X-VECTORS: ROBUST NEURAL EMBEDDINGS FOR SPEAKER RECOGNITION

by

David Snyder

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

March 2020

Abstract

Speaker recognition is the task of identifying speakers based on their speech signal. Typically, this involves comparing speech from a known speaker, with recordings from unknown speakers, and making same-or-different speaker decisions. If the lexical contents of the recordings are fixed to some phrase, the task is considered text-dependent, otherwise it is text-independent. This dissertation is primarily concerned with this second, less constrained problem. Since speech data lives in a complex, high-dimensional space, it is difficult to directly compare speakers. Comparisons are facilitated by *embeddings*: mappings from complex input patterns to low-dimensional Euclidean spaces where notions of distance or similarity are defined in natural ways. For almost ten years, systems based on i-vectors–a type of embedding extracted from a traditional generative model–have been the dominant paradigm in this field. However, in other areas of applied machine learning, such as text or vision, embeddings extracted from discriminatively trained neural networks are the state-of-the-art. Recently, this line of research has become very active in speaker recognition as well. Neural networks are a natural choice for this purpose, as they are capable of learning extremely complex mappings, and when training data resources are abundant, tend to outperform traditional methods. In this dissertation, we develop a next-generation neural embedding–denoted by *x-vector*–for speaker recognition. These neural embeddings are demonstrated to substantially improve upon the state-of-the-art on a number of benchmark datasets.

Dissertation Committee

Sanjeev Khudanpur (Advisor)
Associate Professor
Center for Language and Speech Processing
Department of Computer Science
Department of Electrical and Computer Engineering
Johns Hopkins University

Daniel Povey (Co-advisor)[1]
Chief Speech Scientist
Xiaomi Corporation

Philipp Koehn
Professor
Center for Language and Speech Processing
Department of Computer Science
Johns Hopkins University

Najim Dehak
Assistant Professor
Center for Language and Speech Processing
Department Electrical and Computer Engineering
Johns Hopkins University

[1]Daniel Povey was an Assistant Research Professor in the Center for Language and Speech Processing for the majority of my PhD, from 2013–2019.

Acknowledgments

My advisors

I would like to thank my advisors, Dan Povey and Sanjeev Khudanpur for leading me through the PhD journey. Dan, I am grateful to have shared in a small part of the success of Kaldi. Knowing that my open source contributions have had a positive impact in the speech community has been profoundly satisfying. Sanjeev, thank you for your guidance on all aspects of my PhD, and for being a tireless advocate for my research.

My committee

I want to thank my committee members Philipp Koehn, and Najim Dehak. I sincerely appreciate your time and effort reviewing my dissertation work. Najim, thank you for your lively parties, and for your contributions to speech technology, which are foundations of my dissertation work.

My mentors

I would also like to thank Greg Sell, Alan McCree and Daniel Garcia-Romero of the HLTCOE for generously sharing their time and expertise with me. Daniel, thank you for your collaboration and commitment to the x-vector work. The impact of your contributions on the success of this work cannot be understated. Paola Garcia, thank you for so many words of encouragement and for sharing your knowledge of speech technology in industry.

The CLSP community

I am grateful to have been part of one of the best research centers in human language technology. The accomplishments of the CLSP community have been inspiring. Thank you Guoguo and Vijay for your advice at the beginning of my PhD. Thank you labmates: Hainan, Pegah, Matt, Vimal, Xiaohui, Yiming, Ke, Jinyi and Desh and my peers in other labs: Adi, Katie, Ruizhi and Suzanna for your collaboration and friendship over the last six years. I would also like to thank Ruth Scally and Zack Burwell for helping resolve so many administrative challenges over the years.

My industry collaborators

I would like to thank my industry collaborators who have helped me focus on the practical applications of my research. Thank you Ming Sun and Shiv Vitaladevuni for your patient mentorship during my internship at Amazon Alexa. Yishay Carmiel, I am grateful for your collaboration; our work together was crucial for the early success of the x-vector framework.

My family

Finally, I would like to thank my parents for supporting my academic pursuits. I would like to thank my brother, Vernon, for sharing his insights into academia and for helping me navigate the graduate school experience. Most of all, I would like to thank my wife, Katya, for being involved in every decision and sharing in all my successes and failures. In particular, thank you for listening to so many practice talks.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Capturing speaker characteristics is important for speech-enabled technologies that support multiple users and require authentication or personalization. For example, a telephone banking system may use speaker identity to control access or a home device may utilize this information to provide personalized interaction. Tackling these problems involves extracting embeddings from speech segments, which are then compared to produce same-speaker or different-speaker decisions. Since its introduction by Dehak et al. (2011), the i-vector has become the most popular embedding used in speaker recognition, and has also been widely used in related tasks, such as spoken language recognition and speaker diarization. The i-vector framework is described in more detail in Section 2.4.1.

Due to many sources of variability, speaker recognition is a challenging problem. A significant body of work has been devoted to compensating for channel variability: variations due to device (e.g., landline or cellphone) or environment. Historically, the field has focused primarily on telephone speech. However, the rise of speech-enabled home devices and online video audio expose speaker recognition to more unconstrained speech, where degradation due to speaker distance, reverberation, external noises and extremely short recordings is common. Therefore, it is important to develop embeddings (and methods to compare embeddings) that are robust to these sources of variability.

1.1 Contributions

The focus of this dissertation is the development of more powerful embeddings for capturing speaker characteristics. The dissertation proposes the use of deep neural networks that map variable-length speech segments to fixed dimensional representations, called x-vectors. The bulk of this dissertation is concerned with application in text-independent speaker recognition. A considerable, though smaller number of pages are devoted to the related tasks of speaker diarization and spoken language recognition.

The main contributions and findings of this dissertation are:

- The development of a novel embedding, called x-vectors, for text-independent speaker recognition;
- Studies which compare a variety of methods to train the embedding extractor, using different objective functions and data augmentation techniques;
- Compared to conventional systems, x-vectors better utilize large-scale training datasets and data augmentation techniques, and are more robust in noisy, unconstrained recording conditions;
- Demonstration that the proposed representation achieves state-of-the-art performance on standard speaker recognition evaluation datasets and outperforms conventional systems based on i-vectors;
- Demonstration that x-vectors are applicable to related areas of speech processing, and achieve state-of-the-art performance in spoken language identification and speaker diarization.

In addition, most of the algorithms and source code relevant to the studies performed in this thesis have been made freely available (under an Apache 2.0 open source license) in the Kaldi toolkit.

1.2 Dissertation Organization

This chapter provides a high-level overview of the dissertation and its impact. Chapter 2 provides background on the speaker recognition task and related work. The remaining chapters of the dissertation are organized primarily based on the order in which the studies were performed. Chapter 3 describes an end-to-end speaker recognition system, which was our first attempt at replacing i-vectors for speaker recognition with neural network based models. In Chapter 4, the end-to-end approach is split into two parts, a DNN to compute embeddings (called x-vectors), and a separately trained backend to compare pairs of embeddings. In Chapter 5 we show that the x-vector architecture can be significantly improved with data augmentation. Appendix A describes the augmentation corpus used, which was created as part of this dissertation. The remaining chapters are devoted to other applications of x-vectors in speech processing. Chapters 6 and 7 demonstrate how x-vectors can be adapted to spoken language recognition and speaker diarization, respectively. Chapter 8 combines the work of Chapters 5 and 7 and shows how x-vector-based diarization and speaker recognition can be combined to handle multi-speaker recordings. Finally, Chapter 9 concludes the dissertation with a summary and discussion of future work.

Chapter 2

Speaker Recognition

2.1 Overview

Speaker recognition is the task of authenticating the claimed identity of a speaker, based on some speech signal and previously enrolled speaker recording. A block-diagram of a speaker recognition system is illustrated in Figure 2.1. If the identity of the speaker is known in advance, we can *enroll* the speaker into a database of speaker profiles. Speech from an unknown speaker is referred to as *test* speech, and is often accompanied by an identity claim. Typically, low-dimensional representations rich in speaker information are extracted for both the enrollment speech (corresponding to the claimed identity) and the test speech. Until recently, these representations were usually

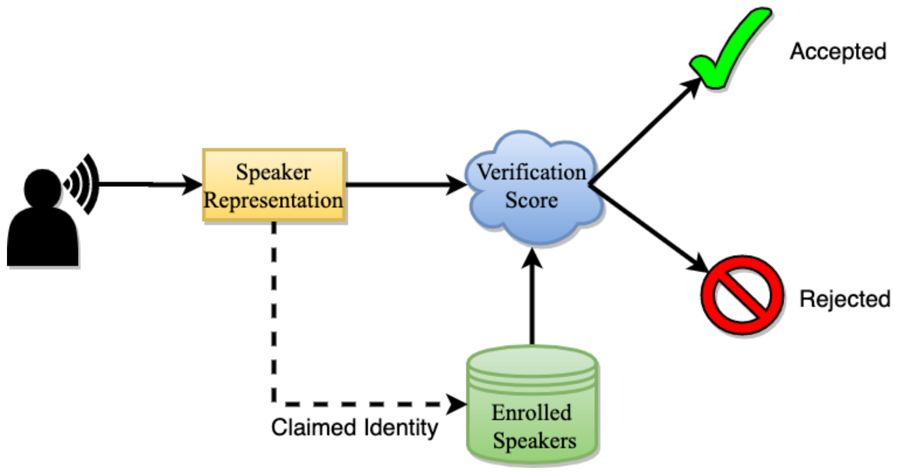


Figure 2.1: Diagram of a speaker recognition system.

i-vectors. The enroll and test representations are then compared to produce a score. If the score is sufficiently high, we "accept" the claim that the enrolled speaker is present in the test utterance. Conversely, if the score is low, we "reject" this claim.

If the lexical content of the utterances is fixed to some phrase, the task is considered text-dependent, otherwise it is text-independent. This dissertation is primarily concerned with text-independent speaker recognition. In Chapters 6 and 7 we also consider the related problems of spoken language recognition and speaker diarization.

