropriate time to re-address worst-case impostors in the context of regular ASV as well. The initial spark for this work stems from our recent work (Vestman et al., 2020) (inspired by Lau et al., 2004) where we addressed a specific research hypothesis relating to potentially emerging, yet cursorily addressed vulnerability of ASV technology *against itself*. The idea was that an attacker could use (public-domain) ASV system as a voice search engine to identify suitable target speaker (specifically, the closest one), such as a celebrity or any person who uploads a lot of his/her voice or video samples to the Internet. After identifying a suitable target, the attacker would attempt to attack another ASV system (e.g., at bank) using natural (possibly mimicked[2]) voice. Despite the relatively large VoxCeleb corpus with more than 7000 target speakers, none of our attackers were successful in getting falsely accepted.[3] While good news concerning security of ASV, the finding was on specific ASV systems, attackers and target corpus. One reason why the finding in Vestman et al. (2020) might have been negative is that the attacker's ASV (designed to be purposefully different from the attacked one) was not powerful enough. Nonetheless, we saw transferability across our two ASV systems in terms of relative target speaker rankings, suggesting that the attacks might be successful with a scaled-up database. We argue that there *must be* a speaker database size (possibly very large) where one is likely to locate closely-matched non-target voices — effect which we were unable to *observe* under the specific experimental conditions. For these reasons, we wanted to re-address the problem by using a more principled and re-usable setup that requires neither two ASV systems (attacker's ASV and targeted ASV) nor fresh recordings. To be precise, the framework proposed in this study addresses a *worst-case* attack scenario with the following two assumptions:

1. *Assumption 1: known ASV system.* The adversary's ASV system (used for identifying closest targets to attack) is the same as the attacked ASV system.
2. *Assumption 2: access to target's enrollment data.* The adversary has access to the target speaker's enrollment data (alternatively, no domain mismatch exist between target's public-domain and enrollment recordings).

The generative model presented in this work enables us to increase the corpus size indefinitely to establish empirical performance bounds on the false alarm rate, under these two assumptions. As search queries to the attacked system can be limited and the enrollment utterances can be protected by template protection techniques, neither assumption is necessarily realistic from the perspective of the adversary. An evaluation corpus designer, technology vendor, or a bank, however, may still want to assess worst-case performance. Importantly, the above assumptions greatly simplify the set-up over the scenarios addressed in Vestman et al. (2020). The methods developed in this study can be seen as an extension of the arsenal of statistical performance evaluation tools. We address each of the two assumptions in the empirical part.

We summarize our two main contributions as follows. First, we propose a general-purpose performance metric, *worst-case false alarm rate with N impostors* (P_{FA}^{N}). It is the probability of accepting the closest impostor among N available candidate impostors selected randomly for each enrolled speaker. As will be discussed below, the proposed metric reduces to the 'conventional' probability of a false alarm (P_{FA}) if $N = 1$. Second, we devise a hierarchical Bayesian generative model of non-target score distribution to enable prediction of P_{FA}^{N} for arbitrarily large values of N that can exceed the number of non-target speakers in a given corpus. The proposed model allows one to make a prediction of the safety of given ASV technology, in its current state, for arbitrarily large speaker database size consisting of virtual (sampled) speakers. Importantly, as the training data consists of detection scores only, the framework is widely applicable for the analysis of arbitrary ASV system (or even other biometric systems). Further, all

[2] Mimicry is a special skill, based on the idea of a listener trying to match his or her acoustic profile with that of another person. As the acoustic correlates of speaker identity, as learned by current ASV systems, remain largely unknown, human mimicry is generally an inconsistent strategy to spoof ASV systems. This is why Vestman et al. (2020) included ASV system to first identify targets that are similar to attacker's voice.

[3] To be more precise, in Vestman et al. (2020), we did not consider hard binary decisions but analyzed changes in the log-likelihood ratio (LLR) scores of the ASV systems. The nontarget LLRs, whether or not originating from zero-effort or mimicry trials were far below the range of target LLRs.

the model parameters are automatically inferred from data, leaving no manually-tunable control parameters to be set. As a representative snapshot of the current ASV technology and evaluation databases, our proof-of-concept experiments include two widely-used ASV methods based on *i-vector* (Dehak et al., 2011) and *x-vector* (Snyder et al., 2018) embeddings, evaluated on the combined VoxCeleb1 (Nagrani et al., 2017) and VoxCeleb2 (Chung et al., 2018) corpora.

2. Measuring and extrapolating false alarm rates

An automatic speaker verification (ASV) system is a hypothesis testing machine that takes a pair of speech utterances $\mathcal{X}=(\mathcal{X}_\mathrm{e},\mathcal{X}_\mathrm{t})$ — one for enrollment, one for test — and produces a numerical detection score $s\in\mathbb{R}$, with the convention that higher values (in relative terms) indicate stronger support for the *same speaker* (null) hypothesis and low scores for the *different speaker* (alternative) hypothesis. Speech utterances are typically represented as fixed-sized *speaker embeddings* such as *i-vectors* (Dehak et al., 2011) or *x-vectors* (Snyder et al., 2018) and the detection score is a *logarithmic likelihood ratio* (LLR) produced by a statistical back-end model, such as the *probabilistic linear discriminant analysis* (PLDA) (Prince et al., 2007; Kenny, 2010).

2.1. False alarm rate

The detection score s can be interpreted as a realization of a continuous random variable that admits an underlying probability density $p(s)$, with $p(s)\geq 0$ and $\int_{s=-\infty}^{\infty} p(s)\mathrm{d}s=1$. In the conventional ASV set-up (as in NIST SREs Doddington et al., 2000; Sadjadi et al., 2017), the performance of an ASV system is assessed using two types of users, *targets* and *nontargets*. The former means that speaker identities of $\mathcal{X}_\mathrm{e}$ and $\mathcal{X}_\mathrm{t}$ match, while the latter means that they differ. We denote the class-conditional score densities of targets and nontargets by $p(s|\mathrm{tar})$ and $p(s|\mathrm{non})$, respectively.

Our focus is on ASV security against impostors, characterized by the nontarget score distribution. In specific, an ASV system is characterized by the probability of accepting a random impostor (sometimes known as *zero-effort* impostor), known as *false alarm rate* (or *false acceptance rate*). It is defined as the following non-increasing function of detection threshold $\tau\in\mathbb{R}$:

$$\mathrm{P}_\mathrm{FA}(\tau)=\int_{\tau}^{\infty} p(s|\mathrm{non})\,\mathrm{d}s, \tag{1}$$

where τ is fixed in advance to set $P_\mathrm{FA}(\tau)$ to a desirable level (increasing τ reduces false alarm rate but increases target rejection rate, also known as *miss rate*).

As we do not have access to $p(s|\mathrm{non})$, in practice $\mathrm{P}_\mathrm{FA}(\tau)$ is usually approximated using *Monte-Carlo* (MC) methods (Robert and Casella, 2005). Monte-Carlo integration is a class of numerical methods that can be used to evaluate expected values of complicated functions. It replaces integrals in expectations by finite sums with the help of independent samples drawn from the underlying probability distribution. By using $\mathbb{I}\{\cdot\}$ to denote an indicator function that equals 1 for a true proposition and 0 otherwise, we write the MC-approximated false alarm rate as,

$$\begin{aligned}\mathrm{P}_\mathrm{FA}(\tau) &= \textstyle\int_{-\infty}^{\infty} p(s|\mathrm{non})\mathbb{I}\{s>\tau\}\,\mathrm{d}s\\ &=\mathbb{E}_{s\sim p(s|\mathrm{non})}[\mathbb{I}\{s>\tau\}]\approx\frac{1}{R}\sum_{r=1}^{R}\mathbb{I}\{s_r>\tau\},\quad s_r\sim p(s|\mathrm{non}),\end{aligned} \tag{2}$$

by assuming one is able to obtain R independent samples s_r from the non-target score distribution. Here, $\mathbb{E}_{s\sim p(s)}[g(s)]$ denotes expected (average) value of function $g(s)$ w.r.t. the distribution $p(s)$. Usually we have just a finite collection of detection scores $\{s_r\}_{r=1}^{R}$ with no further knowledge of $p(s|\mathrm{non})$.

2.2. Reinterpreting false alarm rate as averaged speaker-pair conditioned false alarm rate

In the following, we provide an alternative view of the false alarm rate as an average of speaker-pair specific false alarm rates, useful in paving way towards a new performance metric and a generative model designed to extrapolate its values beyond available datasets. To that end, note first that the detection scores $\{s_r\}$ are obtained through an ASV system that processes some pre-defined *trial list* formed from a finite set of pairwise speaker comparisons. Thus, the terms in (2) can be divided into groups corresponding to unique pairs of speakers. In the special case when these groups are of equal size, we can rewrite the sum in (2) as

$$\frac{1}{R}\sum_{r=1}^{R}\mathbb{I}\{s_r>\tau\}=\frac{1}{T}\sum_{i=1}^{T}\underbrace{\frac{1}{L_i}\sum_{l=1}^{L_i}\mathbb{I}\{s_{i,l}>\tau\}}_{\substack{\text{speaker-pair specific}\\ \text{probability of a false alarm}}}, \tag{3}$$

where $s_{i,l}$ denotes the lth trial score from speaker pair i, L_i is the total number of scores for the ith speaker pair, and T is the total number of speaker pairs, such that $L_1=L_2=\ldots=L_T=L$ and $R=T\cdot L$. Here, the inner sum can be interpreted as the probability of a trial from a given pair of speakers being incorrectly accepted, with the outer sum forming average of the speaker-pair specific false alarm probabilities.

The above simple reformulation provides a bridge towards our proposed framework detailed below. As our approach enables extrapolation of $P_\mathrm{FA}(\tau)$ estimates beyond a given speech corpus, it is necessary to proceed from the empirical averaged false alarm rate (3) towards a continuous-space formulation. In specific, as illustrated in Fig. 1, we require a model that enables sampling both speakers and speaker-pair specific scores from continuous distributions. Note, first, that the distribution of non-target scores $p(s|\text{non})$ can be seen as a continuous mixture of score distributions between all possible pairs of speakers,

$$p(s|\text{non}) = \iint p(s|\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t})p(\boldsymbol{y}_\text{e})p(\boldsymbol{y}_\text{t})\,\mathrm{d}\boldsymbol{y}_\text{e}\mathrm{d}\boldsymbol{y}_\text{t}, \tag{4}$$

where we have introduced two new vector-valued variables $\boldsymbol{y}_\text{e}$ and $\boldsymbol{y}_\text{t}$, viewed as so-called *latent identity variables* (Prince et al., 2007; Kenny, 2010). Let $\boldsymbol{y} \in \mathcal{Y}$ be an element of some space $\mathcal{Y}$. The latent identity variable framework (Prince et al., 2007) assumes that $\boldsymbol{y}$ is a pure representation of a person's identity and that there is a distribution on $\mathcal{Y}$ with known probability density function $p(\boldsymbol{y})$. Given a likelihood function for the latent identity variable (e.g., *meta-embedding* Brümmer et al., 2018), one can make inferences about speaker identities within a set of speech utterances. Examples of such tasks include speaker verification, identification and clustering (Brümmer and de Villiers, 2010). For instance, speaker verification involves testing whether two sets of utterances belong to the same or to different speakers. In this setup the unit of observations, a speech utterance, corresponds to a single speaker identity.

The same framework can also be used in the score domain where observations correspond to pairs of identities. Given a pair of (unknown) identity variables $\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t} \in \mathcal{Y}$, one can describe the distribution of similarity scores between the corresponding speakers by the density function $p(s|\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t})$. This allows to conduct a test of alternative hypotheses such as: (i) two sets of scores belong to different pairs of speakers, (ii) two sets of scores share one common speaker, (iii) two sets of scores belong to the same pair of speakers.

The representation (4) allows us to rewrite the false alarm probability $\text{P}_\text{FA}(\tau)$ akin to (3), namely,

$$\begin{aligned}\text{P}_\text{FA}(\tau) &= \int_\tau^\infty p(s|\text{non})\,\mathrm{d}s = \int_\tau^\infty \left(\iint p(s|\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t})p(\boldsymbol{y}_\text{e})p(\boldsymbol{y}_\text{t})\mathrm{d}\boldsymbol{y}_\text{e}\mathrm{d}\boldsymbol{y}_\text{t}\right)\mathrm{d}s \\ &= \iint \underbrace{\left(\int_\tau^\infty p(s|\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t})\,\mathrm{d}s\right)}_{\substack{\text{speaker}-\text{pair specific} \\ \text{probability of a false alarm}}} p(\boldsymbol{y}_\text{e})p(\boldsymbol{y}_\text{t})\,\mathrm{d}\boldsymbol{y}_\text{e}\mathrm{d}\boldsymbol{y}_\text{t},\end{aligned} \tag{5}$$

where the inner integral is the speaker-pair specific probability of a false alarm and the outer two integrals correspond to summing over all possible speaker pairs.

Given a trial list with speaker IDs, one can obtain the estimate of $\text{P}_\text{FA}(\tau)$ using so-called *nested* Monte-Carlo (Rainforth et al., 2018). It uses MC estimate of the inner integral in (5) to compute MC estimate of the outer integral. The corresponding nested sampling scheme consists of sampling a pair of speakers, followed by sampling a set of scores from the speaker-pair specific score distribution. In practice, any trial list consisting of T unique speaker pairs and the corresponding scores can be thought as being generated according to this scheme. For instance (see Fig. 1), the following generative process produces the scores suitable for computing the nested MC estimate of $\text{P}_\text{FA}(\tau)$:

1. sample an enrolled speaker $\boldsymbol{y}_\text{e}^{(i)} \sim p(\boldsymbol{y})$
2. sample a test speaker $\boldsymbol{y}_\text{t}^{(i)} \sim p(\boldsymbol{y})$
3. sample n_e utterances of the enrolled speaker $\boldsymbol{x}_{e,j} \sim p(\boldsymbol{x}|\boldsymbol{y}_\text{e}^{(i)}),\ j = 1,2,\ldots,n_\text{e}$
4. sample n_t utterances of the test speaker $\boldsymbol{x}_{\text{t},k} \sim p(\boldsymbol{x}|\boldsymbol{y}_\text{t}^{(i)}),\ k = 1,2,\ldots,n_\text{t}$
5. compute $L_i = n_\text{e} \cdot n_\text{t}$ pairwise scores $s_{j,k} = \text{score}(\boldsymbol{x}_{e,j},\boldsymbol{x}_{\text{t},k})$ using an ASV system.

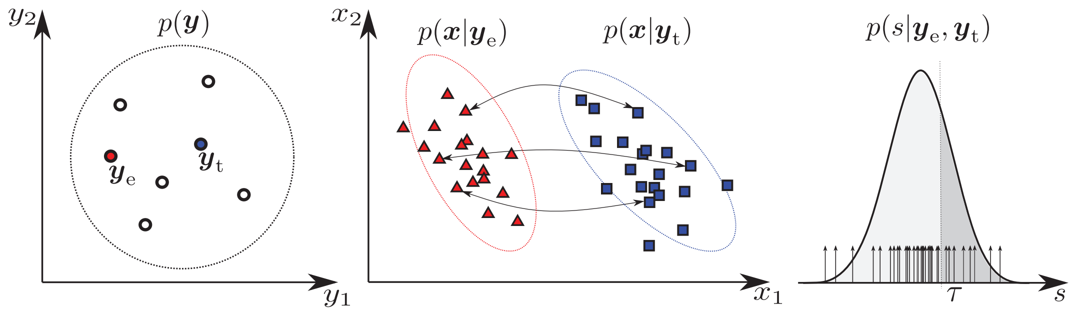

y_2
$p(\boldsymbol{y})$
$\boldsymbol{y}_\text{e}$
$\boldsymbol{y}_\text{t}$
y_1
x_2
$p(\boldsymbol{x}|\boldsymbol{y}_\text{e})$
$p(\boldsymbol{x}|\boldsymbol{y}_\text{t})$
x_1
$p(s|\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t})$
τ
s

Fig. 1. Illustration of the steps to obtain the speaker-pair conditioned score distribution. (Left) Latent speaker identity space. Each element of this space corresponds to unique identity. Small circles represent a dataset consisting of 7 speakers. (Middle) Observation space. Here, $p(\boldsymbol{x}|\boldsymbol{y})$ is the distribution of utterances $\boldsymbol{x}$ of the speaker $\boldsymbol{y}$. (Right) Score space. Here, $p(s|\boldsymbol{y}_\text{e},\boldsymbol{y}_\text{t})$ is the distribution of similarity scores between utterances of the pair of speakers. Samples from this distribution are shown as vertical arrows. Shaded area corresponds to the speaker-pair conditioned false alarm probability which depends on the decision threshold τ.