In this chapter, we describe several facets of text-independent speaker recognition research which are relevant to this dissertation. Section 2.2 describes the corpora which are commonly used in speaker recognition. In Section 2.4 we review current speaker recognition technology. Section 2.4.1

gives an overview of i-vectors, the most popular embedding, and Section 2.4.1.2 describes probabilistic linear discriminant analysis, used in speaker comparison (also called scoring) with i-vectors. Finally, in Section 2.4.2 we review studies using direct neural network approaches, which the work in this dissertation uses as a stepping stone.

2.2 Corpora

In this section, we will discuss datasets that are commonly used in speaker recognition literature.

The National Institute for Standards in Technology (NIST) has played an important role in speaker recognition research. Since 1996, NIST has organized a speaker recognition evaluation (SRE) every one or two years. Long after the evaluation officially ends, the datasets that NIST releases as part of these evaluations continue to be used to benchmark speaker recognition performance. Older SRE datasets are often incorporated into system training data, which is then used to train models for later evaluations. Other commonly used training resources in academic literature include corpora available through the Linguistic Data Consortium (LDC), such as Switchboard and Fisher English.

To date, the majority of the NIST SREs consist primarily of telephone speech, with a smaller amount of microphone-recorded interviews. Although comprised mostly of English speech, some SRE datasets have included a large amount of non-English speech; notably, the *NIST Speaker Recognition Evaluation 2016* (2016) consisted entirely of Cantonese and Tagalog speech.

To address gaps in the NIST SREs, SRI International created an evaluation dataset called Speakers in the Wild (McLaren et al., 2016). This dataset consists of unconstrained wideband audio, extracted from videos. Compared to conventional speaker recognition corpora, video data is more diverse. Due to a wide variety of devices, audio codecs and recording environments, speaker recognition in the multimedia domain can be very difficult.

Recently, a large amount of training resources have been released in this video-audio domain, through the VoxCeleb corpus (Nagrani, Chung, and Zisserman, 2017).

2.3 Evaluating Performance

In this section, we discuss several commonly used metrics for evaluating the performance of speaker recognition systems. The evaluation metrics for the related task of speaker diarization are briefly discussed in Section 7.2.3. As described in greater detail in Section 2.4.1.2, we typically use probabilistic linear discriminant (PLDA) to compare pairs of embeddings. PLDA produces a comparison score, which is the log of the ratio of the probability that the embeddings were produced by the same speaker, versus the probability that they were produced by different speakers. If the comparison score is above a decision threshold θ, we posit that the embeddings belong to the same speaker, otherwise we declare that they belong to different speakers.

enroll	**test**	**trial type**
Alice	reco_1	target
Alice	reco_2	nontarget
Bob	reco_1	nontarget
Charles	reco_3	target
⋮	⋮	⋮

Table 2.1: An example of a trials file

Evaluating performance is enabled by a *trials* file, illustrated in Table 2.1. This file defines which enrolled speakers are compared with which test utterances, and what type of trial it is. There are two trial types, *target* which denotes trials where the enrollment and test speakers are the same, and *non-target*, where the speakers are different. Since we treat speaker recognition as a binary detection problem, there are two types of errors:

1. **Misses**: false negatives that occur when the score for a target trial is less than θ

2. **False alarms:** false positives that occur when the score for a non-target

trial is greater than or equal to θ

Given a threshold θ, we can compute the empirical probability of false alarms, $P_{\text{FA}}(\theta)$ and the probability of misses, $P_{\text{Miss}}(\theta)$. These two quantities will serve as the basis for all performance metrics we use in the speaker recognition portion of the dissertation. Equal error rate (EER) is a very commonly used error metric that is obtained by sweeping thresholds until a value θ_{EER} is found such that $P_{\text{FA}}(\theta_{\text{EER}})$ and $P_{\text{Miss}}(\theta_{\text{EER}})$ are equal. Therefore, when reporting this error rate, it is sufficient to report only $P_{\text{FA}}(\theta_{\text{EER}})$, once an appropriate threshold is determined. Another commonly used threshold is the detection cost function (DCF), which is a weighted sum of $P_{\text{FA}}(\theta)$ and $P_{\text{Miss}}(\theta)$. This is computed as $C_{\text{det}}(\theta) = C_{\text{miss}} P_{\text{miss}}(\theta) P_{\text{tar}} + C_{\text{FA}} P_{\text{FA}}(\theta)(1 - P_{\text{tar}})$. There are three application-dependent parameters: C_{miss} which defines the cost of misses, C_{FA} which provides the cost of false alarms, and P_{tar} which defines the prior probability of a target trial. Usually $C_{\text{FA}} = C_{\text{miss}} = 1$. A popular variant is the minimum of the detection cost function $C_{\text{det}}^{\text{min}} = \min_{\theta} C_{\text{det}}(\theta)$.

2.4 Related Work

This chapter describes a subset of foundational work, and is focused primarily on i-vector-based systems. This choice is for practical reasons. Until recently, i-vectors were the most popular method for text-independent speaker recognition (in addition to several related areas), and so they will serve as a baseline for all experiments in this dissertation. In addition, there exists a large body of work focusing on i-vector "backend" technology: methods for transforming i-vectors, making comparisons, calibrating comparison scores, and adapting systems. In Chapters 4 and 5 we will reuse this backend for x-vectors.

2.4.1 I-vectors

The i-vector paradigm uses factor analysis to compresses multiple sources of speech variability (including speaker characteristics) into a fixed-length, low-dimensional representation called an i-vector (Dehak et al., 2011). In most state-of-the-art systems, a probabilistic linear discriminant analysis backend compensates for undesirable channel characteristics captured in the i-vectors, and provides a mechanism to classify pairs of i-vectors as belonging to the same speaker or different speakers.

2.4.1.1 The Frontend: I-vector Extraction

Standard i-vector systems use a universal background model (UBM) in conjunction with standard acoustic features to collect sufficient statistics for i-vector extraction. Traditionally, the UBM is a Gaussian mixture model (GMM) trained on a large amount of speech to produce a model of the "average" speaker. Usually, it is trained using the expectation maximization (EM) algorithm on a large speech collection comprised of thousands of speakers. A GMM is a mixture of C multivariate Gaussians. Each Gaussian corresponds to an arbitrary acoustic or phonetic class. The acoustic features $\{\boldsymbol{x}_t\}$ are sequences of F-dimension vectors (e.g., 20-dimensional mel frequency cepstral coefficients or MFCCs) each extracted from a small (e.g., 20ms) sliding window over the audio. The GMM is parameterized by a set of scalar weights $\{w_0, w_1, \cdots w_C\}$ that sum to 1, F-dimensional means $\{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \cdots \boldsymbol{\mu}_C\}$ and $F \times F$ dimensional covariances $\{\boldsymbol{\Sigma}_0, \boldsymbol{\Sigma}_1, \cdots \boldsymbol{\Sigma}_C\}$. This is written compactly as $\lambda = \{\{w_c\}, \{\boldsymbol{\mu}_c\}, \{\boldsymbol{\Sigma}_c\}\}$. The GMM computes the likelihood that $\boldsymbol{x}_t$ is generated from the GMM parameterized by λ.

$$p(\boldsymbol{x}_t \mid \lambda) = \sum_{c=1}^{C} w_c N(\mathbf{x}_t; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{2.1}$$

In the foundational work by Reynolds, Quatieri, and Dunn (2000), speaker-level GMMs are created by adapting the GMM-UBM, which was trained on a large number of speakers, to a specific speaker using maximum a posteriori (MAP) adaptation. Given speech from an unknown speaker, we can determine how likely the speech is generated from an adapted speaker model versus how likely it was generated by the UBM, enabling a likelihood ratio test. It has been observed empirically that it is only necessary to adapt the UBM means (rather than weights or covariances). Therefore, the speaker characteristics are captured entirely by a shift with respect to the C UBM means. The i-vector framework builds on this, and posits that the total variability of a recording u is captured in a low-dimension subspace of that mean shift, $\boldsymbol{m}_c(u) = \boldsymbol{\mu}_c + \mathbf{T}_c\boldsymbol{w}(u)$, where $\boldsymbol{m}_c(u)$ is the adapted mean. Given a sequence of T speech features $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T\}$ extracted from a recording u, $\boldsymbol{w}(u) \sim N(\mathbf{0}, I)$ is a low-dimensional vector that captures the variability in the recording. This results in a modification to the form of the Gaussian means in Equation 2.1:

$$p(\boldsymbol{x}_t \mid \lambda) = \sum_{c=1}^{C} w_c N(\boldsymbol{x}_t; \boldsymbol{\mu}_c + \boldsymbol{T}_c\boldsymbol{w}(u), \boldsymbol{\Sigma}_c) \tag{2.2}$$

The relationship between the utterance-level means and the UBM means can be represented compactly as follows:

$$\boldsymbol{m}(u) - \boldsymbol{\mu} = \boldsymbol{T}\boldsymbol{w}(u) \tag{2.3}$$

Where $\boldsymbol{m}(u)$ and $\boldsymbol{\mu}$ are FC-dimensional “supervectors,” the concatenation of the C Gaussian means. For example, $\boldsymbol{\mu}^\top = \left[\boldsymbol{\mu}_1^\top \boldsymbol{\mu}_2^\top \cdots \boldsymbol{\mu}_C^\top\right]^\top$. The matrix $\boldsymbol{T}$ is a tall skinny projection matrix with dimensions $FC \times W$, that is trained using EM to maximize the likelihood of the training data. The i-vector $\boldsymbol{w}(u)$ is of dimension W (usually 100–600). Once the projection matrix $\boldsymbol{T}$ is trained, i-vectors are computed by accumulating 0th and centered 1st order statistics (Equations 2.4 and 2.6) from the UBM.

$$\boldsymbol{N}_c(u) = \sum_{t=1}^{T} p(c \mid \boldsymbol{x}_t) \tag{2.4}$$

$$\boldsymbol{F}_c(u) = \sum_{t=1}^{T} p(c \mid \boldsymbol{x}_t)(\boldsymbol{x}_t) \tag{2.5}$$

$$\tilde{\boldsymbol{F}}_c(u) = \sum_{t=1}^{T} p(c \mid \boldsymbol{x}_t)(\boldsymbol{x}_t - \boldsymbol{\mu}_c) \tag{2.6}$$

The i-vector is a MAP point estimate, obtained as follows in Equation 2.7.

$$\boldsymbol{w}(u) = (I + \boldsymbol{T}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{N}(u) \boldsymbol{T})^{-1} \boldsymbol{T}^\top \boldsymbol{\Sigma}^{-1} \tilde{\boldsymbol{F}}(u) \tag{2.7}$$

Where $\boldsymbol{N}(u)$ is a diagonal matrix of dimension $FC \times FC$, with diagonal blocks equal to $\boldsymbol{N}_c(u) I_{F \times F}$ and $\tilde{\boldsymbol{F}}(u)$ is an $F \times C$-dimensional matrix whose rows are equal to $\tilde{\boldsymbol{F}}_c(u)$.