Here, the index i runs over all speaker pairs and $p(\boldsymbol{x}|\boldsymbol{y})$ denotes the conditional distribution of speech utterances $\boldsymbol{x}$ belonging to speaker $\boldsymbol{y}$. Here, the last step can be equivalently re-formulated as sampling from the distribution of scores conditioned on a pair of speakers:

1. sample an enrolled speaker $\boldsymbol{y}_{\mathrm{e}}^{(i)} \sim p(\boldsymbol{y})$
2. sample a test speaker $\boldsymbol{y}_{\mathrm{t}}^{(i)} \sim p(\boldsymbol{y})$
3. sample L_i scores $s_l \sim p(s|\boldsymbol{y}_{\mathrm{e}}^{(i)}, \boldsymbol{y}_{\mathrm{t}}^{(i)}),\ l = 1, 2, \ldots, L_i$.

Now, the nested MC estimate of (5) can be found as

$$\mathrm{P}_{\mathrm{FA}}(\tau) \approx \frac{1}{T}\sum_{i=1}^{T} \mathrm{P}_{\mathrm{FA}}^{(i)}(\tau), \tag{6}$$

where

$$\begin{aligned}\mathrm{P}_{\mathrm{FA}}^{(i)}(\tau) &= \int_{\tau}^{\infty} p(s|\boldsymbol{y}_{\mathrm{e}}^{(i)}, \boldsymbol{y}_{\mathrm{t}}^{(i)})\mathrm{d}s \\ &\approx \frac{1}{L_i}\sum_{l=1}^{L_i} \mathbb{I}\{s_l > \tau\}, \quad s_l \sim p(s|\boldsymbol{y}_{\mathrm{e}}^{(i)}, \boldsymbol{y}_{\mathrm{t}}^{(i)}), \quad \boldsymbol{y}_{\mathrm{e}}^{(i)}, \boldsymbol{y}_{\mathrm{t}}^{(i)} \sim p(\boldsymbol{y}).\end{aligned} \tag{7}$$

We refer to $\mathrm{P}_{\mathrm{FA}}^{(i)}(\tau)$ as *speaker-pair conditioned* false alarm rate. It is the fraction of similarity scores between these speakers being above the decision threshold τ.

The $\mathrm{P}_{\mathrm{FA}}(\tau)$ can be estimated based on either a model or available empirical data. In the former case one needs a probabilistic model of between-speaker similarity scores and an algorithm to generate samples from this model. In specific, one must be able to obtain samples from the distribution of speaker identities $p(\boldsymbol{y})$ and from the distribution of similarity scores $p(s|\boldsymbol{y}_{\mathrm{e}}, \boldsymbol{y}_{\mathrm{t}})$ given an arbitrary speaker pair $(\boldsymbol{y}_{\mathrm{e}}, \boldsymbol{y}_{\mathrm{t}})$. An example of such a model will be described in Section 2.4. In the latter case, the distribution $p(\boldsymbol{y})$ is a uniform distribution over speakers' IDs and the observed between-speaker scores can be viewed as being samples drawn from an unknown distribution $p(s|\boldsymbol{y}_{\mathrm{e}}, \boldsymbol{y}_{\mathrm{t}})$. That is, the $\mathrm{P}_{\mathrm{FA}}(\tau)$ can be estimated by repeated selection of random pairs of speakers from a dataset and computing similarity scores between random subsets of their sessions. Algorithm 2.1 summarizes a procedure to estimate the probability of accepting a zero-effort impostor, $\mathrm{P}_{\mathrm{FA}}(\tau)$, given a set of utterances with speaker labels.

One should note that in general case, i.e., when speaker-pair specific subsets have different number of scores, L_i, the estimators defined by (2) and (6) produce different results. The former estimator relies on the unrealistic i.i.d. assumption and does not take into account data dependencies resulting from multiple appearances of the same speaker in a given trial list. In practice, however, limited resources usually do not allow to collect sufficiently many unique pairs of speakers to satisfy this assumption. As a result, the estimate may be biased if some speaker pairs have disproportionately large number of trials compared to the rest. The estimator in (6) compensates this bias by assigning weights to the terms in the sum which are inversely proportional to the number of trials. A more in-depth discussion of data dependence in speaker recognition evaluation can be found in Wu et al. (2017).

Algorithm 2.1

```
Input: Dataset with speaker labels
Result: $\mathrm{P}_{\mathrm{FA}}(\tau)$
for $i = 1 \ldots T$ do
    Select random enrolled (target) speaker, $\boldsymbol{y}_{\mathrm{e}}^{(i)}$
    Select random test speaker, $\boldsymbol{y}_{\mathrm{t}}^{(i)}$
    Compute $L_i$ scores $\{s_l\}$ between $\boldsymbol{y}_{\mathrm{e}}^{(i)}$ and $\boldsymbol{y}_{\mathrm{t}}^{(i)}$
    Compute the MC estimate of the speaker-pair conditioned false alarm probability:
        $\mathrm{P}_{\mathrm{FA}}^{(i)}(\tau) \approx \frac{1}{L_i}\sum_{l=1}^{L_i} \mathbb{I}\{s_l > \tau\}$,
end
Compute the MC estimate of $\mathrm{P}_{\mathrm{FA}}(\tau)$:
    $\mathrm{P}_{\mathrm{FA}}(\tau) \approx \frac{1}{T}\sum_{i=1}^{T} \mathrm{P}_{\mathrm{FA}}^{(i)}(\tau)$
```

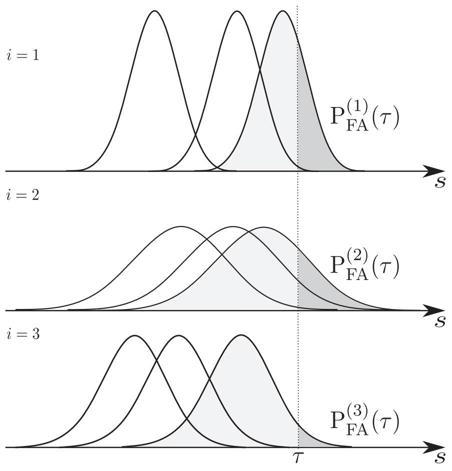


Fig. 2. Illustration of a few iterations of the Algorithm 2.3 in the case of $N = 3$. At each iteration the algorithm samples N non-target speakers. The corresponding speaker-pair conditioned score distributions are depicted as Gaussians. The algorithm selects a distribution with the largest mean value. This distribution is shown as the one with shaded area under the curve. Finally, the algorithm computes the probability of the score being above the decision threshold τ for the selected distribution. This probability, denoted as $\mathrm{P}_{\mathrm{FA}}^{(i)}$, equals the area under the curve to the right of τ. Since the score distributions are not available in practice, one can only compute the empirical estimates of $\mathrm{P}_{\mathrm{FA}}^{(i)}$.

2.3. *Worst-case false alarm rate with* N *impostors*

As (6) suggests, the probability of accepting an impostor speaker can be estimated by averaging the speaker-pair conditioned false alarm probabilities. In particular, Algorithm 2.1 repeats simulation of the zero-effort attack scenario where an impostor speaker is selected at random from the general population.

We propose a new characteristic of ASV systems which generalizes $\mathrm{P}_{\mathrm{FA}}(\tau)$ to attack scenarios where an impostor speaker is selected among N speakers with the intention to fool an ASV system. We call it the *worst-case false alarm rate with N impostors*, denoted by $\mathrm{P}_{\mathrm{FA}}^{N}(\tau)$. Algorithm 2.2 outlines the steps to estimate $\mathrm{P}_{\mathrm{FA}}^{N}(\tau)$. Here, similarity(· , ·) is an arbitrary similarity measure between speakers. The similarity function could be defined, for instance, in a speaker embedding space. In this work, all our models are defined in the score domain. One possible strategy to select the closest speaker is to sample N sets of scores from $p(s|\boldsymbol{y}_{\mathrm{e}}^{(i)}, \boldsymbol{y}_{\mathrm{t},j}^{(i)})$ for $j = 1, \ldots, N$ and select the set with the highest mean value. We adopt this strategy. Fig. 2 illustrates progression of Algorithm 2.2.

Algorithm 2.2 reduces to the zero-effort imposture case if $N = 1$, or if one selects a *random* (among N available) test speaker, rather than the closest one to the enrolled speaker. Fig. 3 demonstrates differences between these cases.

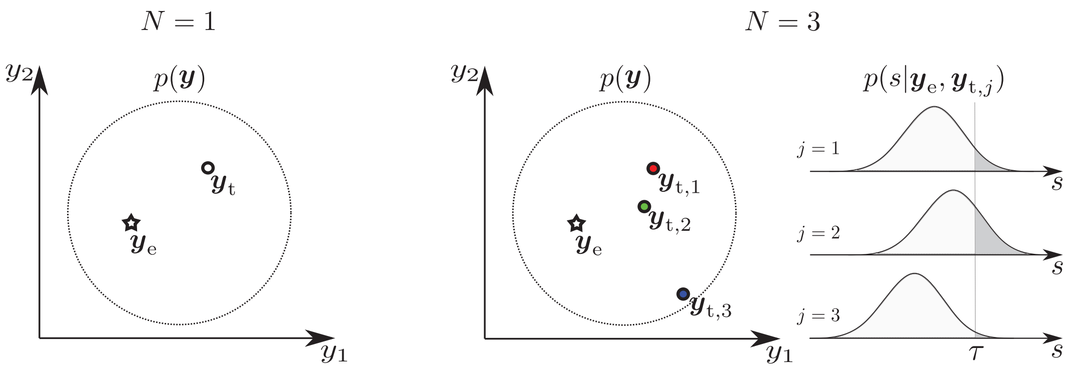

$N = 1$
y_2
$p(\boldsymbol{y})$
$\boldsymbol{y}_{\mathrm{t}}$
$\boldsymbol{y}_{\mathrm{e}}$
y_1
$N = 3$
y_2
$p(\boldsymbol{y})$
$\boldsymbol{y}_{\mathrm{t},1}$
$\boldsymbol{y}_{\mathrm{t},2}$
$\boldsymbol{y}_{\mathrm{e}}$
$\boldsymbol{y}_{\mathrm{t},3}$
y_1
$p(s|\boldsymbol{y}_{\mathrm{e}}, \boldsymbol{y}_{\mathrm{t},j})$
$j = 1$
$j = 2$
$j = 3$
s
τ

Fig. 3. Illustration of two evaluation scenarios where an impostor speaker is selected among $N = 1$ (zero-effort attack) and $N = 3$ impostor speakers. (Left and Middle) Latent speaker identity space. Star represents an enrolled speaker and circles correspond to the impostor speakers. (Right) Distributions of scores between the enrolled speaker $\boldsymbol{y}_e$ and each of the N impostor speakers $\boldsymbol{y}_{\mathrm{t},j}$ for $j = 1, \cdots, \mathrm{N}$.

Algorithm 2.2

```
Input: Dataset with speaker labels
Result: $\mathrm{P}^{N}_{\mathrm{FA}}(\tau)$
for $i = 1...T$ do
    Select random enrolled (target) speaker, $\boldsymbol{y}^{(i)}_{\mathrm{e}}$
    Select $N$ random test speakers, $\boldsymbol{y}^{(i)}_{\mathrm{t},1}, \boldsymbol{y}^{(i)}_{\mathrm{t},2}, ..., \boldsymbol{y}^{(i)}_{\mathrm{t},N}$
    Find the closest speaker $\boldsymbol{y}^{(i)}_{\mathrm{t},k}$, where
        $k = \arg\max_{j} \mathrm{similarity}(\boldsymbol{y}^{(i)}_{e}, \boldsymbol{y}^{(i)}_{\mathrm{t},j})$
    Compute $L_i$ scores $\{s_l\}$ between $\boldsymbol{y}^{(i)}_{\mathrm{e}}$ and $\boldsymbol{y}^{(i)}_{\mathrm{t},k}$
    Compute the MC estimate of the speaker-pair conditioned false alarm probability:
        $\mathrm{P}^{(i)}_{\mathrm{FA}}(\tau) \approx \frac{1}{L_i} \sum_{l=1}^{L_i} \mathbb{I}\{s_l > \tau\}$,
end
Compute the MC estimate of $\mathrm{P}^{N}_{\mathrm{FA}}(\tau)$:
    $\mathrm{P}^{N}_{\mathrm{FA}}(\tau) \approx \frac{1}{T} \sum_{i=1}^{T} \mathrm{P}^{(i)}_{\mathrm{FA}}(\tau)$
```

2.4. Performance extrapolation through generative model of scores

Note that in the above strategy, the value of N is limited by the number of speakers in the dataset. Here, we describe an approach to extrapolate $\mathrm{P}^{N}_{\mathrm{FA}}(\tau)$ for values of N greater than the number of speakers in a dataset. Our main assumption is to approximate the speaker-pair conditioned score distribution $p(s|y_{\mathrm{e}},y_{\mathrm{t}})$ as a (univariate) Gaussian. It should be noted that this assumption, by itself, does not put too many constraints on the shape of the distribution $p(s|\mathrm{non})$ which can be asymmetric and/or heavy-tailed.

In the sequel we describe a probabilistic model of between-speaker scores which follows the generative process in Algorithm 2.2. It will allow to obtain estimates of $\mathrm{P}^{N}_{\mathrm{FA}}(\tau)$ for arbitrary values of N. We introduce two sets of latent variables: $\boldsymbol{\eta}_{\mathrm{e}}$ and $\boldsymbol{\eta}_{\mathrm{t}}$. The variables $\boldsymbol{\eta}_{\mathrm{e}}$ are *shared* among N speaker pairs and represent individual characteristics of the enrolled speaker y_{e}. The variables $\boldsymbol{\eta}_{\mathrm{t},j}$, in turn, are responsible for differences between score distributions within a set of test speakers $y_{\mathrm{t},j}$.