2.4.1.2 The Backend: Channel Compensation and Scoring

Most modern speaker recognition systems use probabilistic linear discriminant analysis (PLDA) to compare pairs of i-vectors. PLDA was developed independently by Ioffe (2006) and Prince and Elder (2007), though the latter's work is most often cited in the speaker recognition community.

The standard PLDA model assumes that an i-vector $\boldsymbol{w}(u)$ extracted for a recording u that was generated by a speaker s can be decomposed into the sum of (i) an offset $\boldsymbol{m}$, (ii) a (typically) low-dimensional speaker specific component $\boldsymbol{S}\boldsymbol{z}_s$, and (iii) a within-speaker variability term $\boldsymbol{\epsilon}_{u,s}$ to model session variability.

$$\boldsymbol{w}(u) = \boldsymbol{m} + \boldsymbol{S}\boldsymbol{z}_s + \boldsymbol{\epsilon}_{u,s} \tag{2.8}$$

The term $\boldsymbol{m}$ is a speaker-independent mean, $\boldsymbol{z}_s \sim N(\boldsymbol{0}, I)$ is the latent factor for speaker s, and $\boldsymbol{\epsilon}_{u,s} \sim N(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a full-covariance matrix.

Given two i-vectors $\boldsymbol{w}(u_1)$ and $\boldsymbol{w}(u_2)$ extracted from utterances u_1 and u_2, and a trained PLDA model, we can compute the likelihood ratio between the hypothesis H_{same} that the i-vectors share the same speaker factor $\boldsymbol{z}$, versus the hypothesis H_{diff} that the i-vectors were produced by different speakers, with factors $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. The PLDA score is the log of this quantity, namely

$$\text{score} = ln \frac{p(\boldsymbol{w}(u_1), \boldsymbol{w}(u_2) \mid H_{\text{same}})}{p(\boldsymbol{w}(u_1) \mid H_{\text{diff}}) p(\boldsymbol{w}(u_2) \mid H_{\text{diff}})}, \tag{2.9}$$

and is a sufficient statistic under the Gaussian assumptions for same-or-different speaker decisions.

According to Garcia-Romero and Espy-Wilson (2011), i-vectors exhibit non-Gaussian behavior. The authors proposed a process of whitening and length-normalization, prior to providing them as input to a PLDA model. This encourages i-vectors to conform to the Gaussian assumptions of PLDA, and, empirically, it tends to improve performance. In this dissertation, we either apply this form of post-processing to the new embeddings, or a modification that replaces the whitening transform with one learned from linear discriminant analysis (LDA).

2.4.1.3 Discriminative Training in I-vector-based Systems

Ordinarily, the components of an i-vector system are not directly optimized for speaker recognition. Burget et al. (2011) proposed replacing the typically generative PLDA model with one trained to discriminate between same-speaker and different-speaker trials. Glembek et al. (2011) went deeper into the i-vector pipeline to discriminatively train the $\boldsymbol{T}$ matrix in the i-vector extractor. Our end-to-end objective function in Section 3.4 bears some similarity with the objective function used in this approach, although in our work, we use it to train a DNN rather than the parameters of the i-vector system.

2.4.1.4 I-vectors with DNNs

The DNNs most often found in speaker recognition are trained as acoustic models for automatic speech recognition (ASR), and then used to enhance phonetic modeling in the i-vector UBM: either posteriors from the ASR DNN replace those from a Gaussian mixture model (GMM) (Lei et al., 2014; Kenny et al., 2014), or bottleneck features (BNFs) are extracted from the DNN and combined with acoustic features (McLaren, Lei, and Ferrer, 2015). If DNN posteriors are used, then Equations 2.4–2.6 are modified so that $p(c \mid \boldsymbol{x}_t)$ is computed by the ASR DNN instead of from the GMM. Using BNFs is more straightforward: we simply concatenate BNFs (usually 30-80 dimensional) on to the acoustic features (usually MFCCs). In either case, if the ASR DNN is trained on in-domain data, the improvement over traditional acoustic i-vectors is substantial (Garcia-Romero et al., 2014; Snyder, Garcia-Romero, and Povey, 2015; Sadjadi, Pelecanos, and Ganapathy, 2016). However, this approach introduces the need for transcribed speech to train the DNNs, and greatly increases computational complexity compared to traditional i-vectors.

2.4.2 Direct DNN Methods

Alternatively, neural networks can be directly optimized to discriminate between speakers. This has potential to produce powerful, compact systems (Heigold et al., 2016), that only require speaker labels to train.

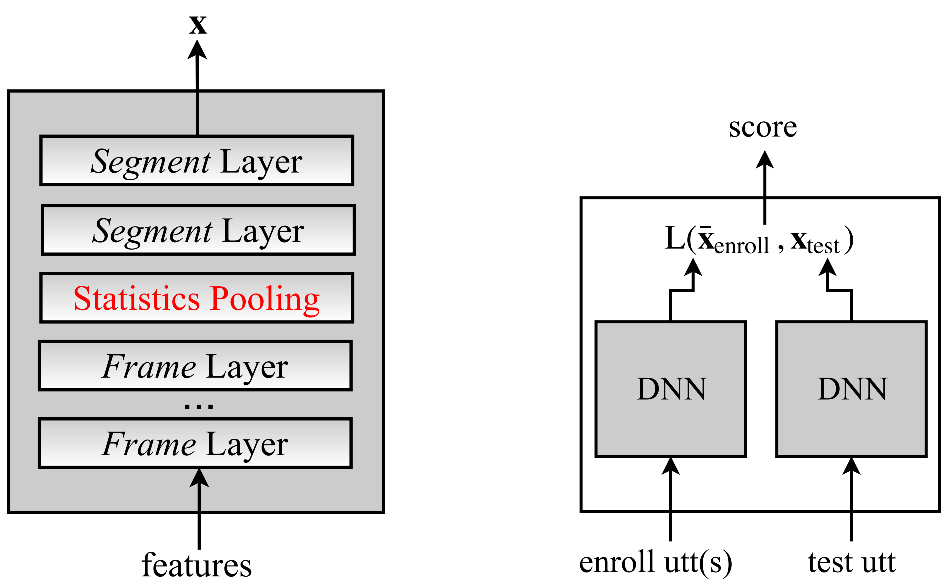
In early systems of this kind, neural networks were trained to separate speakers, and frame-level representations were extracted from the network and used as features for Gaussian speaker models (Konig et al., 1998; Heck et al., 2000; Salman, 2012; Yamada, Wang, and Kai, 2013) or i-vectors (Ghalehjegh and Rose, 2015). In work by Variani et al. (2014) completed at Google for the "OK Google" application, text-dependent speaker representations called *d*-vectors are created by averaging the final hidden layer activations of a DNN trained for speaker identification. Heigold et al. (2016) built on this work for the same application, and introduced an end-to-end system that jointly learns an embedding along with a similarity metric to compare pairs of embeddings. The work in this dissertation adapted this approach to a text-independent application by inserting a temporal pooling layer into the network to handle the wider variability of recording length found in this less constrained domain (Chapter 3). This work was continued in (Snyder et al., 2017), where we split the end-to-end approach into two parts: a DNN to produce embeddings and a separately trained classifier to compare them (Chapter 4). This facilitates the use of all the accumulated backend technology developed over the years for i-vectors, such as length-normalization, PLDA scoring, and domain adaptation techniques. This approach was greatly improved using data augmentation in (Snyder et al., 2018b) (Chapter 5) and applied to the related tasks of spoken language recognition (Mccree et al., 2018; Snyder et al., 2018a) and speaker diarization (Sell et al., 2018) (Chapter 6 through 8).

Chapter 3

End-to-end Speaker Recognition

3.1 Overview

In our first investigation of discriminative neural networks for speaker recognition, we developed an end-to-end system. In contrast to a traditional i-vector system, all components of this system are jointly optimized to distinguish between pairs of utterances belonging to the same speaker (on one hand) versus pairs of utterances from different speakers (see Figure 3.1b). This architecture was partially inspired by the end-to-end text-dependent speaker verification system developed by Heigold *et al.* for "OK Google" (Heigold et al., 2016). To adapt this architecture to our text-independent application, we introduced a statistics pooling layer (see Figure 3.1a) that aggregates information across



(a) DNN Architecture

(b) Scoring Schema

Figure 3.1: Diagram of the DNN and scoring method. The network consists of several frame-level layers, the statistics pooling layer and subsequent layers that operate on the entire input segment.

variable-length input sequences. An additional difference between our proposed work and that of Heigold et al. (2016) is our objective function, which is inspired by PLDA.