The proposed probabilistic model consists of the distribution of observations $p(s|\boldsymbol{\eta}_{\mathrm{e}}, \boldsymbol{\eta}_{\mathrm{t}})$, which is assumed to be Gaussian, and the prior distribution of latent variables $p(\boldsymbol{\eta}_{\mathrm{e}}, \boldsymbol{\eta}_{\mathrm{t}}) = p(\boldsymbol{\eta}_{\mathrm{t}}|\boldsymbol{\eta}_{\mathrm{e}})p(\boldsymbol{\eta}_{\mathrm{e}})$. Assuming that one can generate random samples of these variables, sampling scores from the model can be done according to the following steps (index i is omitted for clarity):

1. sample $\boldsymbol{\eta}_{\mathrm{e}} \sim p(\boldsymbol{\eta}_{\mathrm{e}})$
2. sample $\boldsymbol{\eta}_{\mathrm{t},j} \sim p(\boldsymbol{\eta}_{\mathrm{t}}|\boldsymbol{\eta}_{\mathrm{e}})$ for $j = 1 \ldots N$
3. sample N sets of scores $\mathcal{S}_j = \{s_{j,l}\}$, where $s_{j,l} \sim p(s|\boldsymbol{\eta}_{\mathrm{e}}, \boldsymbol{\eta}_{\mathrm{t},j})$

We consider a particular instance of such model where $\boldsymbol{\eta}_{\mathrm{t}} = \{\mu\}$, $\boldsymbol{\eta}_{\mathrm{e}} = \{m, \lambda, \sigma^2\}$ and the joint probability density function of the observed score and latent variables is factorized as follows:

$$p(s, \boldsymbol{\eta}_{\mathrm{e}}, \boldsymbol{\eta}_{\mathrm{t}}) = p(s|\mu, \sigma^2)p(\mu|m, \lambda, \sigma^2)p(m)p(\lambda)p(\sigma^2).$$

The individual factors are outlined below:

$$\begin{aligned} &p(s|\mu, \sigma^2) = \mathcal{N}(s|\mu, \sigma^2) \\ &p(\mu|m, \lambda, \sigma^2) = \mathcal{N}(\mu|m, \sigma^2/\lambda) \\ &p(m) = \mathcal{N}(m|\mu_0, \sigma_0^2) \\ &p(\lambda) = \mathrm{Gam}(\lambda|\alpha_\lambda, \beta_\lambda) \\ &p(\sigma^2) = \mathrm{InvGam}(\sigma^2|a_\sigma, b_\sigma). \end{aligned}$$

Here, $\boldsymbol{\theta} = \{\mu_0, \sigma_0^2, a_\sigma, b_\sigma\, \alpha_\lambda, \beta_\lambda\}$ are hyper-parameters which can be estimated on the training set of scores formed according to Algorithm 2.2. Given hyper-parameters, the model can be used to predict $\mathrm{P}^{N}_{\mathrm{FA}}(\tau)$ for arbitrary values of N using Algorithm 2.3. It differs from Algorithm 2.2 in a way that the observed scores are replaced by samples from a generative model meant to approximate the unknown distribution of scores. In the special case of the proposed model the $\mathrm{P}^{N}_{\mathrm{FA}}(\tau)$ can be estimated without explicit sampling of scores. The assumption of the speaker-pair conditioned score distribution being Gaussian allows to compute the

Algorithm 2.3

```
Input: Generative model of scores
Result: $\mathrm{P}^N_{\mathrm{FA}}(\tau)$
for $i = 1...T$ do
    Sample $N$ sets of scores from the model: $\mathcal{S}_j$ for $j = 1...N$
    Find the set with the highest mean score
        $k = \arg\max_j \mathrm{mean}(\mathcal{S}_j)$
    Compute the MC estimate of the speaker-pair conditioned false alarm probability:
        $\mathrm{P}^{(i)}_{\mathrm{FA}}(\tau) \approx \frac{1}{|\mathcal{S}_k|} \sum_{l=1}^{|\mathcal{S}_k|} \mathbb{I}\{s_l > \tau\}, s_l \in \mathcal{S}_k$
end
Compute the MC estimate of $\mathrm{P}^N_{\mathrm{FA}}(\tau)$:
    $\mathrm{P}^N_{\mathrm{FA}}(\tau) \approx \frac{1}{T} \sum_{i=1}^{T} \mathrm{P}^{(i)}_{\mathrm{FA}}(\tau)$
```

estimate as

$$P^N_{\mathrm{FA}}(\tau) \approx \frac{1}{T}\sum_{i=1}^{T} 1 - \Phi\left(\tau|\ \max_{j=1...N}(\{\mu_{i,j}\}), \sigma_i^2\right),$$

where m_i, λ_i, σ_i^2 and $\mu_{i,j}$ are sampled from the corresponding distributions. Here, $\Phi(\cdot)$ denotes cumulative distribution function of the Gaussian distribution.

This model assumes shared variance among score distributions $p(s|\mathbf{y}_\mathrm{e}, \mathbf{y}_{\mathrm{t},j})$ for $j = 1, \cdots, \mathrm{N}$ given a target speaker $\mathbf{y}_\mathrm{e}$. This assumption as well as the choice of specific distributions are primarily motivated by the convenience of computing the posterior distribution of the latent variables. In particular, using *conjugate pairs* of distributions (Gelman et al., 2013) as building blocks in the model allows to devise efficient algorithms to obtain approximate posterior distribution. This leads to closed-form updates in the *expectation-maximization* (EM) algorithm (Dempster et al., 1977) used to estimate the model hyper-parameters, with the details provided in Appendix I. Further insight to the form of the score distributions implied by our model (including its limitations) is provided in Appendix II. In Section 5 we provide discussion of the adequacy of the model assumptions and potential alternatives.

Fig. 4 depicts the Bayesian network of the proposed model. A Bayesian network is a directed graphical model (Bishop, 2006) that represents a set of random variables and their conditional dependencies via a directed acyclic graph. Empty circles denote latent variables, shaded circles denote observed variables and nodes without circles denote deterministic parameters. A group of nodes surrounded by a box, called a plate, labeled with T indicates that the subgraph inside a plate is duplicated T times (Buntine, 1994). The arrows between the nodes point from the parent variables to their children variables and represent the conditional dependencies between these variables.

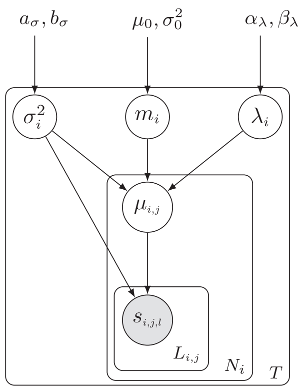


Fig. 4. A graphical representation of the generative model. Here, T is the number of target speakers, N_i is the number of non-target speakers for the ith target speaker, and $L_{i,j}$ is the number of similarity scores between the ith target speaker and the jth non-target speaker.

Table 1
Details of the ASV systems used in this study.

	i-vector system	x-vector system
Acoustic features	24-dimensional MFCCs + delta + double-delta coefficients; energy based speech activity detection	30-dimensional MFCCs; energy based speech activity detection
Background model	Gaussian mixture model of 2048 components with full covariance matrices; trained using the whole training data	–
Embedding extractor	Trained with 100 000 longest utterances in the training set	Trained using the whole training data plus 1000 000 utterances obtained by data augmentation (reverb, noise, babble, music)
Embeddings	400-dimensional i-vectors	512-dimensional x-vectors
LDA and PLDA	Both trained using the whole training data; dimensionality reduction to 200-D with LDA	Both trained using the whole training data; dimensionality reduction to 200-D with LDA

3. Experimental setup

This section describes the ASV systems, protocols, and the dataset we use for the experiments with the proposed worst-case false alarm rate with *N* impostors (P_{FA}^{N}) metric.

3.1. Dataset

A suitable dataset for our experiments has to fulfill two requirements. First, it must have a large number of speakers to not only train well-performing ASV systems, but to have enough speakers in the evaluation side to produce good P_{FA}^{N} estimates from the ASV scores. Second, each speaker in the evaluation side should have enough utterances to produce a sufficiently large number of scores between each pair of speakers, required for reliable P_{FA}^{N} estimation. For these reasons, we chose the VoxCeleb datasets (VoxCeleb1 Nagrani et al., 2017 & VoxCeleb2 Chung et al., 2018). When combined, the datasets contain 7365 speakers and, on average, each speaker has well over 100 utterances, which typically originate from about 20 sessions.

We divided the available speakers into three disjoint sets containing 5345, 40, and 2000 speakers. The first set of 5345 speakers is used to train the ASV systems. The second set of 40 speakers consists of the test speakers in the standard VoxCeleb1 ASV evaluation protocol, which is used for evaluating performance of our ASV systems. The third, gender-balanced set contains 1000 male and 1000 female speakers and is used for the experiments with P_{FA}^{N} estimation. The speakers in this last set were chosen so that each had utterances from at least 18 different sessions; otherwise the split between the first and the last set was random.

3.2. Automatic speaker verification systems

We provide experimental results for two different ASV systems, based on the two most commonly used speaker embeddings, i-vectors (Dehak et al., 2011) and x-vectors (Snyder et al., 2018). We trained both systems using Kaldi (Povey et al., 2011) recipes for VoxCeleb using our custom train-test data division. Both systems use mel-frequency cepstral coefficients (MFCCs) as acoustic features and a combination of *linear discriminant analysis* (LDA) and *probabilistic* LDA (PLDA) in the scoring backend. The fundamental difference between the i-vector and x-vector systems is that the former is based on Gaussian generative model, while the latter is trained discriminatively and utilizes longer time context via time-delay neural network. Another major difference is that the x-vector system is trained with a larger training set leveraging from data augmentation. For further details of the systems, refer to Table 1.

3.3. Evaluation protocols

We used two ASV protocols to serve two different purposes. First, we adopted the standard VoxCeleb1 ASV protocol to assess the performance of our ASV systems. This protocol contains 40 speakers, and $37,720$ evaluation trials with a balanced number of target (same speaker) and non-target (different speaker) trials. The second protocol is used to obtain a large number of non-target scores for a large number of speaker pairs to estimate P_{FA}^{N}. For each of the 2000 speakers in the testing set, we randomly chose 18 utterances so that all the utterances were from different sessions. Then, for each pair of speakers, we obtained $18^2 = 324$ trials by forming all the utterance pairs between the two speakers. In total, we had $1\,999\,000 \times 324 = 647\,676\,000$ trials, where 1 999 000[4] is the total number of unique speaker pairs. The above number includes cross-gender trials. Including only speaker pairs within one gender, we have 161 838 000 trials for both males and females.

4. Results

4.1. Performance of speaker verification systems

Before proceeding to our proposed generative approach, we validate correctness of the ASV implementations through standard performance metrics. To this end, we report *equal error rate* (EER) and *minimum normalized detection cost function* (minDCF)

[4] $2000!/(2!(2000-2)!) = 1\,999\,000$ (number of 2-combinations in a set of 2000 speakers)

Table 2
Parameters of three different detection cost functions (DCF) used in this study. The system thresholds (τ) that mimimize these DCFs are used to estimate false alarm rates in the following experiments.

	P_{target}	C_{miss}	C_{fa}
minDCF$_1$	0.5	10	1
minDCF$_2$	0.5	1	1
minDCF$_3$	0.5	1	10

(Alvin and Martin, 2004). EER is obtained by setting the system threshold τ so that false alarm and miss rates equal each other. The threshold selection for minDCF, in turn, is governed by the parameters P_{target} (prior probability of target speaker), C_{miss} (cost of missing the target speaker), and C_{fa} (cost of falsely accepting a non-target speaker). For this study, we adopt three different sets of parameters (Table 2): the first set has high cost for misses, the second set has equal costs for misses and false alarms, and the last one penalizes false alarms more. From the security perspective, DCF$_3$ is the most relevant, whereas the other two DCFs can be utilized in applications where high security is not required.

Table 3 shows the EERs and minDCFs for i-vector and x-vector systems using VoxCeleb test protocol (category ‘all’). In addition, we split the protocol based on genders and also report a result for the ‘pooled’ category, which does not contain inter-gender trials making the original test protocol more difficult. Our results are in line with the results reported in the original Kaldi recipes. As we used about 2000 speakers less for system training, our EER for x-vector system is about 0.5% (absolute) higher than what is reported in the original recipe.

In addition to the overall performance difference between i-vector and x-vector systems, these systems differ in their ability to recognize speakers from different genders. For the i-vector system, the performance for males is considerably better, whereas for the x-vector system the difference between the genders is smaller.

In Fig. 5, we display score distributions for the VoxCeleb1 test protocol and for our custom protocol containing non-target scores only. The VoxCeleb1 protocol is used to set the system thresholds τ for the P_{FA}^{N} estimation experiments presented in the next section. In these experiments, we use gender-specific thresholds obtained via minimizing DCF separately on male and female trials. Note that for clarity, Fig. 5 does not show gender-specific thresholds, but instead it shows the thresholds for ‘pooled’ category.

4.2. *Estimation of worst-Case false alarm rates*

We estimated worst-case false alarm rates empirically using Algorithm 2.2 by randomly selecting enrolled speaker $T = 1000$ times. Similarly, we use $T = 1000$ in Algorithm 2.3 to obtain model-based estimates. The estimates are shown in Fig. 6 for both ASV systems using three different thresholds obtained using the DCF parameter sets in Table 2. We find that model-based approaches give good estimates when the threshold is low (higher cost for misses). When the threshold is higher than in the minDCF$_1$ case, the model-based estimates can be seen as a conservative upper bounds for the P_{FA}^{N} rates. We also find that the differences between the empirical and the model-based estimates are greater for females than for the males. To obtain further insight, we depict the score distributions of the closest impostors for population size of $N = 1000$ in Fig. 7. The figure indicates that especially for females, the model-based score distributions tend to be too wide and slightly shifted to the right, which causes higher false acceptance rates when a high threshold value is used.

Table 3
Performance of i-vector and x-vector systems on VoxCeleb1 test protocol. The original protocol (‘all’) contains both intra- and inter-gender trials. The numbers under the category ‘pooled’ are computed using only intra-gender trials from both genders.

	minDCF$_1$	minDCF$_2$	minDCF$_3$	EER (%)
i-vector				
male	0.43	0.14	0.31	6.97
female	0.53	0.17	0.37	8.80
pooled	0.44	0.14	0.34	7.18
all	0.30	0.11	0.27	5.62
x-vector				
male	0.27	0.09	0.24	4.71
female	0.30	0.10	0.24	5.19
pooled	0.28	0.09	0.23	4.75
all	0.21	0.07	0.19	3.61

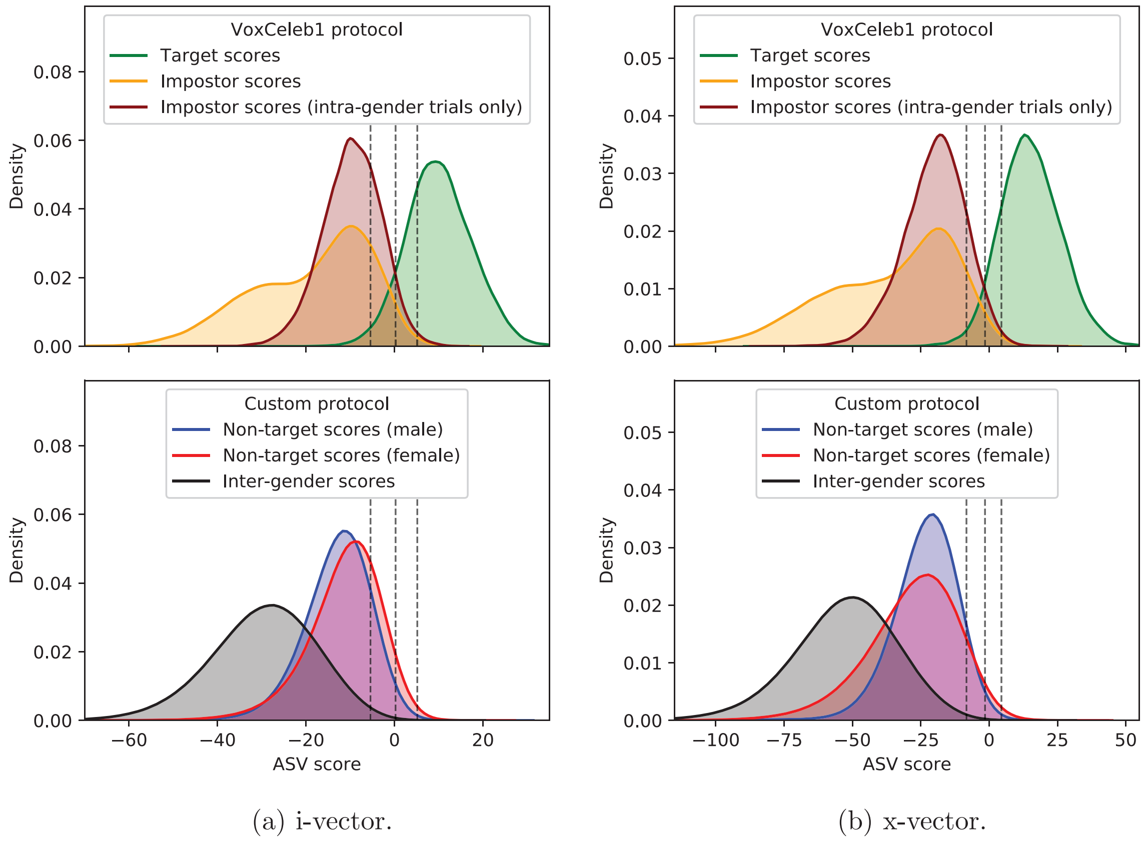


Fig. 5. Score distributions for the standard VoxCeleb protocol and the custom protocol obtained using i-vector and x-vector systems. Dashed lines represent minDCF thresholds for 'pooled' scores (see Table 3) using different sets of cost parameters presented in Table 3.

Using the estimates, we can predict that for the minDCF$_1$ threshold, P_{FA}^{N} is 95–98% for an impostor population of size 100 000. For minDCF$_3$ threshold, we can rely only on the empirical estimates, which tell us that, depending on the system and gender, P_{FA}^{N} rate of 12–28% is obtained for a population of size 1000. Note that our populations contain only speakers from one gender. If the population would contain speakers from both genders, the false alarm rates would be lower.

5. Discussion

Before concluding, the authors would like to address two relevant concerns, the over-estimated false alarm rates at high threshold, and the worst-case attack assumption.

5.1. Analysis of the model-based worst-case false alarm estimation

From Fig. 7, we observe two apparent problems in the score distributions given by the model for the closest impostors: (a) they are shifted to the right and (b) they have too large variances. As a result, some of the generated scores of the closest impostors are too high, which results in over-estimated false alarm rates given by the model. We have identified three causes for the problems.

First, we found that the empirical distribution of μ, which is the distribution of score means of speaker pairs, is skewed to the left (negative skewness, see Table 4). Consequently, the fitted normal distribution (assumed in the model) has longer tail on the right than what the original score data had. The right tail of the distribution of μ is where we will find the closest impostors in Algorithm 2.3. As a result, the scores of the closest impostors are shifted to the right.

Additionally, we observed that the variation in target-vs-impostor scores is smaller for speaker pairs with the closest impostors than for random impostors. In other words, the closer the impostor's voice is to the target speaker's voice, the smaller is the variance in scores between the two speakers. As our model does not take this into account, the speaker pairs with closest impostors tend to have too large score variances.

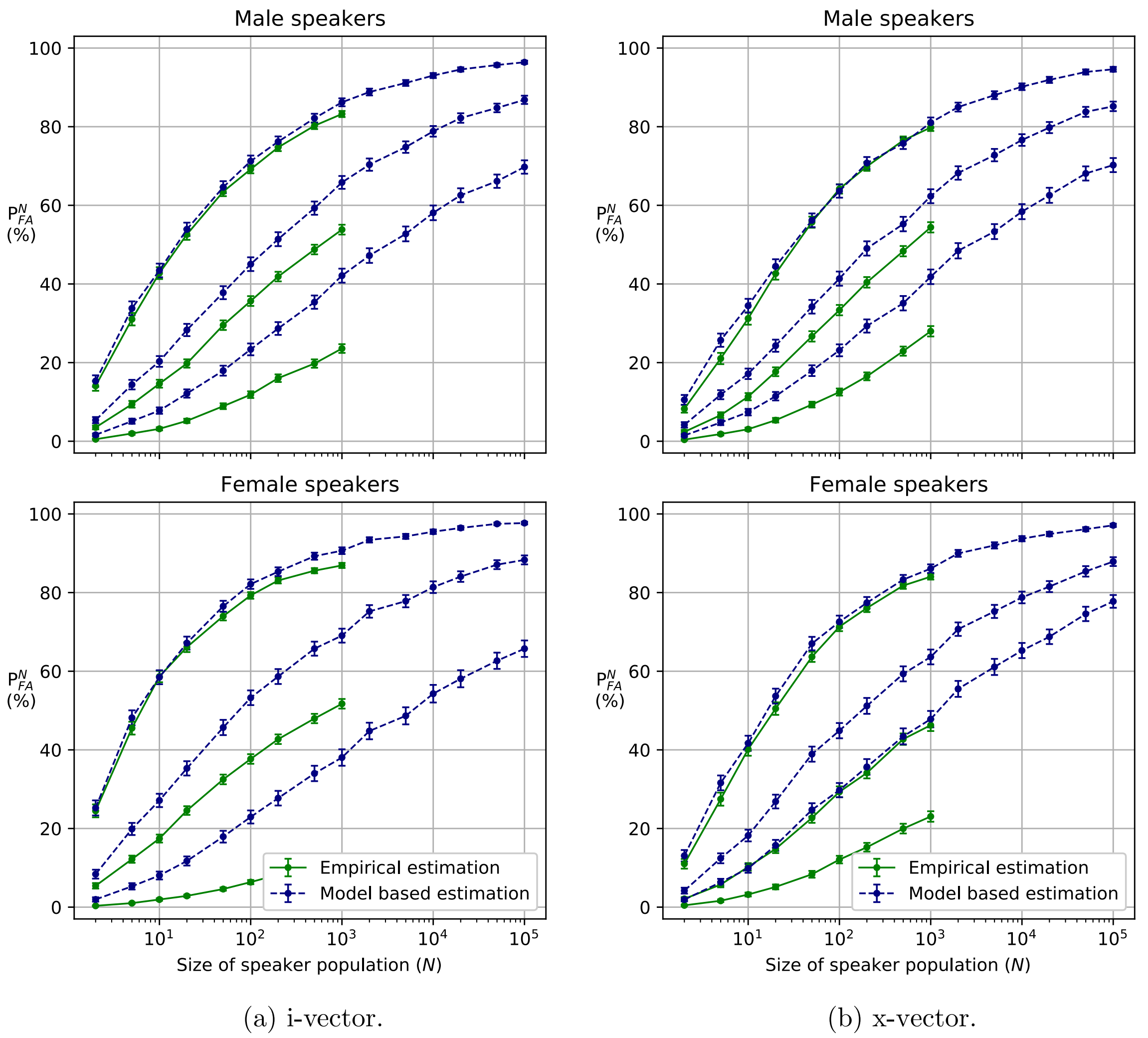


Fig. 6. Empirical and model-based estimates of worst-case false alarm rates with N impostors for various sizes of speaker populations. In each plot, three empirical and model-based estimates are shown for three different thresholds. These thresholds are obtained separately for each plot using cost parameters defined in Table 2. The curves from top to bottom correspond to the thresholds of minDCF$_1$, minDCF$_2$, and minDCF$_3$, respectively. The model-based estimates follow closely emprical estimates for low threshold values, but as the threshold gets stricter, the difference between the empirical and model-based estimates grows. The curves are obtained using $T = 1000$ in Algorithms 2.3 and 2.4. The mean values obtained using these algorithms are shown together with their 99% confidence intervals.

Finally, the scores between the speaker pairs are also skewed to the left, which is another source of mismatch between empirical scores and scores generated by the model.

These observations open two potential directions towards increasing the prediction accuracy. The first direction is to revise the proposed generative model to take into account the skew of score distributions, as well as by relaxing the assumption of a shared variance. This can be done at the cost of losing conjugacy between distributions in the model, leading to increased computational complexity of hyper-parameter estimation. An alternative, second direction would be supervised fine-tuning to optimize some loss function between the empirical and the model-based estimates. As our model is parameterized only by six numbers, we believe that a good hyper-parameter configuration can be found in reasonable time using one of the *derivative-free* optimization methods (Larson et al., 2019). To give some empirical evidence for this claim, Fig. 8 displays an example where we tuned our model parameters *manually*. As seen, the model itself is actually flexible enough to fit the empirical false alarm rates accurately — but the purely generative training criterion does not find the parameter values that achieve this. With the manually corrected model, we obtained 54% worst-case false acceptance rate estimate for population size of 100, 000 for the strictest minDCF$_3$ threshold, whereas the original model clearly over-estimated this by giving FA rate of 70% as shown in the top-right panel of Fig. 6.

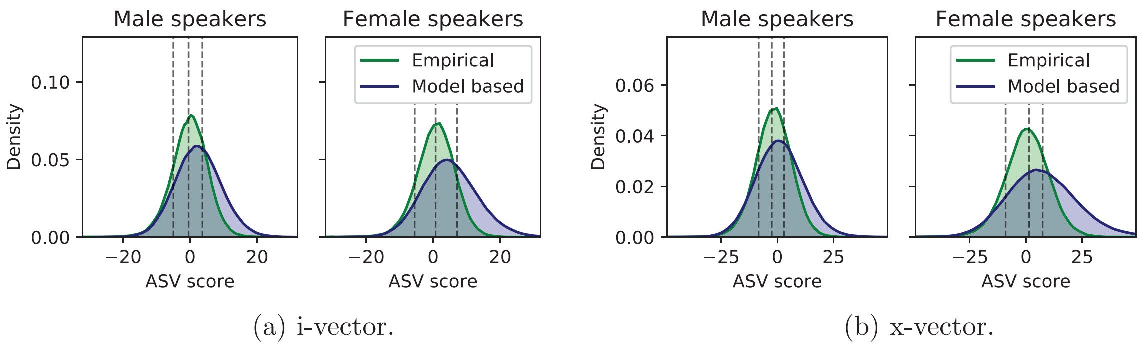


Fig. 7. Score distributions of the closest impostors ($N=1000$) pooled together from $T=1000$ samplings/simulations for the empirical and model-based approaches. Dashed lines represent minDCF thresholds obtained using different sets of cost parameters, which are presented in Table 3. For the strictest threshold, the area under the density curves on the right side of the threshold is larger for the model-based estimation, which explains why the model-based estimation lead to higher false alarm rates as shown in Fig. 6.

Table 4
Sources of mismatch between the observed and generated scores. The generative score model assumes that pairwise non-target scores and means of pair-wise scores (μ) are normally distributed, while the analysis shows that they are skewed to the left. Additionally, scores with the closest impostors tend to have smaller variances than scores with random impostors, which is not factored into the model.

	i-vector system		x-vector system	
	males	females	males	females
Avg. skewness of pairwise scores	−0.20	−0.29	−0.20	−0.27
Skewness of μ	−0.86	−0.99	−0.99	−0.62
Avg. STDEV* of scores with the closest impostors	5.1	5.5	7.5	9.4
Avg. STDEV* of scores with random impostors	6.2	7.1	9.6	13.7

* Computed as a square root of an average of variances.

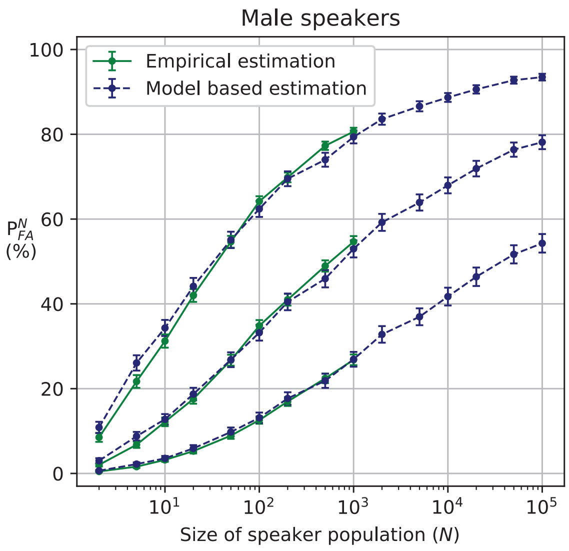
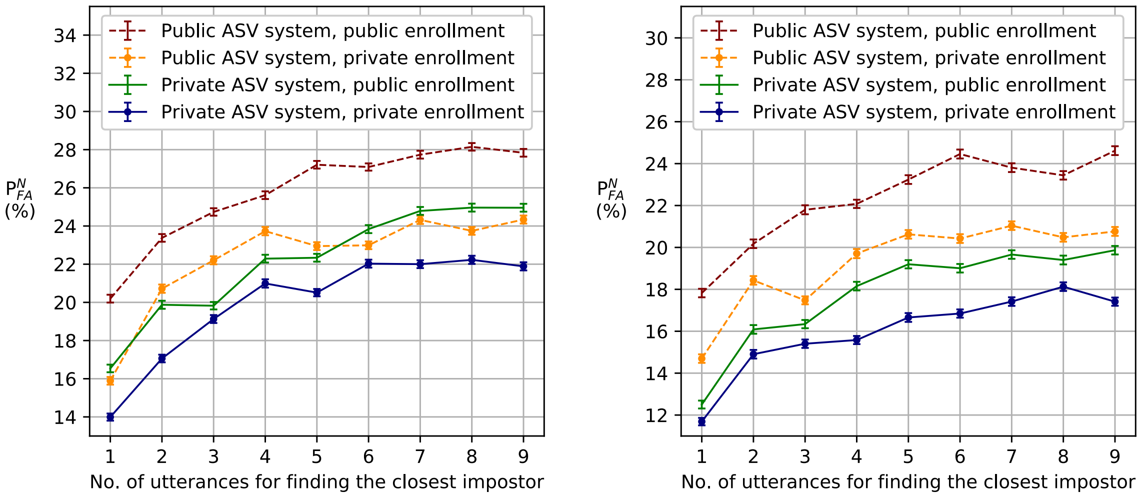


Fig. 8. Worst-case false alarm estimates for male scores given by x-vector system after tweaking the model hyper-parameters manually. First, the parameter α_λ was adjusted until the variances of distributions in Fig. 7 matched and then μ_0 was adjusted to fix the shifting misalignment. As a result, the model-based estimates follow closely the empirical estimates unlike in Fig. 6.



(a) Male speakers. (b) Female speakers.

Fig. 9. Empirical estimation of P_{FA}^{N} ($N = 1000$) in various scenarios for x-vector system with minDCF$_3$ threshold (see Table 2). 'Public ASV system' refers to a case where the closest impostors in Algorithm 2.3 are selected using the same x-vector system. To simulate a scenario, where attacker uses another ASV system for impostor selection due to not knowing the details of the deployed system ('private ASV system'), an i-vector system is used to select the closest impostors. Further, 'public enrollment' and 'private enrollment' refer to such cases, where the attacker has access (public) or does not have access (private) to the enrolled target speaker's enrollment data. If the enrollment data and the ASV system are public, the selection of the closest impostor is easier, which results in higher false acceptance rates.

5.2. *False alarm estimation in simulated attack scenarios*

So far we have considered worst-case false alarm estimation from the system deployer's perspective. From the presented results, we can gain understanding on how many enrolled speakers systems with specific thresholds can handle without starting to confuse speakers to each other too much.

Next, let us consider a scenario, in which a malicious attacker is utilizing ASV technology to find similar sounding speakers to the enrolled target speaker's voice to break the ASV system. As discussed in Section 1, the previously presented results can be considered as the worst-case situation, where the attacker has access to both the deployed ASV system as well as to the target speaker's enrollment data. In reality, the attacker would be unlikely to have access to either of them. Instead, the attacker would first have to set up another ASV system and then collect some speech data from the target speaker to perform the speaker search. These steps will make the attack more difficult as the closest impostor obtained using attacker's system and data might not be the same as what would be the closest impostor when using the attacked system and the real enrollment data.

To study the effect of system/data mismatch in the impostor selection, we set up a following experiment. First, we divided the available 18 utterances for each speaker into two disjoint sets of nine utterances. The first one was used for impostor selection and the second for speaker enrollment. We compared this setup to a case, where the same set of nine utterances was used both for impostor selection and enrollment to address the effect of data mismatch. Further, we also varied the number of utterances used for impostor selection from one to nine to see the effect of the amount of data used for impostor search. We simulated the ASV system mismatch by using i-vector system to select closest impostors, while the x-vector system was considered to be the attacked system. This was compared to the case where the impostor search was done using the same attacked x-vector system.

The results are shown in Fig. 9, which reveals the expected patterns: when there is no data mismatch or ASV system mismatch, false acceptance rates are highest, which means that the attacks are most successful. If there is either data mismatch or system mismatch, the false acceptance rates drop. The lowest false acceptance rates are obtained, when both types of mismatches are present and when the number of utterances available for impostor search is low.

As another future direction, we consider designing a model for joint modeling of scores from two different ASV systems suitable for more realistic scenario where the attacker does *not* have access to the target speaker's enrollment data and the deployed ASV system.