The proposed architecture, sometimes called a Siamese network (Bromley et al., 1994), is a feed-forward DNN that extracts statistics over a sequence of stacked MFCCs and maps it to a speaker embedding. The objective function operates on pairs of embeddings, and maximizes a same-speaker probability for pairs of embeddings from the same speaker, and minimizes the same probability for pairs of embeddings from different speakers. Our system is built using the nnet3 neural network library in the Kaldi speech recognition toolkit (Povey et al., 2011).

3.2 Features

The features are 20-dimensional MFCCs with a frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds. Nine frames are spliced together to create a 180-dimensional input vector to the DNN. After splicing, a frame-level GMM-based speech activity detection (SAD) system filters out nonspeech frames.

3.3 Neural Network Architecture

The network, illustrated in Figure 3.1a, consists of four hidden layers that sequentially process the 180-dimensional input sequence, followed by a temporal pooling layer. The pooling layer aggregates the output of the hidden layers over time and computes its average and standard deviation. These statistics are concatenated together, propagated to a final hidden layer, followed by a linear transformation that produces the speaker embedding $\mathbf{x}$. The network activations are a type of network-in-network (NIN) nonlinearity introduced in (Ghahremani et al., 2016). The NIN component maps an input of dimension d_i to output of dimension d_o. Internally, a set of n micro neural networks (Lin, Chen, and Yan, 2013) project the input to a d_h-dimensional space. Within a NIN component, micro neural networks share parameters, and are composed of a stack of three rectified linear units (ReLUs) connected by affine transformations. Refer to (Ghahremani et al., 2016) for implementation details. In this chapter, our network uses the NIN configuration $\{n = 150, d_i = 600, d_h = 2000, d_o = 3000\}$, which results in a model with 6.7 million parameters. In all later chapters, we replace the NIN with more commonly used neural network nonlinearities.

3.4 Training

We model the probability that embeddings $\mathbf{x}$ and $\mathbf{y}$ belong to the same speaker by the logistic function in Equation 3.1. $L(\mathbf{x}, \mathbf{y})$, defined in Equation 3.2, is a quadratic similarity metric, similar to discriminative PLDA (Burget et al., 2011). Let S_{diff} and S_{same} be the set of different-speaker and same-speaker pairs, respectively. The objective function (Equation 3.3) is the log probability of the correct choice for each pair. Since there are typically many more training examples in the set S_{diff} than in S_{same}, we introduce a constant α so that each set has the same weight in the objective function.

$$Pr(\mathbf{x}, \mathbf{y}) = \frac{1}{1 + e^{-L(\mathbf{x}, \mathbf{y})}} \tag{3.1}$$

$$L(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y} - \mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{y}^T\mathbf{A}\mathbf{y} + b \tag{3.2}$$

$$E = -\sum_{\mathbf{x}, \mathbf{y} \in S_{\text{same}}} ln\left(Pr(\mathbf{x}, \mathbf{y})\right) - \alpha \sum_{\mathbf{x}, \mathbf{y} \in S_{\text{diff}}} ln\left(1 - Pr(\mathbf{x}, \mathbf{y})\right) \tag{3.3}$$

Training examples are organized as pairs of same-speaker feature chunks. Minibatches are formed by picking N pairs, such that no two pairs are from the same speaker. Combining chunks across pairs results in an additional $N(N+1)/2$ different-speaker pairs. To handle channel variability, all chunks in the minibatch come from different recordings. Speaker embeddings are extracted from all $2N$ chunks and passed to the objective function. The derivatives with respect to $\mathbf{A}$, b and embeddings $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_{2N}$ are computed, and backpropagated through the DNN for parallelized stochastic gradient descent (Povey, Zhang, and Khudanpur, 2015). GPU memory limitations in conjunction with very long features (e.g., up to 3000 frames in this work) restricted the size of the minibatches to $N = 10$ same-speaker pairs. Training pairs are shuffled after each iteration to ensure that many different speakers and speech cuts are compared.

In our application, it is important to minimize sensitivity to speech duration. A straightforward strategy is to train the DNN on chunks that vary widely in duration. However, we found that beginning training with the full range of chunk lengths results in unstable convergence. Our solution is to use a two-stage curriculum learning approach, where long duration chunks are presented first, followed by a mixture of short and long chunks. In the first stage, we train for two epochs using 10–30 second chunks, followed by another two epochs with 1–30 second chunks in the second stage.

3.5 Speaker Embeddings

Although an embedding can be extracted from a recording of any length, we found it convenient from a memory standpoint to extract embeddings from 30 second chunks and average to get an utterance-level representation. A single embedding is generated from the entire utterance if it is shorter than 30 seconds. Enrollment embeddings are extracted from one or more utterances, and averaged to create a speaker-level representation. As illustrated in Figure 3.1b, enroll and test utterances $\mathbf{x}$ and $\mathbf{y}$ are scored by the similarity metric used in the objective function (Equation 3.2).

3.6 Baseline I-vector System

The baseline is a standard i-vector system that is based on the GMM-UBM Kaldi recipe described in (Snyder, Garcia-Romero, and Povey, 2015). The front-end features consist of 20 MFCCs with a frame-length of 25ms that are mean-normalized over a sliding window of up to 3 seconds. First and second order temporal differences of the MFCC vector, sometimes called delta and acceleration, are appended to create 60-dimension feature vectors. A frame-level GMM-based speech activity detection (SAD) system selects feature vectors corresponding to speech frames. The UBM is a 4096-component full-covariance GMM. The system uses a 600-dimension i-vector extractor. Prior to PLDA scoring, i-vectors are centered and length normalized. To compensate for duration mismatch, we use the strategy of truncating PLDA training data (Hasan et al., 2013). The training dataset is copied and randomly cropped to the first 1–20 seconds. The PLDA model is trained on either the short version alone, or on the combination of the short and maximum length versions.

3.7 Spoken Communications Dataset

The datasets consist of US English telephone speech. Calls are sampled at 8kHz and compressed using an internal process. Table 3.1 lists statistics for

Table 3.1: Statistics of the Spoken Communications training and evaluation datasets. The column titled #spkr is the number of speakers in the set. The next column is the total number of recordings, followed by the average number of recordings per speaker, and finally the average duration of the recordings.

	#spkr	**tot #rec**	**avg #rec/spkr**	**avg. dur**
train5k	5k	25k	4.93	81s
train15k	15k	53k	3.53	84s
train102k	102k	226k	2.22	91s
enroll	2419	2915	1.21	91s
test	2419	2419	1	1–92s

the datasets. The full training dataset, train102k, comprises roughly 102,000 speakers and more than 5,700 hours of speech. To explore the effect of training dataset size on performance, we found it useful to create reduced datasets train5k and train15k, with 5,000 and 15,000 speakers respectively (see Section 3.8.2).

The evaluation dataset consists of a set of 2,419 speakers that does not overlap with the set of training speakers. Trials were constructed by randomly pairing up recordings from the evaluation speakers such that roughly 80 percent are nontarget. In total, there are 12,362 trials. Gender labels are not used, so our evaluation contains same and cross gender trials. The test conditions consist of full-length enrollment recordings, as well as test segments of various shorter durations. Test segments are created by truncating the recordings to the first T seconds of speech, as detected by our GMM frame-level SAD. The SAD uses a threshold that corresponds to the equal error rate of a speech detection task on an in-domain development set. This helps to ensure that, on average, the test condition labels (e.g., “1s”, “2s”, etc) faithfully report the actual speech duration in the test segments. The same list of trials is used for all the duration conditions.

3.8 Results

In this study, we had access to a very large proprietary dataset provided by our industry collaborator Spoken Communications. Non-overlapping portions of this dataset were used for training and evaluation. Due to the interests of our industry collaborator, we focused on real-time recognition. To reduce latency, it is important to minimize the amount of speech required to achieve an accurate verification. Our evaluation follows from this, and involves full-length enrollment recordings and test speech ranging from one second to full-length. The average duration of a full length recording, as noted in Table 3.1, is about 90 seconds.

Table 3.2: EER(%) for duration robust systems on the Spoken Communications evaluation datasets. The results are broken down by the test utterance duration, denoted by the column labels "1s," "2s," "3s," etc. The column titled "pool" combines all durations in a single evaluation condition. All enrollment recordings are full-length. The rows labeled "1–20s", "1–30s", etc, denote the amounts of training speech used for adaptation.

ivec102k	**1s**	**2s**	**3s**	**5s**	**10s**	**20s**	**full**	**pool**
1–20s	14.1	8.7	6.7	4.9	3.7	3.2	2.8	8.5
1–20s+full	15.0	9.4	7.0	5.1	3.8	3.1	2.6	10.0
full	16.4	9.9	7.3	5.2	3.8	2.8	2.4	10.6
dnn102k	**1s**	**2s**	**3s**	**5s**	**10s**	**20s**	**full**	**pool**
1–20s	12.6	7.5	6.0	4.2	3.4	2.6	2.5	6.0
1–30s	13.8	8.7	6.2	4.6	3.4	2.6	2.4	6.6

3.8.1 Duration Robustness

Our application requires high accuracy on short test segments and calibrated scores across test conditions. We therefore examine several methods to compensate for duration variability. The labels on the the first seven columns of Table 3.2 denote the amount of speech retained in the test segments. The final column is for pooled results. The row labels describe how the training data is configured. For i-vectors, this involves training a PLDA model on randomly truncated speech as described in Section 3.6. The end-to-end system uses the two stage method described in Section 3.4 to handle duration variability.

We denote i-vector and DNN systems trained on train102k as ivec102k and dnn102k respectively. For ivec102k, training the PLDA model on 1–20 second segments results in the best performance on the shortest five test conditions

Table 3.3: EER(%) on Spoken Communications evaluation datasets (broken down by test utterance duration) as a function of training dataset size. The rows "5k," "10k," and "102k" denote systems trained on 5,000, 10,000 and about 102,000 speakers respectively. The best results for each training set and test condition pair are *italicized*.