6. Conclusions

Seamless integration of artificial intelligence to our daily lives, including speech technology products, raises growing concern of their trustworthiness and safety. Our study resides in the landscape of automatic speaker verification (ASV), or voice biometrics security. One unique feature of voice (and face) biometrics is that, unlike traditional physical biometrics — fingerprints, iris, retina, DNA to name a few — is that much of the biometric data is *publicly available in the Internet* through social media, news, interviews, lectures, and workplace websites to name a few. An important concern is the relation of false alarm (false acceptance) and database size: regardless of the selected ASV technology, given a large enough database, one will eventually have speaker collisions. A technology-aware attacker may increase the likelihood of such collisions through the use of public-domain ASV system to identify target speakers from a public database (Vestman et al., 2020). Even without dedicated attacks, however, the number of different voices (subject to extrinsic and intrinsic speech variations) is not infinite. Therefore, eventually, ASV performance will be capped at some database size.

The methodology concept put forward in this study gives us novel tool to address the dependency of false alarm rate and database size beyond the size of a given evaluation corpus. The proposed model produced reasonable match with empirical scores and displayed the expected trend of increasing false alarm rate as a function of database size. Our model is general and can be applied to analyze (and optimize) any black-box ASV system to produce graphs similar to those in Fig. 6, based on detection scores only. As such, these graphs are *predictions* by the model — how the ASV system will behave if one were able to collect more speakers assuming the speaker sampling process remains the same. Even if it is not easy to experimentally validate the model beyond a given training corpus size, the general trends what we saw in our pilot experiments with VoxCeleb corpus are deemed as expected: false alarm rates increase as a function of database size and will eventually saturate.

Our work has a number of limitations as well. First, as the results indicate, the generative model overestimated the false alarm rates, especially for the high-security operating region (high threshold). In real-world deployment of ASV, the detection threshold τ needs to be optimized to achieve a desirable security–convenience trade-off based on some development data and application (DCF setting). If the false alarm rate is over-estimated, the threshold τ would have to be increased (relative to the value it would have been set with precise knowledge of the false alarm rate), leading to decreased user convenience due to increased miss rate. Nonetheless, as Section 5 indicates, our proposed generative model is flexible enough to be adjusted so that the empirical and predicted false alarm rates will match closely. Our model design philosophy has been simplicity: all the parameters are automatically learned from data and we leverage from conjugate families of distributions to enable efficient inference. The suggested future improvements include revising our distributional assumptions, and combining generative modeling with discriminative fine-tuning.

Second, our model assumes a worst-case scenario where the attacker has access to the target speaker's enrollment data, as well as the attacked ASV system. This is no different from standard NIST SRE style evaluations where an evaluator reports standard evaluation metrics (such as EER, minDCF, or P_{FA}) on a given, fixed evaluation corpus with known trial key. In future, we are interested in extending our generative model to model interaction between two different ASV systems and across different data domains.

Furthermore, it would be interesting to compare different ASV systems and database qualities. VoxCeleb data was selected for the experiments primarily due to the large number of speakers and the amount of intra-speaker scores. Nonetheless, being representative of *found Internet data*, VoxCeleb contains many style, channel and environment variations. At least for academic curiosity, it would be interesting to repeat our simulations on more controlled database for reference purposes. Further, it will be interesting to analyze the impacts of i-vector and x-vector dimensionality, dimensionality reduction of these embeddings, and speaker subspace size in PLDA. Finally, it would be interesting to apply the proposed methods to speaker diarization or other use cases within ASV, such as score normalization with increased speaker cohort size.

Acknowledgment

The work has been supported by Academy of Finland (proj. no. 309629 entitled "NOTCH: NOn-cooperaTive speaker CHaracterization") and by the Doctoral Programme in Science, Technology and Computing (SCITECO) of the UEF. The work of the first author was partially financially supported by the Government of the Russian Federation (Grant 08-08). A part of the work of the third author was supported by NEC internship program. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

Appendix A. parameter inference

In the following we describe the algorithm used to estimate hyper-parameters of the proposed generative model. We find the values of hyper-parameters $\boldsymbol{\theta}$ that maximize the likelihood function – the joint probability density of the observed data viewed as a function of $\boldsymbol{\theta} = \{\mu_0, \sigma_0^2, a_\sigma, b_\sigma, \alpha_\lambda, \beta_\lambda\}$:

$$\mathscr{L}(\boldsymbol{\theta}) = \int \prod_{i=1}^{T} \mathcal{N}(m_i|\mu_0, \sigma_0^2)\mathrm{Gam}(\lambda_i|\alpha_\lambda, \beta_\lambda)\mathrm{InvGam}(\sigma_i^2|a_\sigma, b_\sigma)$$
$$\prod_{j=1}^{N_i} \mathcal{N}(\mu_{i,j}|m_j, \lambda_i, \sigma_i^2) \prod_{l=1}^{L_{i,j}} \mathcal{N}(s_{i,j,l}|\mu_{i,j}, \sigma_i^2)\, \mathrm{d}m_i\, \mathrm{d}\lambda_i\, \mathrm{d}\sigma_i^2\, \mathrm{d}\mu_{i,j}$$

For many probabilistic models with latent variables, including the proposed one, this objective function is intractable (cannot be evaluated). A commonly adopted strategy to avoid this obstacle is to use the *expectation-maximization* (EM) algorithm (Dempster et al., 1977), an iterative optimization method to find the local extrema of the likelihood function. The EM algorithm alternates between two steps: *expectation* step (E-step) and *maximization* step (M-step). On the E-step it computes (approximate) posterior distribution of the latent variables and on the M-step it updates all the hyper-parameters of the model.

Inference for latent variable models can be conducted through *variational Bayes* (Bishop, 2006, Chapter 10) or *Monte-Carlo* techniques (Bishop, 2006, Chapter 11). We choose the former approach due to its better scalability in terms of computational costs. The variational Bayesian inference approximates the exact posterior distribution $p(\{m_i\},\{\lambda_i\},\{\sigma_i^2\},\{\mu_{i,j}\}|\{s_{i,j,l}\})$ by a *variational* distribution q from a restricted family of distributions. The variational distribution is found by minimizing the Kullback−Leibler divergence from the posterior distribution.

One of the commonly adopted strategies, known as *black-box variational inference* (BBVI) (Ranganath et al., 2014), is to explicitly define the family of variational distributions and use stochastic optimization (Robbins and Monro, 1951) to minimize the objective. Another strategy to define q is the *mean-field* approximation (Bishop, 2006; Jordan et al., 1999) which assumes the variational distribution to be fully factorized:

$$q(\{m_i\},\{\lambda_i\},\{\sigma_i^2\},\{\mu_{i,j}\})=\prod_{i=1}^{T} q(m_i)q(\lambda_i)q(\sigma_i^2)\prod_{j=1}^{N_i} q(\mu_{i,j}) \tag{8}$$

but with no further assumptions imposed on the functional forms of the factors. For conditionally conjugate models (Wang and Blei, 2013) this approach leads to closed form solutions in a coordinate descent optimization algorithm which iteratively updates the parameters of one factor while holding the others fixed. Since the proposed model is conditionally conjugate we choose to use the mean-field approach due to its lower computational complexity. The inference algorithm performs the following updates performed until convergence.

Expectation step (E-step):

- Updating $q(m_i)$:

$$q(m_i)=\mathcal{N}(m_i|\widehat{m}_i,s_i^2)$$
$$s_i^2=\left(N_i\mathbb{E}[\lambda_i]\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]+\frac{1}{\sigma_0^2}\right)^{-1}$$
$$\widehat{m}_i=\left(N_i\mathbb{E}[\lambda_i]\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]+\frac{1}{\sigma_0^2}\right)^{-1}\left(\mathbb{E}[\lambda_i]\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]\left(\sum_{j=1}^{N_i}\mathbb{E}\left[\mu_{i,j}\right]\right)+\frac{\mu_0}{\sigma_0^2}\right)$$
$$\mathbb{E}[m_i]=\widehat{m}_i,\ \mathbb{E}[m_i^2]=\widehat{m}_i^2+s_i^2$$

- Updating $q(\sigma_i^2)$:

$$q(\sigma_i^2)=\mathrm{InvGam}\left(\sigma_i^2|\widehat{a}_i,\widehat{b}_i\right)$$
$$\widehat{a}_i=a_\sigma+\frac{N_i}{2}+\sum_{j=1}^{N_i}L_{i,j}$$
$$\widehat{b}_i=b_\sigma+\frac{1}{2}\mathbb{E}\left[\sum_{j=1}^{N_i}\sum_{l=1}^{L_{i,j}}(s_{i,j,l}-\mu_{i,j})^2\right]+\frac{1}{2}\mathbb{E}[\lambda_i]\mathbb{E}\left[\sum_{j=1}^{N_i}(\mu_{i,j}-m_i)^2\right]$$
$$\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]=\frac{\widehat{a}_i}{\widehat{b}_i},\ \mathbb{E}[\log\sigma_i^2]=\log\widehat{b}_i-\psi(\widehat{a}_i)$$

- Updating $q(\lambda_i)$:

$$q(\lambda_i)=\mathrm{Gam}\left(\lambda_i|\widehat{\alpha}_i,\widehat{\beta}_i\right)$$
$$\widehat{\alpha}_i=\alpha_\lambda+\frac{N_i}{2}$$
$$\widehat{\beta}_i=\beta_\lambda+\frac{1}{2}\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]\mathbb{E}\left[\sum_{j=1}^{N_i}(\mu_{i,j}-m_i)^2\right]$$
$$\mathbb{E}[\lambda_i]=\frac{\widehat{\alpha}_i}{\widehat{\beta}_i},\ \mathbb{E}[\log\lambda_i]=\psi(\widehat{\alpha}_i)-\log\widehat{\beta}_i.$$

- Updating $q(\mu_{i,j})$:

$$q(\mu_{i,j})=\mathcal{N}\left(\mu_{i,j}|\widehat{\mu}_{i,j},\widehat{s}_{i,j}^2\right)$$
$$\widehat{\mu}_{i,j}=\frac{\sum_{l=1}^{L_{i,j}}+\mathbb{E}[\lambda_i]}{L_{i,j}+\mathbb{E}[\lambda_i]}$$
$$\widehat{s}_{i,j}^2=\left(\mathbb{E}\left[\frac{1}{\sigma_i^2}\right](L_{i,j}+\mathbb{E}[\lambda_i])\right)^{-1}$$
$$\mathbb{E}[\mu_{i,j}]=\widehat{\mu}_{i,j},\ \mathbb{E}[\mu_{i,j}^2]=\widehat{\mu}_{i,j}+\widehat{s}_{i,j}^2$$

Here, $\mathbb{E}[\,\cdot\,]$ denotes the expected value of a random variable.

Maximization step (M-step):

Given an approximate posterior distribution found on the E-step, the M-step proceeds by updating the hyper-parameters $\boldsymbol{\theta}$ as follows:

- Updating μ_0:

$$\mu_0 = \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}[m_i]$$

- Updating σ_0^2:

$$\sigma_0^2 = \frac{1}{T}\sum_{i=1}^{T}(\mathbb{E}[m_i]-\mu_0)^2$$

- Updating α_λ and β_λ:

$$\alpha_\lambda, \beta_\lambda = \text{arg max}_{\alpha,\beta} T\Big(\alpha\log\beta - \log\Gamma(\alpha)\Big) + (\alpha-1)\sum_{i=1}^{T}\mathbb{E}[\log\lambda_i] - \beta\sum_{i=1}^{T}\mathbb{E}[\lambda_i]$$

- Updating a_σ and b_σ:

$$a_\sigma, b_\sigma = \text{arg max}_{a,b} T\Big(a\log b - \log\Gamma(a)\Big) + (a-1)\sum_{i=1}^{T}\mathbb{E}[\log\sigma_i^2] - b\sum_{i=1}^{T}\mathbb{E}\left[\frac{1}{\sigma_i^2}\right]$$

The last two updates are two-dimensional convex optimization problems. Their solutions can be obtained using numerical optimization algorithms specialized to these tasks (Minka, 2002; Llera and F. Beckmann, 2016). Our approach is a less elaborate version of Minka (2002) where we use general-purpose root-finding algorithms which can be found in any commonly-adopted mathematical library.

The EM algorithm repeats the E- and M-steps outlined above until the convergence. In our experiments we found that a few iterations are sufficient to reach a point where any further iterations do not substantially change the values of hyper-parameters.

Appendix B. Score Distribution of the model is approximately Gaussian

In the sequel we show how to obtain the marginal distribution of the observations

$$p(s) = \int p(s|\mu,\sigma^2)p(\mu|m,\lambda,\sigma^2)p(m)p(\lambda)p(\sigma^2)\,\mathrm{d}\mu\,\mathrm{d}m\,\mathrm{d}\lambda\,\mathrm{d}\sigma^2$$

by integrating out all the latent variables in the model one-by-one. We begin by noting that convolution of two Gaussians is another Gaussian with summed variances, to integrate out μ:

$$p(s) = \int \mathcal{N}(s|m,\sigma^2+\sigma^2/\lambda)\mathcal{N}(m|\mu_0,\sigma_0^2)\text{Gam}(\lambda|\alpha_\lambda,\beta_\lambda)\text{InvGam}(\sigma^2|a_\sigma,b_\sigma)\,\mathrm{d}m\,\mathrm{d}\lambda\,\mathrm{d}\sigma^2$$

Further, since the inverse gamma distribution is a conjugate prior for Gaussian distribution with fixed mean, we arrive at the following:

$$p(s) = \int t_{2a_\sigma}\Big(s|m, b_\sigma/a_\sigma(1+1/\lambda)\Big)\mathcal{N}(m|\mu_0,\sigma_0^2)\text{Gam}(\lambda|\alpha_\lambda,\beta_\lambda)\,\mathrm{d}m\,\mathrm{d}\lambda$$

where $t_\nu(s|\eta,\varsigma^2)$ denotes the non-standardized t-distribution with ν degrees of freedom, mean η and variance ς^2. Since the t-distribution can be closely approximated by a Gaussian distribution, which is its limiting case when $\nu\to\infty$, even for moderate values of ν, we can approximate the score distribution by a continuous mixture of Gaussians with gamma as the mixing distribution:

$$p(s) \approx \int \mathcal{N}\Big(s|\mu_0, \sigma_0^2+b_\sigma/a_\sigma(1+1/\lambda)\Big)\text{Gam}(\lambda|\alpha_\lambda,\beta_\lambda)\,\mathrm{d}\lambda$$

Note that the distributions inside the integral resemble a conjugate pair, which would lead to $p(s)$ being the t-distribution. Therefore, we speculate that the distribution $p(s)$ can be roughly approximated by a Gaussian. In fact, our simulations indicate that sampling scores from the model results in bell curve shaped histograms. The analysis above reveals a potential limitation of the proposed model – the assumption that the distribution is symmetric around the mean.

References

Alvin, M.P., Martin, A., 2004. NIST speaker recognition evaluation chronicles. In: Proceedings of the Odyssey 2004, The Speaker and Language Recognition Workshop.
Bishop, C.M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.