	# training speakers	**1s**	**2s**	**3s**	**5s**	**10s**	**20s**	**full**	**pool**
i-vector	**5k**	*14.8*	*11.4*	*9.0*	*7.0*	*5.4*	*4.4*	*3.5*	*8.6*
dnn	5k	17.5	12.6	10.7	8.7	7.2	6.4	6.2	10.6
i-vector	15k	*13.8*	*9.0*	*7.0*	*5.1*	*3.9*	*3.0*	*2.7*	*8.0*
dnn	15k	14.2	10.7	8.0	6.5	5.4	4.9	4.9	8.3
i-vector	102k	14.1	8.7	6.7	4.9	3.7	3.2	2.8	8.5
dnn	102k	*12.6*	*7.5*	*6.0*	*4.2*	*3.4*	*2.6*	*2.5*	*6.0*

and pooled. Adapting dnn102k to 1–20 instead of 1–30 second chunks produces equivalent or better results on all but the full-length condition. With respect to systems using 1–20 second chunks, dnn102k outperforms ivec102k on all conditions, and achieves a relative improvement of 13% in terms of average EER and 29% in pooled EER. Since we are more concerned with the short-duration conditions in this setting, we use the 1–20 second adaptation methods for the remaining experiments.

3.8.2 Training Data Size

In this section, we study the relationship between system performance and training dataset size. Trained on the smaller datasets, we find that the i-vector system outperforms the DNN on all conditions. This is especially noticeable

Table 3.4: EER(%) of i-vector and end-to-end systems on the Spoken Communications evaluation dataset (broken down by duration). Both systems are trained on 102,000 training speakers and the individual systems are compared with their fusion.

	1s	**2s**	**3s**	**5s**	**10s**	**20s**	**full**	**pool**
ivec102k	14.1	8.7	6.7	4.9	3.7	3.2	2.8	8.5
dnn102k	12.6	7.5	6.0	4.2	3.4	2.6	2.5	6.0
fusion	10.2	6.1	4.3	3.4	2.4	1.9	1.6	5.3

on the long duration conditions: trained on 5,000 speakers, the i-vector system is 44% better than the end-to-end system on the full condition, although it is only 15% better on the 1s condition. However, the average performance seems to reach diminishing returns for the i-vector system at 15,000 speakers, and additional speakers do not consistently improve the results. On the other hand, the end-to-end DNN appears better able to exploit a substantial increase in the number of speakers. For the end-to-end system, the average EER improves by 21% after increasing the number of speakers from 5,000 to 15,000, and 29% from 15,000 to 102,000.

It seems reasonable to conclude from this that the neural method both requires and is able to leverage a large amount of labeled training data.

3.8.3 System Combination

The DNN performs well by itself, but due to the significant architectural differences between it and the i-vector baseline, we anticipate that the systems

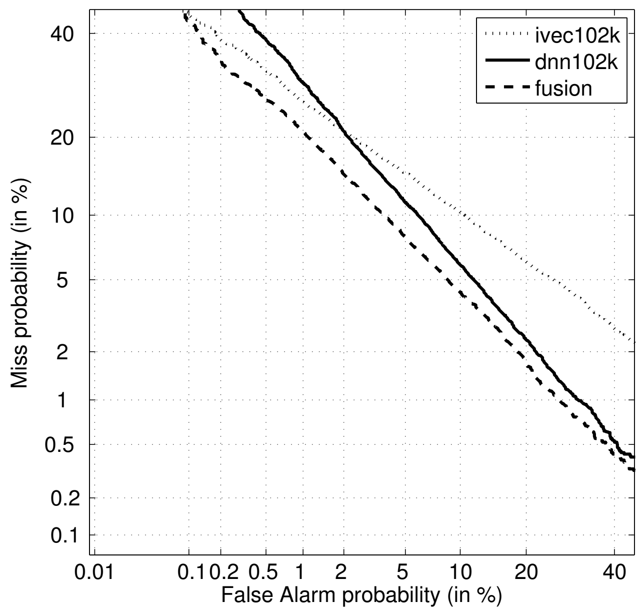


Figure 3.2: DET curve for the pooled 1s, 2s, 3s, and 5s conditions.

are excellent candidates for fusion. To fuse ivec102k and dnn102k, we first normalize the scores using mean and variance calculated from all pooled scores and add them together. Relative to the baseline, the fused system is 32% and 38% better, in terms of average and pooled EER.

3.8.4 DET Curves

So far, we have compared systems at the EER operating point, in which the false alarm rate (FAR) equals the miss rate (MR). However, for many verification applications, avoiding false alarms is prioritized. Therefore, we plot detection error tradeoff (DET) curves for the individual and fusion systems. Relative to the i-vector, the DNN performs better at a low MR and worse at a

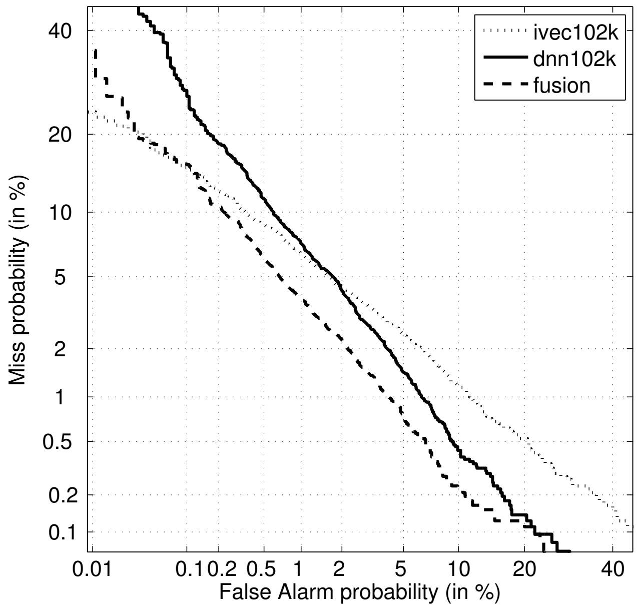


Figure 3.3: DET curve for the pooled 10s, 20s, and full conditions.

low FAR. Figure 3.2 plots a DET curve for the 1–5 second test conditions. We see that ivec102k and dnn102k overlap at 2% FAR and 20% MR. The baseline ivec102k is better for FAR less than 2%, although the DNN is better everywhere else. With the exception of extremely low FAR, the fusion system is the same or better than the individual systems. Figure 3.3 shows a similar pattern for the long duration test conditions, but the cross over occurs at 2% FAR and 4.5% MR. This indicates that the DNN dominates over a larger set of operating points for short duration test conditions than for long.

Table 3.5: Comparison of the EER of the end-to-end system and the i-vector system on the NIST SRE 2010. The systems are trained on speech from about 13,000 speakers.

system	**EER(%)**
i-vector	2.49
dnn	7.26

3.8.5 NIST SRE 2010

In the previous sections, we studied the performance of the end-to-end system on a proprietary dataset. However, we saw in Section 3.8.2 that while the system outperforms i-vector systems with more than 100,000 speakers, it lagged behind the i-vector system with 15,000 speakers or less. In this section, we see if this performance trend persists using the publicly available datasets frequently used in the speaker recognition community. In Table 3.5, the systems are trained on about 13,000 speakers from NIST SREs prior to 2010, Switchboard 2 Phases 1, 2, and 3, and Switchboard Cellular, and Fisher English. The end-to-end and i-vector architectures use architectures that are similar to those used in the rest of this chapter. We evaluate performance on the commonly used NIST SRE 2010 condition 5 (see *The 2010 NIST Speaker Recognition Evaluation* (2010)), and report results using EER. Unfortunately, we find that the end-to-end system's error rate is almost three times higher than the traditional i-vector system. Clearly, in order for this method to be adopted widely, performance must be improved on datasets of modest size. In the next chapter, we find that an alternative system can achieve results which are more competitive with i-vectors on datasets of this size. Later, in Chapter 5, we show how data augmentation can improve the results even further, until it is substantially better than i-vectors.

3.9 Conclusions

We studied a neural network architecture that extracts speaker embeddings from variable-length speech segments, and scores them using the similarity metric from the objective function. Heigold et al. (2016) suggested that neural network-based end-to-end architectures are generally applicable to verification tasks. Our findings agree with this, and show promising results for text-independent speaker verification, given an adequate number of training speakers. We found that the proposed architecture outperformed our i-vector baseline by 13% average and 29% pooled EER. A larger relative improvement for the pooled error rate and on the short test conditions suggests that the DNN-based embeddings may be more robust to duration variability, and better at modeling speaker characteristics from small amounts of speech. Although these results are quite encouraging, we observed that performance is strongly tied to the amount of training speakers. For training datasets with less than 15,000 speakers, performance lagged behind a traditional i-vector system. This presents a barrier to widespread adoption of this approach. In the next two chapters, we work to mitigate this limitation, and eventually achieve state-of-the-art performance even when only modest amounts of labeled training data are available.

Chapter 4

X-vectors: from End-to-end to Embeddings

4.1 Overview

In Chapter 3, we saw that an end-to-end approach outperformed a traditional i-vector system when trained on a large proprietary dataset with over 100,000 speakers. Unfortunately, the end-to-end approach was not as competitive when trained on a smaller number of speakers. This chapter describes modifications to the end-to-end system to achieve competitive performance when trained and tested on much smaller, publicly available corpora. This

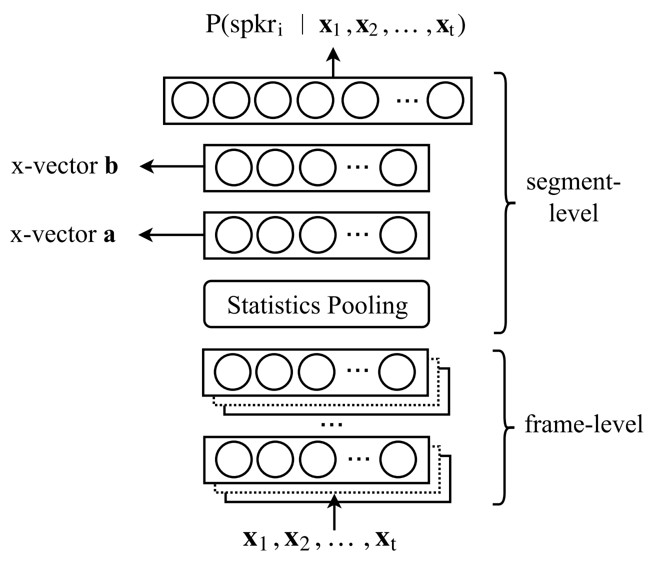

P(spkr$_i$ | $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$)
x-vector **b**
x-vector **a**
Statistics Pooling
segment-level
frame-level
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$

Figure 4.1: Diagram of the DNN. Segment-level embeddings (e.g., x-vector **a** or **b**) can be extracted from any layer of the network after the statistics pooling layer.

is achieved by splitting the end-to-end system into two parts: a DNN to extract embeddings, which we call *x-vectors*, and a separately trained backend to compare pairs of embeddings. This facilitates reuse of all the accumulated backend technology developed over the years for i-vectors, such as length-normalization, PLDA scoring and domain adaptation techniques. This flexibility is in contrast to the similarity metric used in the end-to-end system in Equation 3.2, which does not directly support use of existing domain adaptation techniques.