Brümmer, N., du Preez, J., 2006. Application-independent evaluation of speaker detection. Comput. Speech Lang. 20 (2), 230–275.
Brümmer, N., Silnova, A., Burget, L., Stafylakis, T., 2018. Gaussian meta-embeddings for efficient scoring of a heavy-tailed PLDA model. In: Proceedings of Odyssey 2018. International Speech Communication Association, pp. 349–356.
Brümmer, N., de Villiers, E., 2010. The speaker partitioning problem. In: Proceedings of the Odyssey. ISCA.
Buntine, W.L., 1994. Operations for learning with graphical models. J. Artif. Int. Res. 2 (1), 159–225.
Carlini, N., Mishra, P., Vaidya, T., Zhang, Y., Sherr, M., Shields, C., Wagner, D., Zhou, W., 2016. Hidden voice commands. In: Proceedings of the 25th USENIX Security Symposium (USENIX Security 16). Austin, TX.
Chung, J.S., Nagrani, A., Zisserman, A., 2018. VoxCeleb2: deep speaker recognition. In: Proceedings of the INTERSPEECH.
Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P., 2011. Front-end factor analysis for speaker verification. IEEE Trans. Audio Speech & Lang. Process. 19 (4), 788–798.
Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. J. Royal Stat. Soc. Ser. B 39 (1), 1–38.
Doddington, G.R., Przybocki, M.A., Martin, A.F., Reynolds, D.A., 2000. The NIST speaker recognition evaluation - overview, methodology, systems, results, perspective. Speech Commun. 31 (2–3), 225–254.
Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., Rubin, D., 2013. Bayesian Data Analysis, third edition, Chapman and Hall/CRC, London.
González Hautamäki, R., Sahidullah, M., Hautamäki, V., Kinnunen, T., 2017. Acoustical and perceptual study of voice disguise by age modification in speaker verification. Speech Commun. 95, 1–15.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. The MIT Press.
Greenberg, C.S., Martin, A.F., Barr, B.N., Doddington, G.R., 2011. Report on performance results in the NIST 2010 speaker recognition evaluation. In: Proceedings of the Interspeech, pp. 261–264. Florence, Italy.
Hansen, J.H.L., Nandwana, M.K., Shokouhi, N., 2017. Analysis of human scream and its impact on text-independent speaker verification. J. Acoust. Soc. Am. 141 (4), 2957–2967.
ISO/IEC 30107-1:2016, 2016. Information technology–Biometric presentation attack detection – Part 1: framework. https://www.iso.org/obp/ui/#iso:std:iso-iec:30107:-1:ed-1:v1:en. (Online; accessed 06 May 2019).
Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K., 1999. An introduction to variational methods for graphical models. Mach. Learn. 37 (2), 183–233.
Kenny, P., 2010. Bayesian speaker verification with heavy-tailed priors.
Kinnunen, T., Lee, K.A., Delgado, H., Evans, N., Todisco, M., Sahidullah, M., Yamagishi, J., Reynolds, D.A., 2018. t-DCF: a detection cost function for the tandem assessment of spoofing countermeasures and automatic speaker verification. In: Proceedings of the Odyssey 2018: The Speaker and Language Recognition Workshop, pp. 312–319.
Larson, J., Menickelly, M., Wild, S.M., 2019. Derivative-free optimization methods. Acta Numer. 28, 287–404.
Lau, Y., Wagner, M., Tran, D., 2004. Vulnerability of speaker verification to voice mimicking. In: Proceedings of the International Symposium on Intelligent Multimedia, Video & Speech Processing (ISIMP'2004), pp. 145–148. Hong Kong.
Lim, M., Yuen, P.C., 2016. Entropy measurement for biometric verification systems. IEEE Trans. Cybern. 46 (5), 1065–1077.
Llera, A., F. Beckmann, C., 2016. Estimating an inverse gamma distribution. https://arxiv.org/abs/1605.01019.
Minka, T., 2002. Estimating a gamma distribution. https://tminka.github.io/papers/minka-gamma.pdf.
Nagrani, A., Chung, J.S., Zisserman, A., 2017. Voxceleb: a large-scale speaker identification dataset. In: Proceedings of the Interspeech 2017, pp. 2616–2620.
Nautsch, A., Rathgeb, C., Saeidi, R., Busch, C., 2015. Entropy analysis of i-vector feature spaces in duration-sensitive speaker recognition. In: Proceedings of the IEEE ICASSP. Brisbane, Queensland, Australia, pp. 4674–4678.
Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al., 2011. The Kaldi speech recognition toolkit. Technical Report.
Prince, S.J.D., Aghajanian, J., Mohammed, U., Sahani, M., 2007. Latent identity variables: Biometric matching without explicit identity estimation. In: Proceedings of the Advances in Biometrics, International Conference, ICB 2007, Seoul, Korea, August 27–29, 2007, pp. 424–434.
Rainforth, T., Cornish, R., Yang, H., Warrington, A., Wood, F., 2018. On nesting Monte Carlo estimators. In: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning. PMLR, Stockholmsmässan, Stockholm Sweden, pp. 4267–4276.
Ranganath, R., Gerrish, S., Blei, D.M., 2014. Black box variational inference. In: Proceedings of the 17th International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22–25, 2014, pp. 814–822.
Ratha, N.K., Connell, J., Bolle, R.M., 2001. Enhancing security and privacy in biometrics-based authentication systems. IBM Syst. J. 40 (3), 614–634.
Reynolds, D.A., 1995. Speaker identification and verification using Gaussian mixture speaker models. Speech Commun. 17 (1–2), 91–108.
Robbins, H., Monro, S., 1951. A stochastic approximation method. Ann. Math. Stat. 22 (3), 400–407.
Robert, C.P., Casella, G., 2005. Monte Carlo Statistical Methods (Springer Texts in Statistics). Springer-Verlag, Berlin, Heidelberg.
Sadjadi, S.O., Kheyrkhah, T., Tong, A., Greenberg, C.S., Reynolds, D.A., Singer, E., Mason, L.P., Hernandez-Cordero, J., 2017. The 2016 NIST speaker recognition evaluation. In: Proceedings of the Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20–24, 2017, pp. 1353–1357.
Sahidullah, M., Delgado, H., Todisco, M., Kinnunen, T., Evans, N.W.D., Yamagishi, J., Lee, K.A., 2018. Introduction to voice presentation attack detection and recent advances. In: Marcel, S., Nixon, M., Fierrez, J., Evans, N. (Eds.), Handbook of Biometric Anti-Spoofing: Presentation Attack Detection. Springer. https://arxiv.org/abs/1901.01085.
Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S., 2018. X-vectors: Robust DNN embeddings for speaker recognition. In: Proceedings of the IEEE ICASSP. Calcary, Canada, pp. 5329–5333.
Vestman, V., Kinnunen, T., Hautamäki, G., Sahidullah, M., 2020. Voice mimicry attacks assisted by automatic speaker verification. Comput. Speech Lang. 59, 36–54.
Wang, C., Blei, D.M., 2013. Variational inference in nonconjugate models. J. Mach. Learn. Res. 14 (1), 1005–1031.
Wu, J.C., Martin, A.F., Greenberg, C.S., Kacker, R., 2017. The impact of data dependence on speaker recognition evaluation. IEEE/ACM Trans. Audio Speech Lang. Process. 25, 5–18.
Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., Xu, W., 2017. Dolphinattack: inaudible voice commands. In: Proceedings of the ACM Conference on Computer and Communications Security. ACM, pp. 103–117.

Paper **IX**

Odyssey 2020 The Speaker and Language Recognition Workshop
1-5 November 2020, Tokyo, Japan

Neural i-vectors

Ville Vestman[1], *Kong Aik Lee*[2], *Tomi H. Kinnunen*[1]

[1]Computational Speech Group, University of Eastern Finland, Finland
[2]Biometrics Research Laboratories, NEC Corporation, Japan

vvestman@cs.uef.fi, kongaik.lee@nec.com, tkinnu@cs.uef.fi

Abstract

Deep speaker embeddings have been demonstrated to outperform their generative counterparts, i-vectors, in recent speaker verification evaluations. To combine the benefits of high performance and generative interpretation, we investigate the use of deep embedding extractor and i-vector extractor in succession. To bundle the deep embedding extractor with an i-vector extractor, we adopt aggregation layers inspired by the Gaussian mixture model (GMM) to the embedding extractor networks. The inclusion of GMM-like layer allows the discriminatively trained network to be used as a provider of sufficient statistics for the i-vector extractor to extract what we call *neural i-vectors*. We compare the deep embeddings to the proposed neural i-vectors on the Speakers in the Wild (SITW) and the Speaker Recognition Evaluation (SRE) 2018 and 2019 datasets. On the core-core condition of SITW, our deep embeddings obtain performance comparative to the state-of-the-art. The neural i-vectors obtain about 50% worse performance than the deep embeddings, but on the other hand outperform the previous i-vector approaches reported in the literature by a clear margin.

1. Introduction

Automatic speaker verification (ASV) systems extract speaker-related information from a pair of speech recordings (enrollment and test) to decide whether the speakers in the two recordings are the same. This is done by computing similarity score between speaker-related features in the two recordings. While the base features have remained the same for decades [1], extraction and comparison of speaker traits from these features has coevolved with advances in machine learning. Much of ASV research has focused on modeling low-level speech feature distributions via *Gaussian mixture models* (GMMs) [2, 3, 4, 5]. Common to models such as GMM with *universal background model* (GMM-UBM) [3], *joint factor analysis* (JFA) [4] and *i-vector* [5] is the use of GMM to model acoustic features within recording(s).

What has changed throughout the years, however, is how speaker comparison is carried out. In the classic GMM pipelines [2, 3], features in the enrollment utterance(s) are used to train a speaker-dependent GMM, and comparison consists of evaluating the likelihoods of the target speaker model and the UBM to form an average log-likelihood ratio over all frames. In contrast to these *frame-based* approaches, the modern approach is to first represent the enrollment and test utterances as vectors of the same dimensionality. They can then be compared using a simple inner product, or a trainable classifier [6]. How these vectors are defined (and called) has changed throughout the years. The early approaches, driven by the success of GMMs, used high-dimensional GMM *supervectors* [7] with inner product scoring, typically implemented using *support vector machines* (SVMs). Through base work in [4], this was followed up by the highly-successful *i-vector* framework [5] where GMM supervectors are presented as points in a low-dimensional latent subspace. Following trends in deep learning, the focus has recently shifted towards deep neural network (DNN) based features [8], called nowadays *embeddings*. The idea to represent utterances as vectors, however, is the same as before, with the same back-end classifiers [9] used with GMM- and neural network based embeddings.

As the title suggests, we focus on i-vector extraction along the lines of classic GMM-based pipelines, but with a 'neural twist'. The general idea, of course, is not new. The three building blocks of any GMM-based method are (a) **a frame-level feature extractor** (*e.g.* MFCC extractor), (b) **a dictionary** (*e.g.* a UBM), and (c) **a posterior estimator** (*e.g.* feature vector alignment to dictionary components), each of which has been successfully replaced in prior work by their neural versions [10, 11]. In contrast to these studies that have focused either on replacing one or two of the components only, or using GMM-inspired components [12, 13, 14] to implement neural embedding extractors, we obtain all the three as 'side-products' from a neural network and proceed with conventional i-vector extractor training on top of them. Noting that (a), (b) and (c) are the only needed building blocks of *any* GMM-based embedding — be it a GMM-UBM [3], JFA [4], GMM-supervector [7], or i-vector [5] — this opens up a pathway to re-address any of the classic pipelines, still respecting the undeniable performance gains demonstrated by the recent neural approaches.

Our focus on i-vectors is arbitrary and the goal of our work is *not* to improve upon state-of-the-art in deep neural network based speaker embeddings. Instead, we aim to demonstrate that classic GMM-based ASV pipelines may not be inferior because of their model structures *per se*, but in the adoption of generic (nondiscriminative) elements. Classic frame-based GMM approaches have certain, nearly forgotten advantages, such as the ability to provide 'partial' scores at a fine temporal scale — the frame level. This might be particularly useful in speaker diarization (not addressed here) and speaker recognition from short utterances. Even if DNN embeddings appear to perform well in short duration ASV tasks [15], we argue that using GMMs retains all the benefits of generative modeling, such as the possibility to do sampling and obtaining uncertainty estimates for features and speaker embeddings. These add up to transparency and explainability demanded with increasing frequency from any machine learning system.

2. Modern speaker embedding extractors

Deep neural networks used for extracting speaker discriminative embeddings typically consist of three main parts (see Figure 1). The first part of the network operates on *frame-level*

10.21437/Odyssey.2020-10

features as an input in order to construct discriminative features from short time contexts, ranging from 100 milliseconds up to a few seconds. The frame-level layers are followed by the second main component, *temporal aggregation layer*, which converts the variable length input feature vector sequence to a fixed-dimensional representation. Finally, the last part of the network, which consists of one or two feedforward layers and the output layer, acts as a *classifier* for speaker identities. The speaker embeddings are usually extracted from the first fully connected layer after the aggregation layer [8].

Each of the three main parts can be implemented in multiple different ways. The frame-level component is often implemented as 1D convolutional neural network (CNN) [16], 2D CNN [17], or as some variant of time-delay neural network [18]. In 1D CNN, the convolution kernel slides over the temporal dimension (frames), whereas in 2D convolution, the kernel slides over both time and frequency dimensions.

There are two commonly used approaches for temporal aggregation. In the first approach [8], relatively *high-dimensional* features are obtained from CNN/TDNN, which are then aggregated by computing the (sample) mean and the standard deviation of the feature vectors over time. The output of the aggregation layer is then formed by concatenating the mean and standard deviation vectors. The second approach [12] of aggregation assumes relatively *low-dimensional* features (akin to conventional hand-crafted acoustic features) from CNN/TDNN but assigns them into multiple clusters (see Figure 2). Here, the aggregation is performed for each cluster separately, resulting in locally aggregated descriptor vectors. Finally, the locally aggregated descriptors are concatenated to form a higher-dimensional residual vector. This approach is analogous to the process how GMM mean supervectors are formed in the GMM-UBM framework.

3. Cluster-wise temporal aggregation

We focus on cluster-wise temporal aggregation methods as they offer a natural pathway to utilize GMM-based speaker verification approaches, such as the i-vector approach, together with discriminatively trained features. In the following, we consider two recent aggregation methods known as *learnable dictionary encoder* (LDE) [19, 12] and *NetVLAD* [20, 13], where VLAD is an acronym for "vector of locally aggregated descriptors". As we will show below, both can be regarded as discriminatively trained GMM-supervector [7] encoders with specific assumptions.

Let us first recall the formula for *posterior* computation of a Gaussian mixture component given a feature vector $\boldsymbol{x}_t$ (time index $t = 1, \ldots, T$). By letting $\boldsymbol{\theta} = \{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, w_c\}_{c=1}^{C}$ be a GMM of C components with mean vectors $\boldsymbol{\mu}_c$, covariance matrices $\boldsymbol{\Sigma}_c$, and component weights w_c, we can compute the posteriors as follows:

$$\gamma_{c,t} = P(c|\boldsymbol{x}_t) = \frac{w_c \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)}{\sum_{l=1}^{C} w_l \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}, \quad c = 1, \ldots, C. \tag{1}$$

By denoting

$$\beta_c = \log\left(\frac{w_c}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_c|}}\right), \tag{2}$$

where D is the dimension of feature vectors, we can expand (1) to form

$$\gamma_{c,t} = \frac{\exp\left[-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_c)^\mathsf{T} \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_c) + \beta_c\right]}{\sum_{l=1}^{C} \exp\left[-\frac{1}{2}(\boldsymbol{x}_t - \boldsymbol{\mu}_l)^\mathsf{T} \boldsymbol{\Sigma}_l^{-1}(\boldsymbol{x}_t - \boldsymbol{\mu}_l) + \beta_l\right]}. \tag{3}$$

Table 1: *Comparison of LDE and NetVLAD.*

Computation step	LDE	NetVLAD
Posterior computation	Eq. (4)	Eq. (7)
Cluster-wise representations	Eq. (5)	Eq. (8)
Supervector normalization	—	Length-norm

3.1. Learnable dictionary encoder

Equation (3) holds for any GMM with unrestricted covariance matrices. In the following, we consider special cases, where the covariance matrices are restricted to have specific forms. First, by assuming *isotropic* covariance matrices (*i.e.*, $\boldsymbol{\Sigma}_c = s_c\mathbf{I}$, with $s_c > 0$), (3) becomes

$$\gamma_{c,t} = \frac{\exp\left[-\frac{1}{2}s_c\|\boldsymbol{x}_t - \boldsymbol{\mu}_c\|^2 + \beta_c\right]}{\sum_{l=1}^{C} \exp\left[-\frac{1}{2}s_l\|\boldsymbol{x}_t - \boldsymbol{\mu}_l\|^2 + \beta_l\right]}, \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm. This is the formulation used for posterior computation in [18] with LDE. Earlier works on LDE [19, 12], did not include the bias terms β_c. Both the parameters s_c that define the isotropic covariance matrices, and the bias terms β_c as well as the cluster centroids $\boldsymbol{\mu}_c$ are learnable parameters of the LDE layer.