4.2 Input Features

The features are 20-dimensional MFCCs with a frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds. An energy-based speech activity detection (SAD) system removes nonspeech frames. Instead of stacking frames at the input as we did in Chapter 3, short-term temporal context is handled by a time-delay DNN architecture (Peddinti, Povey, and Khudanpur, 2015).

4.3 Neural Network Architecture

The network, illustrated in Figure 4.1, consists of layers that operate on speech frames, a statistics pooling layer that aggregates over the frame-level representations to obtain a segment-level representation, additional layers that operate at the segment-level, and finally a softmax output layer. The nonlinearities are rectified linear units (ReLUs).

The first 5 layers of the network work at the frame level, with a time-delay architecture (Peddinti, Povey, and Khudanpur, 2015). Suppose t is the current time step. At the input, we splice together frames at $\{t-2, t-1, t, t+1, t+2\}$. The next two layers splice together the output of the previous layer at times $\{t-2, t, t+2\}$ and $\{t-3, t, t+3\}$, respectively. The next two layers also operate at the frame-level, but without any added temporal context. In total, the frame-level portion of the network has a temporal context of $t-8$ to $t+8$ frames. Layers vary in size, from 512 to 1536, depending on the splicing context used.

We employ the same statistics pooling layer introduced in Chapter 3. It receives the output of the final frame-level layer as input, aggregates over all of the frames of input segment, and computes a segment-level mean and standard deviation. These segment-level statistics are concatenated together and passed to two additional hidden layers with dimension 512 and 300 (either of which may be used to compute embeddings) and finally the softmax output layer. Excluding the softmax output layer (because it is not needed after training) there are a total of 4.4 million parameters in the network.

4.4 Training

The network is trained to classify individual training speakers using a multi-class cross entropy objective function (Equation 4.1). The primary difference between this and the neural network training in (Konig et al., 1998; Heck et al., 2000; Variani et al., 2014) is that our system is trained to predict speakers from variable-length segments, rather than from individual frames. Suppose there are K speakers in the training data. Then $P(spkr_k \mid \mathbf{x}_{1:T}^{(n)})$ is the probability of the speaker being k given T input frames $\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ...\mathbf{x}_T^{(n)}$ of the n^{th} training segment.

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} ln(P(spkr_k \mid \mathbf{x}_{1:T}^{(n)})) \tag{4.1}$$

The quantity d_{nk} is 1 if the speaker label for the features corresponding to the n^{th} segment is k, otherwise it's 0.

As opposed to the formulation in Chapter 3, this DNN is trained on modest-sized publicly available corpora. The training data consists of primarily English conversational telephone speech, drawn from Switchboard corpora and NIST SREs (prior to 2010). The dataset is refined by removing any recordings that are less than 10 seconds long, and any speakers with fewer than 4 recordings. This leaves a total of 4,733 speakers, which determines the size of the softmax output layer.

To reduce sensitivity to utterance length, it is desirable to train the DNN on speech chunks that capture the range of durations we expect to encounter at test time (e.g., a few seconds to a few minutes). However, GPU memory limitations force a tradeoff between minibatch size and maximum training example length. As a comprise, we pick examples that range from 2 to 10 seconds (200 to 1000 frames) along with a minibatch size of 32 to 64. The example speech chunks are sampled densely from the recordings, resulting in about 3,400 examples per speaker. The network is trained for several epochs using natural gradient stochastic gradient descent (Povey, Zhang, and Khudanpur, 2015).

4.5 Speaker Embeddings

Ultimately, the goal of training the network is to produce embeddings that generalize well to speakers that have **not** been seen in the training data. We would like embeddings to capture speaker characteristics over the entire utterance, rather than at the frame-level. Thus, any layer after the statistics pooling layer is a plausible place to extract the embedding from. We do not consider the presoftmax affine layer because of its large size and dependence on the number of speakers. In the network used in this work, we are left with two affine layers from which to extract embeddings. These are depicted in Figure 4.1 as x-vector **a** and x-vector **b**. X-vector **a** is the output of an affine layer directly on top of the segment-level statistics. X-vector **b** is extracted from the next affine layer after a ReLU, and so it is a nonlinear function of the statistics. Since they are part of the same DNN, if x-vector **b** is computed then we get x-vector **a** for free. In later chapters, we will exclusively use embeddings extracted from the first affine layer after the statistics (referred to as "x-vector **a**" in this chapter), and will simply denote this embedding as an "x-vector" without further qualifications.

4.6 PLDA Backend

We use the same backend for i-vectors and x-vectors. Representations are centered and their dimensionality is reduced using LDA. We found that an LDA dimension of 25% of the original worked well for both types of embeddings. After dimensionality reduction, the embeddings are length normalized and pairs of embeddings are compared using PLDA. PLDA scores are normalized using adaptive s-norm (Sturim and Reynolds, 2005; Cumani et al., 2011). As described in Section 4.5, the DNN architecture presents the option of using x-vectors **a** or **b** or their concatenation. Instead of concatenating the embeddings together, we found that it produces better results to compute separate PLDA backends for each x-vector, and average the scores. Results using the combined x-vectors are denoted as *x-vectors* in Section 4.9.

4.7 Baseline I-vector System

The baseline is a traditional i-vector system (see Section 2.4.1) and is based on the GMM-UBM Kaldi recipe described in (Snyder, Garcia-Romero, and Povey, 2015). The front-end features consist of 20 MFCCs with a frame-length of 25ms that are mean-normalized over a sliding window of up to 3 seconds. Delta and acceleration features are appended to create 60 dimension feature vectors. An energy-based SAD selects features corresponding to speech frames. The UBM is a 2048-component full-covariance GMM. The system uses a 600-dimension i-vector extractor. Prior to PLDA scoring, i-vectors are centered, dimensionality reduced to 150 using LDA, and length normalized. PLDA scores are normalized using adaptive s-norm (Cumani et al., 2011).

4.8 Evaluation

We assess performance on the NIST 2010 and 2016 speaker recognition evaluation datasets (*The 2010 NIST Speaker Recognition Evaluation* (2010) and *NIST Speaker Recognition Evaluation 2016* (2016)). In the remaining sections, these will be abbreviated as *SRE10* and *SRE16* respectively. SRE10 consists of English telephone speech. Our evaluation is based on the extended core condition 5 and the 10s-10s condition, the latter of which designates trials consisting of a 10 second enrollment segment and a 10 second test segment. Refer to *The 2010 NIST Speaker Recognition Evaluation* (2010) for additional information on these conditions. To supplement the core SRE10 condition, we produce additional conditions in which the enrollment utterances are full-length, but the test utterances have been truncated to the first $T \in \{5, 10, 20, 60\}$ seconds of speech, as determined by an energy-based SAD. SRE16 is comprised of Tagalog and Cantonese language telephone speech. The enrollment utterances contain about 60 seconds of speech while the test utterances range from 10 to 60 seconds of speech.

In addition to equal error rate (EER), results are reported using the official performance metric for each SRE. For SRE10, this metric was the minimum of the normalized detection cost function (DCF) with $P_{\text{Target}} = 10^{-3}$ (*The 2010 NIST Speaker Recognition Evaluation* 2010). The primary SRE16 metric was a balanced (equalized) DCF averaged at two operating points (*NIST Speaker Recognition Evaluation 2016* 2016). The primary metrics are abbreviated to *DCF10* and *DCF16* respectively. Both EER and DCF were described in more detail in Section 2.3.

Table 4.1: EER(%) on NIST SRE 2010. The column labeled "10s-10s" denotes the condition where both the enroll and test utterances contain 10 seconds of speech. The columns labeled "5s," "10s," etc denote the amount of speech retained in the test utterances.

	10s-10s	**5s**	**10s**	**20s**	**60s**	**full**
i-vector	11.0	9.1	6.0	3.9	2.3	1.9
x-vector **a**	11.0	9.5	5.7	3.9	3.0	2.6
x-vector **b**	9.2	8.8	6.6	5.5	4.4	3.9
x-vectors	7.9	7.6	5.0	3.8	2.9	2.6
fusion	8.1	6.8	4.3	2.9	2.1	1.8

4.9 Results

In the following results, *i-vector* refers to the traditional i-vector baseline described in Section 4.7. The labels *x-vector* ***a*** and *x-vector* ***b*** denote the systems consisting of embeddings extracted from either embedding layer of the same DNN (see Section 4.5) and used as features to their own PLDA backends. The label *x-vectors* is the average of the PLDA backend scores for the individual x-vectors. In the following results, we focus on comparing the i-vector baseline with these combined embeddings. Finally, *fusion* refers to the equally weighted sum fusion of the PLDA scores of *i-vector* and *x-vectors*.

4.9.1 NIST SRE 2010

In this section, we look at performance on the SRE10 dataset described in Section 4.8. Tables 4.1 and 4.2 demonstrate the interplay between utterance-length

Table 4.2: DCF10 on NIST SRE 2010. The column labeled "10s-10s" denotes the condition where both the enroll and test utterances contain 10 seconds of speech. The columns labeled "5s," "10s," etc denote the amount of speech retained in the test utterances.

	10s-10s	**5s**	**10s**	**20s**	**60s**	**full**
i-vector	0.92	0.90	0.75	0.61	0.46	0.40
x-vector **a**	0.91	0.90	0.79	0.65	0.52	0.47
x-vector **b**	0.95	0.93	0.87	0.83	0.78	0.77
x-vectors	0.85	0.88	0.74	0.67	0.57	0.54
fusion	0.86	0.79	0.65	0.56	0.43	0.38

and performance. We see that i-vectors are still dominant for the longest test recordings, and outperform x-vectors at both the EER and DCF10 operating points. However, as the test utterance length decreases, the performance of the DNN embeddings degrades more gracefully relative to the baseline. At 20 seconds of test speech, the combined x-vectors are 3% better than i-vectors in EER but 8% worse at DCF10. With just 10 and 5 seconds of test speech, the x-vectors are 17% and 16% better in EER and slightly better at DCF10. The relative advantage of x-vectors appears to be largest when both enrollment and test utterances are short: in the column labeled *10s-10s* both the test **and** enroll utterances contain only about 10 seconds of speech, and we see that the combined x-vectors are 28% better in EER and 11% better in DCF10.

Since the i-vector and DNN embedding systems are so dissimilar, we expect good performance from their fusion. We observe an improvement over using i-vectors alone at all operating points and conditions. The largest

Table 4.3: EER(%) on NIST SRE 2016. The column labeled "Cantonese" reports results on the Cantonese portion of the dataset, while the column titled "Tagalog" reports results on the Tagalog portion. The column titled "pool" reports results on the entire dataset.

	Cantonese	**Tagalog**	**pool**
i-vector	8.3	17.6	13.6
x-vector **a**	7.7	17.6	13.1
x-vector **b**	7.8	17.4	13.1
x-vectors	6.5	16.3	11.9
fusion	6.3	15.4	11.3

Table 4.4: DCF16 on NIST SRE 2016. The column labeled "Cantonese" reports results on the Cantonese portion of the dataset, while the column titled "Tagalog" reports results on the Tagalog portion. The column titled "pool" reports results on the entire dataset.

	Cantonese	**Tagalog**	**pool**
i-vector	0.55	0.84	0.71
x-vector **a**	0.53	0.84	0.69
x-vector **b**	0.63	0.85	0.74
x-vectors	0.51	0.80	0.66
fusion	0.44	0.79	0.62

improvement is in EER on the 10s condition, which is 28% better than the baseline. Even on the full-length condition, where i-vectors are strongest, there is a 5% improvement over using the i-vectors alone, in both DCF10 and EER.

4.9.2 NIST SRE 2016

In this section, we evaluate the same systems from Section 4.9.1 on SRE16. Using the same embedding and i-vector systems for both SRE10 and SRE16 avoids the complexity of developing variants of each system that are optimized for different evaluations. However, this does cause a mismatch between the predominately English training data (used to optimize both systems) and the Tagalog and Cantonese evaluation speech. As a result, the performance reported here may lag behind that of counterparts optimized specifically for SRE16.

Tables 4.3 and 4.4 report performance in EER and DCF16 respectively. Pooled across languages, we see that the combined embeddings outperforms the i-vector baseline by 13% in EER and 7% in DCF16. After combining with i-vectors, the improvement increases to 17% in EER and 13% in DCF16.

Although the x-vectors also perform better on Tagalog, improvement is largest for the Cantonese portion. Compared to the i-vector baseline, the conbined x-vectors are 22% better in terms of EER and 7% in DCF16. The fused system is even better, and improves on the i-vector baseline by 24% in EER and 19% in DCF16.

4.10 Conclusions

In this chapter, we introduced a deep neural network embedding framework for text-independent speaker verification. Overall, the embeddings, which we call x-vectors, are competitive with a traditional i-vector baseline and are complementary when fused. We found that, although i-vectors are more effective on the full-length SRE10, the combined x-vectors are better on the short duration conditions. This underscores the findings of the previous chapter, that DNNs may be capable of producing more powerful representations of speakers from short speech segments. SRE16 presented the challenge of language mismatch between the predominantly English training data and the Cantonese and Tagalog evaluation. We saw that x-vectors outperformed i-vectors on both languages, suggesting that they may be more robust to this domain mismatch. Compared to the end-to-end approach described in Chapter 3, the work proposed in this chapter substantially reduces the training resources needed to achieve good performance. Nonetheless, the x-vector DNN is still quite data hungry, and would benefit from larger amounts of training data. In the next chapter, we will employ data augmentation techniques which artificially enlarge the amount and diversity of the training resources and result in greatly improved performance.

Chapter 5

Improving X-vectors with Data Augmentation

5.1 Overview

In this chapter, we use data augmentation to improve performance of the x-vector system that we introduced in Chapter 4. In Chapter 3, we saw that DNN embeddings leverage large-scale training datasets better than i-vectors. However, it can be challenging to collect substantial quantities of labeled data for training. We use data augmentation, consisting of adding noise and reverberation to the existing training speech, as an inexpensive method to multiply the amount of training data and to improve robustness.

We compare x-vectors against two standard i-vector baselines on Speakers in the Wild (SITW) and the Cantonese portion of NIST SRE 2016. We find that while augmentation is beneficial in the PLDA classifier, it is not helpful in the i-vector extractor. However, the x-vector DNN effectively exploits data augmentation, due to its supervised training, and achieves superior performance on the evaluation datasets.

5.2 The X-vector System

Our software implementation has been made available in the Kaldi toolkit. An example recipe is in the main branch of Kaldi at `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2` and a pretrained x-vector system can be downloaded from `http://kaldi-asr.org/models.html`.

The input features to the DNN are 24 dimensional filterbanks with a frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds. An energy SAD filters out nonspeech frames.

The DNN architecture is based on the x-vector system described in Section 4.3. The specific DNN configuration used here is outlined in Table 5.1. Suppose an input segment has T frames. The first five layers operate on speech frames, with a small temporal context centered at the current frame t.

Layer	**Layer context**	**Total context**	**Input x output**
frame1	$[t-2, t+2]$	5	120x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xN

Table 5.1: The embedding DNN architecture. X-vectors are extracted at layer *segment6*, before the nonlinearity. The N in the softmax layer corresponds to the number of training speakers.

For example, the input to layer *frame3* is the spliced output of *frame2*, at frames $t-3$, t and $t+3$. This builds on the temporal context of the earlier layers, so that *frame3* sees a total context of 15 frames.

We use the same statistics pooling layer introduced in Chapter 3. The pooling layer aggregates all T frame-level outputs from layer *frame5* and computes its mean and standard deviation. The statistics are 1500 dimensional vectors, computed once for each input segment. This process aggregates information across the time dimension so that subsequent layers operate on the entire segment. In Table 5.1, this is denoted by a layer context of $\{0\}$ and a total context of T. The mean and standard deviation are concatenated together and propagated through segment-level layers and finally the softmax output layer. The nonlinearities are all rectified linear units (ReLUs).

The DNN is trained to classify the N speakers in the training data. A training example consists of a chunk of speech features, which is about 3 seconds on average[1], and the corresponding speaker label. After training, the embedding is extracted from the affine component of layer *segment6* [2]. Excluding *segment7* and the softmax output layer (because they are not needed after training) there are a total of 4.2 million parameters in the DNN.

5.3 Baselines

In this section, we describe two i-vector baselines.

5.3.1 Acoustic I-vector

This baseline is a traditional i-vector system based on the GMM-UBM recipe described in (Snyder, Garcia-Romero, and Povey, 2015) and it serves as our acoustic-feature baseline system, which is referred to as *i-vector (MFCC)* in Table 5.2. It is very similar to the system used in Section 4.7 of the previous chapter. The features are 20 MFCCs with a frame-length of 25ms that are mean-normalized over a sliding window of up to 3 seconds. Delta and

[1]Empirically, we found that training on roughly 3 second segments produces the best results, even though the enrollment and test segments are often much longer than this.

[2]In Chapter 4 we extracted embeddings from two different layers and fused them together. In this chapter (and all subsequent chapters) we streamline the system, and only extract embeddings from the first affine layer after the statistics (called x-vector **a** previously).

acceleration are appended to create 60 dimension feature vectors. An energy-based speech activity detection (SAD) system selects features corresponding to speech frames. The UBM is a 2048 component full-covariance GMM. The system uses a 600 dimensional i-vector extractor and PLDA for scoring (see Section 5.4).

5.3.2 Phonetic Bottleneck I-vector

This i-vector system incorporates phonetic bottleneck features (BNF) from an ASR DNN acoustic model and is similar to (McLaren, Lei, and Ferrer, 2015). It is referred to as *i-vector (+BNF)* in Table 5.2. Its input features are 40 MFCCs with a frame-length of 25ms. Cepstral mean subtraction is performed over a sliding window of 6 seconds. The DNN has six layers, and a total left-context of 13 and a right-context of 9. The hidden layers use the p-norm (where $p = 2$) nonlinearity and have an input dimension of 3500 and an output dimension 350. The penultimate layer is a 60 dimensional linear bottleneck layer. The softmax output layer computes posteriors for 5297 triphone states. No fMLLR or i-vectors are used for speaker adaptation. Excluding the output layer, which is not needed to compute BNFs, the DNN has 9.2 million parameters.

The 60 dimensional BNFs are concatenated with the same 20 dimensional

MFCCs described in Section 5.3.1 plus deltas to create 100 dimensional features. The remaining components of the system (feature processing, UBM, i-vector extractor, and PLDA classifier) are identical to the acoustic system in Section 5.3.1.