After computing the posteriors, the construction of the output of LDE layer is a two step process. First, the input features are temporally aggregated with respect to each cluster. This is done by computing the weighted means $\boldsymbol{m}_c$ of residuals $\boldsymbol{\mu}_c - \boldsymbol{x}_t$ around the cluster centroids for each cluster c:

$$\boldsymbol{m}_c = \frac{\sum_{t=1}^{T} \gamma_{c,t}(\boldsymbol{\mu}_c - \boldsymbol{x}_t)}{\sum_{t=1}^{T} \gamma_{c,t}}. \tag{5}$$

The second step is to concatenate the cluster-wise representations to form a supervector $\boldsymbol{m} = (\boldsymbol{m}_1^\mathsf{T}, \boldsymbol{m}_2^\mathsf{T}, \ldots \boldsymbol{m}_T^\mathsf{T})^\mathsf{T}$, which is the output of the LDE layer.

While the original formulation of LDE uses separate *isotropic* covariance matrices for each component, it is straightforward to modify the LDE layer to operate with *diagonal* covariance matrices, or to use one *shared diagonal* or *spherical* covariance matrix for all components. In our experiments, we consider only the shared diagonal matrix formulation besides the original formulation with non-shared spherical covariances to limit the computational burden.

3.2. NetVLAD encoder

Let us next assume *shared full covariance matrices* (*i.e.*, $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}\ \forall c$), which will lead to the NetVLAD formulation of posterior computation. The shared covariance assumption simplifies (3) to

$$\gamma_{c,t} = \frac{\exp\left[\boldsymbol{\mu}_c^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_t + \log(w_c) - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c\right]}{\sum_{l=1}^{C} \exp\left[\boldsymbol{\mu}_l^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}_t + \log(w_l) - \frac{1}{2}\boldsymbol{\mu}_l^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_l\right]}, \tag{6}$$

which allows us to write

$$\gamma_{c,t} = \frac{\exp\left[\boldsymbol{\omega}_c^\mathsf{T}\boldsymbol{x}_t + \psi_c\right]}{\sum_{l=1}^{C} \exp\left[\boldsymbol{\omega}_l^\mathsf{T}\boldsymbol{x}_t + \psi_l\right]}, \tag{7}$$

where

$$\boldsymbol{\omega}_c = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c \quad \text{and} \quad \psi_c = \log(w_c) - \frac{1}{2}\boldsymbol{\mu}_c^\mathsf{T}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c.$$

Equation (7) can be implemented as an affine transform followed by softmax operation over clusters, which is exactly what is done in NetVLAD layer to compute the posteriors. The NetVLAD layer has $\boldsymbol{\omega}_c$, ψ_c, and $\boldsymbol{\mu}_c$ as its learnable parameters.

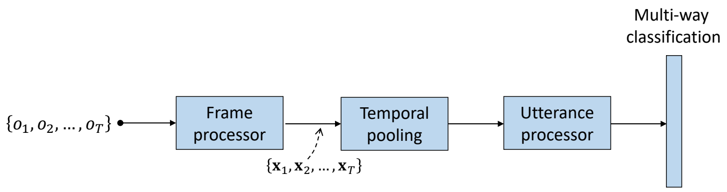

Multi-way classification
$\{o_1, o_2, \ldots, o_T\}$
Frame processor
Temporal pooling
Utterance processor
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$

Figure 1: *An x-vector extractor consists of three functional blocks: a frame-level processor, a temporal pooling layer, and classifier. X-vector embeddings are derived from the affine transformation after the pooling layer.*

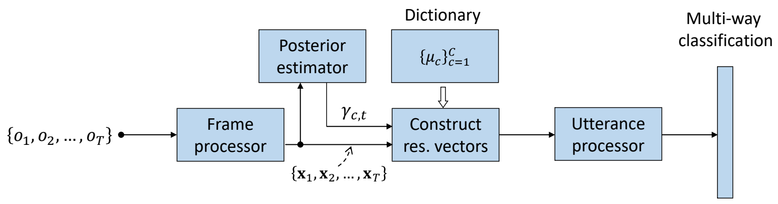


Figure 2: *The centrepiece of learnable dictionary encoder (LDE) and NetVLAD is the frame processor, frame posterior estimator and dictionary that are trained jointly to minimize a classification loss.*

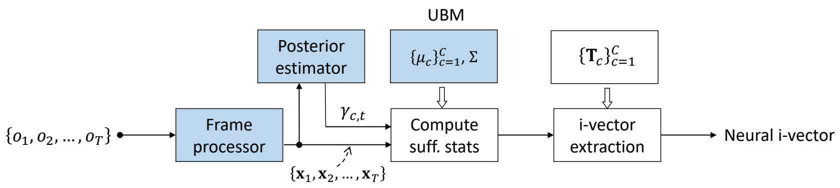

UBM
Posterior estimator
$\{\mu_c\}_{c=1}^C, \Sigma$
$\{\mathbf{T}_c\}_{c=1}^C$
$\gamma_{c,t}$
$\{o_1, o_2, \ldots, o_T\}$
Frame processor
Compute suff. stats
i-vector extraction
Neural i-vector
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$

Figure 3: *The proposed neural i-vector relies on a deep structured front-end (shaded boxes) to extract sufficient statistics, which are then used for generative embedding.*

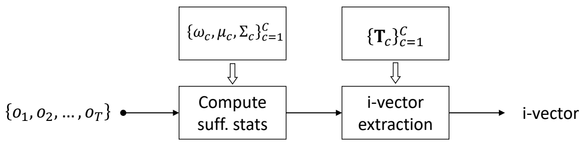

$\{\omega_c, \mu_c, \Sigma_c\}_{c=1}^C$
$\{\mathbf{T}_c\}_{c=1}^C$
$\{o_1, o_2, \ldots, o_T\}$
Compute suff. stats
i-vector extraction
i-vector

Figure 4: *An i-vector extractor is built upon a Universal Background Model (UBM) defined by the parameter set consists of weights, mean vectors, and covariance matrices.*

In terms of the number of learnable parameters, the correspondence between NetVLAD and GMM with shared covariances is not exact as covariance matrix $\mathbf{\Sigma}$ contains $D(D+1)/2$ free parameters, whereas a matrix containing all $\boldsymbol{\omega}_c$ vectors has CD parameters.

In NetVLAD, the construction of the output supervector differs from LDE in two ways. First, NetVLAD length-normalizes the component-wise outputs:

$$\boldsymbol{m}_c = \frac{\sum_{t=1}^{T} \gamma_{c,t}(\boldsymbol{\mu}_c - \boldsymbol{x}_t)}{\left\| \sum_{t=1}^{T} \gamma_{c,t}(\boldsymbol{\mu}_c - \boldsymbol{x}_t) \right\|}. \tag{8}$$

The second difference is that the supervector $\boldsymbol{m}$, obtained by concatenating the cluster-wise outputs $\boldsymbol{m}_c$, is further length-normalized to unit sphere.

The differences between LDE and NetVLAD are summarized in Table 1. As LDE and NetVLAD differ in the posterior computation as well as whether or not length-normalizations are applied, it is challenging to identify the potential causes of the performance difference between the two methods (if such difference is to exist). Therefore, we also study a hybrid approach (referred as NetVLAD/LDE), in which the posterior computation follows the NetVLAD approach, while the rest of the steps follow the LDE approach.

4. Utilizing aggregation statistics for i-vector extraction

Deep speaker embedding [8] has been demonstrated to outperform the i-vector representation shown in Figure 4. The enhanced performance is attained by (1) training the network using large amount of training data via *data augmentation*, and (2) *discriminative training* (e.g., multi-class cross entropy cost, angular margin cost [21]). The drawback is lack of generative interpretation. We propose to combine the benefits from both sides, leading to the so-called **neural i-vector** shown in Figure 3. In the neural i-vector, we utilize the features, posterior estimator, and the UBM that all have been trained discriminatively using speaker labels. This differs from the DNN i-vector presented in [11] as it requires senone labels and does not utilize discriminatively trained features.

To extract neural i-vectors, we do not compute (5) or (8), but instead compute the suffiecient statistics in a standard way [22] as follows:

$$z_c = \sum_{t=1}^{T} \gamma_{c,t}, \quad (9)$$

$$\boldsymbol{f}_c = \sum_{t=1}^{T} \gamma_{c,t} \boldsymbol{x}_t, \quad (10)$$

$$\mathbf{S}_c = \sum_{t=1}^{T} \gamma_{c,t} \boldsymbol{x}_t \boldsymbol{x}_t^{\mathsf{T}}. \quad (11)$$

Here the features and posteriors are extracted from the embedding extractor network. The obtained statistics can be then easily used with any available i-vector code to train the i-vector extractor and to extract the i-vectors.

5. Speaker verification experiments

5.1. Network architectures and training procedure

The network architectures designed for this study are all derived from the standard x-vector architecture presented in [8]. Our most elementary architecture differs from [8] in the following ways:

- As in [16], we use *non-dilated* 1D CNN instead of TDNN with dilations used in [8].
- As in [16], we use *leaky* rectified linear unit (LReLU) activations (with slope of 0.01) instead of ReLUs.
- We have only one embedding layer (rather than two) after the aggregation layer. In our preliminary experiments, we did not find adding another layer to decrease the resulting speaker verification equal error rates.

We extend our default network (referred as TDNN) by adding *squeeze-and-excitation* (SE) modules [23] to the TDNN layers. The SE module aims to improve the representative power of hidden features by reweighting them using information from global temporal statistics of features. Using the terminology of [24], we adopt SE modules to perform *temporal squeeze* and *channel (feature) excitation*. That is, the output features of 1D CNN layer are weighted by factors computed from temporally pooled (non-weighted) features. Our implementation of the SE module is depicted in Figure 5, while Figure 6 illustrates how the SE module is added to the TDNN layer. The resulting TDNN-SE network architecture is presented in Table 2. Our SE module differs from the original in that it computes standard deviations in addition to means during the squeeze phase. In the excitation phase, we add batch normalization between the fully connected layers, as shown in Figure 5.

Inspired by the widely used ResNet architecture [25], our next network variant includes *residual* modules. Our implementation of a residual module (referred as TDNN-RES-SE) includes a fully connected layer, a 1D convolutional layer, and a SE module, as depicted in Figure 7. The network architecture is shown in Table 3. We replace neither the first nor the last TDNN-SE layer with the residual modules, as residual modules require the number of input and output features to be the same. The first layer has relatively low-dimensional MFCCs as its input, while the output size of the last layer depends on the aggregation method used. Networks with the mean and standard deviation pooling produce 1500-dimensional feature vectors at the output of the last TDNN layer. Networks with LDE or NetVLAD, in turn, produce TDNN outputs of 128-dimensions. With LDE and NetVLAD, we use 64 clusters, resulting in 8192-dimensional output vectors from the aggregation layer.

Table 2: *The architecture of TDNN-SE network.*

#	Layer type	CNN kernel size	Output dim.
1	TDNN-SE	5	512
2	TDNN-SE	5	512
3	TDNN-SE	7	512
4	TDNN-SE	1	512
5	TDNN-SE	1	1500
6	Aggregation	—	3000
7	FC-LReLU-BN	—	512
8	FC-softmax	—	#speakers

Table 3: *The architecture of TDNN-RES-SE network. The output sizes of the last TDNN-SE layer and the aggregation layer depend on the aggregation method. If aggregation using means and standard deviations is used, these sizes are* 1500 *and* 3000, *but if LDE or NetVLAD is used, the sizes are* 128 *and* 8192.

#	Layer type	CNN kernel size	Output dim.
1	TDNN-SE	5	512
2,3	TDNN-RES-SE	5	512
4,5	TDNN-RES-SE	7	512
6,7	TDNN-RES-SE	1	512
8	TDNN-SE	1	1500/128
9	Aggregation	—	3000/8192
10	FC-LReLU-BN	—	512
11	FC-softmax	—	#speakers

All our networks are implemented with PyTorch [26]. The Kaldi toolkit [27] is used to extract speech activity labels and 60-dimensional MFCCs (without delta features), used as the input features. PyKaldi [28] is used to load the features in Kaldi format in Python and to perform *cepstral mean normalization* (CMS) for the features.

For network training, we use four second long segments selected from random positions of the training utterances. During training, we feed about 14 000 short segments from each training speaker to the network in minibatches of size 64. Network weights are updated to minimize cross-entropy loss using stochastic gradient descend optimizer with weight decay parameter set to 0.001. We use a learning rate schedule that decreases the learning rate from 0.05 to 0.0002 during the training.

5.2. Neural i-vector training details

We utilized the *augmented* form of i-vector extractor as described in [29][1]. In the augmented form, the UBM mean vectors are augmented into the first column of the *total variability matrix* $\mathbf{T}$ and they are thus updated after the each iteration of extractor training, unlike in the standard formulation. For modeling residual covariances in the total variability model, we used a diagonal covariance matrix that was shared between all components. To initialize the first column of $\mathbf{T}$ and the residual covariance matrix, we used means and covariances computed from the sufficient statistics (9), (10), and (11) of the training data. We set the i-vector dimension to 512, which is the same as the dimension of the network embeddings.

[1]The PyTorch re-implementation of Kaldi's i-vector extractor used in our study is available at `https://github.com/vvestman/pytorch-ivectors`

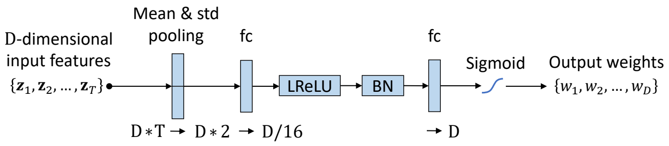

Mean & std pooling
D-dimensional input features
$\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T\}$
fc
LReLU
BN
fc
Sigmoid
Output weights
$\{w_1, w_2, \ldots, w_D\}$
D∗T → D∗2 → D/16
→ D

Figure 5: *The squeeze-and-excitation (SE) module. The output weights are used to weight the input features as shown in Figs. 6 and 7.*

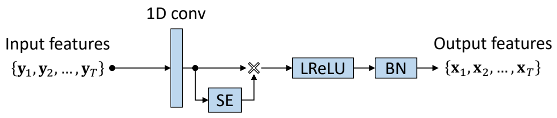

1D conv
Input features
$\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$
SE
LReLU
BN
Output features
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$

Figure 6: *A TDNN module with squeeze-and-excitation (SE). This module is used to build the frame processor of TDNN-SE network.*

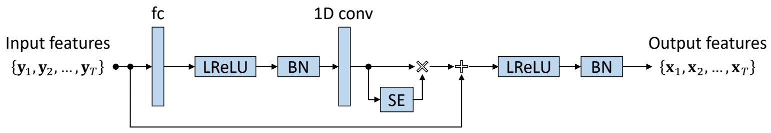

fc
1D conv
Input features
$\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$
LReLU
BN
SE
LReLU
BN
Output features
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$

Figure 7: *A Residual module with squeeze-and-excitation (SE). This module is used to build the frame processor of TDNN-RES-SE network. Compared to the module in Figure 6, this module adds a fully connected layer (fc) and a residual connection. The residual connection adds input features to the features obtained from the SE operation.*

5.3. Training data

To train the neural networks, i-vector extractors, and scoring back-ends, we used 16 kHz speech data from VoxCeleb1 [30] and VoxCeleb2 [17]. VoxCeleb data has been collected from YouTube by automatic means. Like in [18], we concatenated all the segments that were extracted from the same YouTube source video, and used these concatenated segments as the training data. We excluded all concatenated segments less than six seconds long. After filtering out the short segments, we were left with data extracted from 149 754 unique YouTube videos, containing 7365 speakers. This data was then augmented five-fold using Kaldi's augmentation recipe, resulting in total of 748 770 concatenated segments. The augmentation creates copies of data by reverberating speech or by adding noise, babble, or music to the speech.

5.4. Evaluation data and metrics

We evaluated all the ASV systems on *Speakers in the Wild* (SITW) [31] and *NIST Speaker Recognition Evaluation* (SRE) 2018 [32] and 2019 [33] data. From SITW, we selected core-core (referred here as 'core') and core-multi (referred as 'multi') conditions. In both, only a single speaker appears in each of the enrollment segments, but in the multi condition, the test utterances may contain speech from multiple speakers (unlike in the core condition). The core condition evaluation contains 721 788 trials, out of which 3658 are target trials. The multi condition contains 2 010 683 trials, out of which 10 045 are target trials.

The SRE 2018 and the SRE 2019 both consist of two separate evaluations. One is based on telephone speech data in Call My Net 2 (CMN2) corpus, while the other one is based on Video Annotation for Speech Technology (VAST) corpus. In this study, we evaluated only the VAST portions of SREs as VAST data is a better match to our VoxCeleb training data. The SRE 2018 evaluation contains 31 815 trials, out of which 315 are target trials, while the SRE 2019 has 67 348 trials, out of which 452 are target trials.

With SREs, we used the diarization labels provided by NIST for the enrollment side to remove the unwanted portions of speech from the enrollment. We did not perform diarization of the test side for any of the datasets.

For each set of evaluation trials, we report *equal error rate* (EER) and normalized minimum detection cost (minDCF). See [34] for details of minDCF. We adopted the same minDCF parameters as used in SRE 2018 and 2019 evaluations. That is, we set the costs of miss and false alarm equal to one ($C_{\text{miss}} = C_{\text{fa}} = 1$), and the target prior P_{target} to 0.05.

5.5. Scoring back-end

We centered, whitened, and length-normalized (both discriminative and generative) speaker embeddings before simplified PLDA scoring [35]. We did not apply domain adaptation techniques, but simply used the training data (VoxCeleb) to compute centering vector and whitening matrix. Finally, we performed *adaptive symmetric score normalization* (AS-norm) [36]. For AS-norm, we randomly selected 2000 utterances from training data and chose 200 highest scoring utterances for each enrollment or test utterance to compute the normalization statistics.

5.6. Speaker verification results

Table 4 shows the results of our experiments with different systems on multiple speaker verification evaluations. The results for the core condition of SITW are the most representative of the basic accuracy of the ASV systems as it does not have multi-speaker utterances requiring diarization. The other evaluations provide supporting evidence, although the results may be impaired by the lack of diarization.

In general, we find that the differences between the results of different deep embedding extractors are small. For example, when migrating from TDNN to TDNN-SE and to TDNN-RES-SE architectures, the results slightly improve on some evaluations, but get slightly worse on others. Similarly, the differences between the different aggregation methods are relatively minor, which is quite intriguing considering the differences between

Table 4: *Speaker verification results for the systems evaluated in this study. In addition to the deep speaker embedding systems, the results are reported for four neural i-vector systems each of which are based on different variations of the aggregation layer.*

	SITW EVAL CORE		SITW EVAL MULTI		SRE18 EVAL VAST		SRE19 EVAL VAST	
	EER	Min Cost	EER	Min Cost	EER	Min Cost	EER	Min Cost
TDNN (mean & std)	2.21	0.135	**3.46**	**0.183**	12.69	**0.472**	5.97	0.223
TDNN-SE (mean & std)	2.02	0.125	4.03	0.188	12.70	0.473	5.97	0.212
TDNN-RES-SE (mean & std)	2.10	0.123	4.07	0.188	12.02	0.477	5.75	0.216
TDNN-RES-SE (LDE, isotropic)	2.02	0.122	4.04	0.185	12.70	0.497	5.53	0.212
↳ Neural i-vector	2.93	0.173	5.55	0.249	15.92	0.588	6.64	0.254
TDNN-RES-SE (LDE, shared diag.)	**1.83**	0.123	4.42	0.189	**11.75**	0.483	5.34	0.213
↳ Neural i-vector	2.81	0.168	5.40	0.246	15.87	0.522	6.43	0.256
TDNN-RES-SE (NetVLAD)	1.94	**0.117**	4.06	0.184	12.38	0.474	**5.31**	**0.208**
↳ Neural i-vector	3.09	0.175	5.73	0.261	16.51	0.588	6.00	0.290
TDNN-RES-SE (NetVLAD/LDE)	2.02	0.129	4.41	0.199	13.40	0.528	5.53	0.229
↳ Neural i-vector	3.06	0.188	5.65	0.262	15.56	0.596	5.97	0.253

Table 5: *Review of recent single system results for SITW core-core condition. Due to different experimental settings and implementations, the results from different approaches are not directly comparable. Out of the i-vector systems, the proposed neural i-vector obtains the lowest EER. The second lowest EER was obtained by an i-vector system using a dereverberation system (WPE) together with perceptual linear prediction (PLP) and stacked bottleneck features (SBN). Other two included i-vector systems use MFCCs and bottleneck features (BNF). Under the divider line are the systems based on deep speaker embeddings. All systems use either MFCCs or filterbank coefficients (FBANK) as input features. All the embedding networks use either TDNN, extended TDNN (E-TDNN), factorized TDNN (F-TDNN), or ResNet34 based architectures. One system uses additive angular margin (AAM) loss instead of standard cross-entropy. The performance differences between the deep embedding extractors are rather small, except for the last system utilizing tied mixture of factor analyzers (TMFA) layer that is trained on 8 kHz Switchboard and SRE data.*

System & study	EER (%)
Neural i-vector [this study]	2.81
WPE PLP+SBN i-vector [37]	3.38
MFCC i-vector [37]	4.40
BNF i-vector [18]	5.77
TDNN-RES-SE (LDE) [this study]	1.83
FBANK E-TDNN [37]	1.70
MFCC E-TDNN [15]	1.7
MFCC F-TDNN [18]	1.86
FBANK ResNet34+LDE (AAM-softmax) [18]	2.11
FBANK ResNet34+TMFA (8 kHz) [14]	5.74

the standard mean and standard deviation aggregation and the dictionary based methods.

Different variants of neural i-vectors perform almost equally well to each other. The performance of neural i-vectors is way behind the performance of their deep embedding counterparts. On the other hand, the neural i-vectors perform substantially better than the other i-vector systems reported in literature as can be observed from Table 5. The table also shows that our deep embeddings obtain a competitive results in comparison to the results reported in the other studies.

5.7. Visualizations of neural i-vectors

In Figure 8, we illustrate *sampled* neural i-vectors for 5 male speakers in the SITW corpus. From each speaker we selected six utterances and computed the posterior distributions [29, eqs. (3) and (4)] of i-vectors. These distributions were used to sample 50 i-vectors per utterance. From the figure, we can observe that different speakers are well separated and that the utterances with short durations have higher uncertainty (*i.e.*, more spread clusters) than the utterances with long durations, as expected.

Finally, in Figure 9, we depict traces of posterior covariances [29, eq. (3)] of i-vectors for SITW data. The traces reflect the uncertainty in the i-vector estimation [39]. As expected, the longer the duration, the less uncertain the i-vectors are.

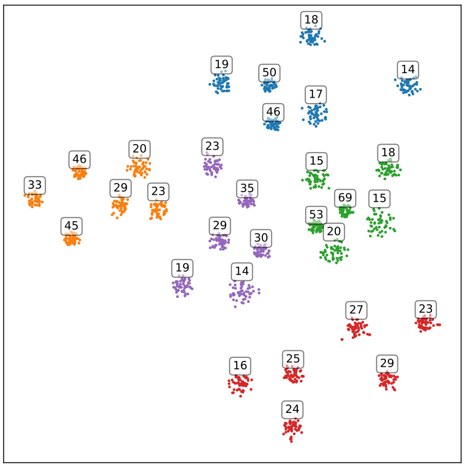


Figure 8: *T-SNE visualization [38] of random neural i-vectors drawn from i-vector posterior distributions of 30 utterances from 5 male speakers in SITW corpus. Different colors represent different speakers. Each of the 30 clusters consists of 50 random i-vectors drawn from the posterior distribution of one i-vector. The numbers show durations of the utterances in seconds after removing non-speech frames. The long utterances have less uncertainty than the short ones, which can be observed from the compactness of the clusters.*

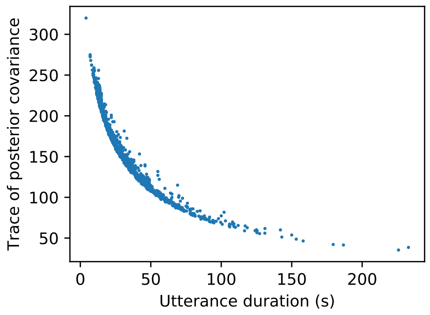


Figure 9: *Trace of i-vector posterior covariance matrix as a function of utterance duration for utterances in SITW core-core condition.*

6. Conclusion

At a broad outlook, the general developments in the field of speaker recognition have involved innovative (and often successful) re-use of previous generation tools to build up next generation recognizers: we have seen steady transition in state-of-the-art from individually trained GMMs to GMM-UBM, GMM supervectors, JFA and i-vectors (in this order). As a community, we have been working on multi-layered (deep) models, formed by stacking frame-level feature processors with utterance-level presentations and speaker latent variable models. Until the recent past, however, these pipelines have not been trained as a whole, but constructed from individually-optimized components. This is where the deep neural networks have come to a rescue, and we are witnessing transition towards the next generation deep models. Nonetheless, deep neural network models have seem to have interrupted the chain of GMM-based systems, particularly as they lack the concept of a universal background model. Some recent work has therefore looked into replacing the global temporal pooling operation of deep embedding extractors with learnable dictionaries, similar to the UBM, with demonstrated improvements.

In an attempt to bridge classic GMM-based technology and the modern deep learning era, we have provided a unified comparison of alternative i-vector extractors that use different variants of deep neural networks to optimize the frame-level features and the UBM. In particular, two recent deep neural network architectures, LDE and NetVLAD, *can be interpreted as GMMs with specific assumptions*. This interpretation enabled us to re-consider classic GMM-based systems using discriminatively obtained features and UBM. As a proof of concept, we decided to focus on the i-vector system, but similar construction is readily applicable to any ASV or diarization system that uses GMMs.

Our results indicate that 'neural i-vectors' outperform all the existing i-vector variants by a wide margin, indicating the importance of using speaker-informative short-term features and speaker-informative dictionary. Even if the corresponding 'purely neural' systems (used for obtaining the components of our i-vector system) outperform the neural i-vector approach, this was *not* the point of our study. The point, instead, is that it is possible to view certain neural architectures as if having a multi-modal aggregator (GMM) built in them. These identified connections may open up fresh ideas in revoking techniques such as uncertainty propagation, data augmentation (by sampling features or speaker embeddings). Potential applications that may benefit from fine-grained frame-by-frame speaker decisions, such as speaker diarization, provide another potential topic of future studies.

This work was partially supported by Academy of Finland (project 309629).

7. References

[1] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, August 1980.

[2] Douglas A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech Communication*, vol. 17, pp. 91–108, August 1995.

[3] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1, pp. 19 – 41, 2000.

[4] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. Audio, Speech & Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[5] Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech and Language Processing*, 2010.

[6] Pavel Matějka, Ondřej Glembek, Fabio Castaldo, Md Jahangir Alam, Oldřich Plchot, Patrick Kenny, Lukáš Burget, and Jan Černocky, "Full-covariance UBM and heavy-tailed PLDA in i-vector speaker verification," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 4828–4831.

[7] William M Campbell, Douglas E Sturim, and Douglas A Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006.

[8] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[9] Simon J. D. Prince and James H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, 2007, pp. 1–8.

[10] Yao Tian, Meng Cai, Liang He, and Jia Liu, "Investigation of bottleneck features and multilingual deep neural networks for speaker verification," in *Proc. Interspeech 2015*, 2015.

[11] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 1695–1699.

[12] Weicheng Cai, Zexin Cai, Xiang Zhang, Xiaoqi Wang, and Ming Li, "A novel learnable dictionary encoding layer

for end-to-end language identification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5189–5193.

[13] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

[14] Nanxin Chen, Jesús Villalba, and Najim Dehak, "Tied mixture of factor analyzers layer to combine frame level representations in neural speaker embeddings," *Proc. Interspeech 2019*, pp. 2948–2952, 2019.

[15] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5796–5800.

[16] Hossein Zeinali, Lukas Burget, Johan Rohdin, Themos Stafylakis, and Jan Honza Cernocky, "How to improve your speaker embeddings extractor in generic toolkits," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6141–6145.

[17] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "VoxCeleb2: deep speaker recognition," in *Proc. Interspeech 2018*, 2018, pp. 1086–1090.

[18] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak, "State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and Speakers in the Wild evaluations," *Computer Speech & Language*, vol. 60, 2020.

[19] Hang Zhang, Jia Xue, and Kristin Dana, "Deep ten: Texture encoding network," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 708–717.

[20] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307.

[21] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 212–220.

[22] Patrick Kenny, "A small footprint i-vector extractor," in *Odyssey*, 2012, vol. 2012, pp. 1–6.

[23] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[24] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger, "Concurrent spatial and channel 'squeeze & excitation' in fully convolutional networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 421–429.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[27] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[28] Dogan Can, Victor R Martinez, Pavlos Papadopoulos, and Shrikanth S Narayanan, "Pykaldi: A python wrapper for Kaldi," in *International Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5889–5893.

[29] Ville Vestman, Kong Aik Lee, Tomi H. Kinnunen, and Takafumi Koshinaka, "Unleashing the Unused Potential of i-Vectors Enabled by GPU Acceleration," in *Proc. Interspeech 2019*, 2019, pp. 351–355.

[30] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "VoxCeleb: A large-scale speaker identification dataset," *Proc. Interspeech 2017*, pp. 2616–2620, 2017.

[31] Mitchell McLaren, Luciana Ferrer, Diego Castan, and Aaron Lawson, "The speakers in the wild (SITW) speaker recognition database.," in *Proc. Interspeech 2016*, 2016, pp. 818–822.

[32] *NIST 2018 Speaker Recognition Evaluation Plan*, 2018 (accessed January 24, 2020), `https://www.nist.gov/system/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf`.

[33] *NIST 2019 Speaker Recognition Evaluation Plan*, 2019 (accessed January 24, 2020), `https://www.nist.gov/system/files/documents/2019/08/16/2019_nist_multimedia_speaker_recognition_evaluation_plan_v3.pdf`.

[34] Seyed Omid Sadjadi, Craig Greenberg, Elliot Singer, Douglas Reynolds, Lisa Mason, and Jaime Hernandez-Cordero, "The 2018 NIST Speaker Recognition Evaluation," in *Proc. Interspeech 2019*, 2019, pp. 1483–1487.

[35] Daniel Garcia-Romero and Carol Y Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Proc. Interspeech 2011*, 2011.

[36] Sandro Cumani, Pier Domenico Batzu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis, "Comparison of speaker recognition approaches for real applications," in *Proc. Interspeech 2011*, 2011.

[37] Pavel Matějka, Oldřich Plchot, Hossein Zeinali, Ladislav Mošner, Anna Silnova, Lukáš Burget, Ondřej Novotný, and Ondřej Glembek, "Analysis of BUT Submission in Far-Field Scenarios of VOiCES 2019 Challenge," in *Proc. Interspeech 2019*, 2019, pp. 2448–2452.

[38] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[39] Amir Hossein Poorjam, Rahim Saeidi, Tomi Kinnunen, and Ville Hautamäki, "Incorporating uncertainty as a quality measure in i-vector based language recognition.," in *Odyssey*, 2016, pp. 74–80.

VILLE VESTMAN

Speech is a fundamental form of human communication. Besides the message, speech contains information about the speaker, allowing recognition of the speaker's identity. This dissertation focuses on automatic speaker recognition with the aid of modern machine learning methods. The work develops new methods to enhance speed, robustness, and security of automatic speaker recognition systems. Speaker recognition has applications in authentication, surveillance, and forensics.

UNIVERSITY OF
EASTERN FINLAND

uef.fi

PUBLICATIONS OF
THE UNIVERSITY OF EASTERN FINLAND
Dissertations in Forestry and Natural Sciences

ISBN 978-952-61-3483-3
ISSN 1798-5668