5.4 PLDA Classifier

The same type of PLDA (Ioffe, 2006) classifier, described in Section 2.4.1.2, is used for the x-vector and i-vector systems. The representations (x-vectors or i-vectors) are centered, and projected using LDA. The LDA dimension was tuned on the SITW development set to 200 for i-vectors and 150 for x-vectors. After dimensionality reduction, the representations are length-normalized and modeled by PLDA. The scores are normalized using adaptive s-norm (Sturim and Reynolds, 2005; Cumani et al., 2011).

5.5 Experimental Setup

5.5.1 Training Data

The training data consists of both telephone and microphone speech, the bulk of which is in English. In order to match the sampling rate of the the telephone speech in the training data and that of the NIST SRE 2016 evaluation data, all wideband audio is downsampled to 8kHz.

The *SWBD* portion consists of Switchboard 2 Phases 1, 2, and 3 as well as Switchboard Cellular. In total, the SWBD dataset contains about 28k recordings from 2.6k speakers. The *SRE* portion consists of NIST SREs from 2004 to 2010 along with Mixer 6 and contains about 63k recordings from 4.4k speakers. In the experiments in Sections 5.6.1–5.6.4 the extractors (UBM/$\mathbf{T}$ for i-vectors or embedding DNN for x-vectors) are trained on SWBD and SRE and the PLDA classifier are trained on just SRE. Data augmentation, which is a focus of this chapter, is described in more detail in Section 5.5.3 and is applied to these datasets as explained throughout Section 5.6.

In the last experiment in Section 5.6.5 we incorporate audio from the new *VoxCeleb* dataset (Nagrani, Chung, and Zisserman, 2017) into both extractor and PLDA training lists. The dataset consists of videos from 1,251 celebrity speakers. Although SITW and VoxCeleb were collected independently, we discovered an overlap of 60 speakers between the two datasets. We removed the overlapping speakers from VoxCeleb prior to using it for training. This reduces the size of the dataset to 1,191 speakers and about 20k recordings.

The ASR DNN used in the i-vector (+BNF) system was trained on the Fisher English corpus. To achieve a limited form of domain adaptation, the development data from SITW and SRE16 is pooled and used for centering and score normalization. No augmentation is applied to these lists.

5.5.2 Evaluation

Our evaluation consists of two distinct datasets: Speakers in the Wild (SITW) (McLaren et al., 2016) and the Cantonese portion of the NIST SRE 2016 evaluation (SRE16) (*NIST Speaker Recognition Evaluation 2016* 2016). SITW consists of unconstrained audio from video of English speakers, with naturally occurring noises, reverberation, as well as device and codec variability. Here we only report results on the *CORE-CORE* condition of SITW, where each recording contains exactly one speaker. Chapter 8 studies speaker recognition on recordings containing multiple speakers, and there we report results on the more challenging multi-speaker conditions. The SRE16 portion consists of Cantonese conversational telephone speech. Both enrollment and test SITW utterances vary in length form 6–240 seconds. For SRE16, the enrollment utterances contain about 60 seconds of speech while the test utterances vary from 10–60 seconds.

We report results in terms of equal error rate (EER) and the minimum of the normalized detection cost function (DCF) at $P_{\text{Target}} = 10^{-2}$ and $P_{\text{Target}} = 10^{-3}$. Note that the SRE16 results have not been "equalized (*NIST Speaker Recognition*

Evaluation 2016 2016)." Refer to Section 2.3 for more information about the evaluation metrics.

5.5.3 Data Augmentation

Augmentation increases the amount and diversity of the existing training data. Our strategy employs additive noises and reverberation. Reverberation involves convolving room impulse responses (RIR) with audio. We use the simulated RIRs described by Ko et al. (2017), and the reverberation itself is performed with the multicondition training tools in the Kaldi *ASpIRE* recipe (Povey et al., 2011). For additive noise, we use the MUSAN dataset, which consists of over 900 noises, 42 hours of music from various genres and 60 hours of speech from twelve languages (Snyder, Chen, and Povey, 2015). The MUSAN dataset is described in more detail in Appendix A. Both MUSAN and the RIR datasets are freely available from `http://www.openslr.org`.

We use a 3-fold augmentation that combines the original "clean" training list with two augmented copies. To augment a recording, we choose between one of the following distortions randomly:

- **babble**: Three to seven speakers are randomly picked from MUSAN speech, summed together, then added to the original signal at 13-20dB SNR.

- **music**: A single music file is randomly selected from MUSAN, trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- **noise**: MUSAN noises are added at one second intervals throughout the recording (0-15dB SNR).
- **reverb**: The training recording is artificially reverberated via convolution with simulated RIRs.

5.6 Results

			SITW Core			**SRE16 Cantonese**		
Data Augmentation			**EER**	**DCF1**	**DCF2**	**EER**	**DCF1**	**DCF2**
5.6.1	None	i-vector (MFCC)	9.3	0.62	0.79	9.2	0.57	0.74
		i-vector (+BNF)	*9.1*	*0.56*	*0.72*	9.7	0.57	0.77
		x-vector	9.4	0.63	0.79	*8.0*	*0.49*	*0.70*
5.6.2	PLDA	i-vector (MFCC)	8.6	0.59	0.76	8.9	0.54	0.72
		i-vector (+BNF)	8.0	*0.51*	*0.69*	8.8	0.53	0.73
		x-vector	*7.6*	0.59	0.75	*7.5*	*0.46*	*0.67*
5.6.3	Extractor	i-vector (MFCC)	8.9	0.63	0.79	9.2	0.58	0.75
		i-vector (+BNF)	7.3	*0.53*	0.73	8.9	0.57	0.78
		x-vector	*7.2*	0.54	*0.72*	*6.3*	*0.43*	*0.63*
5.6.4	PLDA and extractor	i-vector (MFCC)	8.0	0.58	0.75	9.0	0.56	0.72
		i-vector (+BNF)	6.5	*0.49*	0.69	8.3	0.53	0.75
		x-vector	*6.0*	*0.49*	*0.68*	*5.9*	*0.41*	*0.59*
5.6.5	Added VoxCeleb	i-vector (MFCC)	7.5	0.55	0.72	9.2	0.56	0.74
		i-vector (+BNF)	6.1	0.47	0.66	8.1	0.52	0.75
		x-vector	*4.2*	*0.39*	*0.61*	*5.7*	*0.40*	*0.57*

Table 5.2: Results using data augmentation in various systems. "Extractor" refers to either the UBM/**T** or the embedding DNN. For each experiment, the best results are *italicized*.

The main results are presented in Table 5.2 and are referenced throughout Sections 5.6.1–5.6.5. We compare performance of two i-vector systems, labeled *i-vector (MFCC)* and *i-vector (+BNF)*, with the *x-vector* system. The systems are described in Sections 5.3.1, 5.3.2 and 5.2, respectively. Throughout the following sections, we use the term *extractor* to refer interchangeably to either the UBM/**T** (for i-vector) or the embedding DNN (for x-vector).

5.6.1 Original Systems

In this section, we evaluate systems without data augmentation. The extractors are trained on the SWBD and SRE datasets described in Section 5.5.1. The PLDA classifiers are trained on just the SRE dataset. Without using augmentation, the best results on SITW are obtained by i-vector (+BNF), which is 12% better than the x-vector system at DCF10^{-2}. The MFCC i-vector system also achieves slightly lower error rates than the x-vector system on SITW. However, even without augmentation, the best results for SRE16 Cantonese are obtained by the x-vectors. In terms of DCF10^{-2}, these embeddings are about 14% better than either i-vector system. We observe that i-vector (+BNF) has no advantage over i-vector (MFCC) for this Cantonese speech. This echoes recent studies that have found that the large gains achieved by BNFs in English speech are not necessarily transferable to non-English data (Novotný et al., 2016).

5.6.2 PLDA Augmentation

In this experiment, the augmentation strategy described in Section 5.5.3 is applied to only the PLDA training list. We use the same extractors as the previous section, which were trained on the original datasets. PLDA augmentation results in a clear improvement for all three systems relative to Section 5.6.1. However, it appears that the x-vectors may benefit from the PLDA augmentation more than the baseline systems. On SITW, the x-vector system achieves slightly lower error rates than i-vector (MFCC), but continues to lag behind i-vector (+BNF) at most operating points. On SRE16, the x-vectors maintain an advantage over the i-vectors by about 14% in DCF10^{-2}.

5.6.3 Extractor Augmentation

We now apply data augmentation to the extractor (UBM/$\mathbf{T}$ or embedding DNN) training lists but *not* the PLDA list. The effect of augmenting the UBM/$\mathbf{T}$ is inconsistent in the i-vector system. This observation is supported by prior studies on i-vectors, which have found that augmentation is only effective in the PLDA classifier (Lei et al., 2012; Garcia-Romero, Zhou, and Espy-Wilson, 2012). On the other hand, augmenting the embedding DNN training list results in a large improvement. In contrast to the i-vector systems, this is considerably more effective than augmenting the PLDA training list.

On SITW, the x-vector system achieves lower error rates than i-vector (MFCC) and has now caught up to the i-vector (+BNF) system. On SRE16, the x-vectors are now 25% better than the i-vectors in DCF10^{-2}, which is almost double the improvement the DNN embeddings had with PLDA augmentation alone. The findings in this section indicate that data augmentation is only beneficial for extractors trained with supervision.

5.6.4 PLDA and Extractor Augmentation

In the previous sections, we saw that PLDA augmentation was helpful in both i-vector and DNN embedding systems, although extractor augmentation was only clearly beneficial in the embedding system. In this experiment, we apply data augmentation to both the extractor *and* PLDA training lists. We continue to use SWBD and SRE for extractor training and only SRE for PLDA. On SITW the x-vectors are now 10-25% better than i-vector (MFCC) and are slightly better than i-vector (+BNF) at all operating points. On SRE16 Cantonese, the x-vectors continue to maintain the large lead over the i-vector systems established in Section 5.6.3.

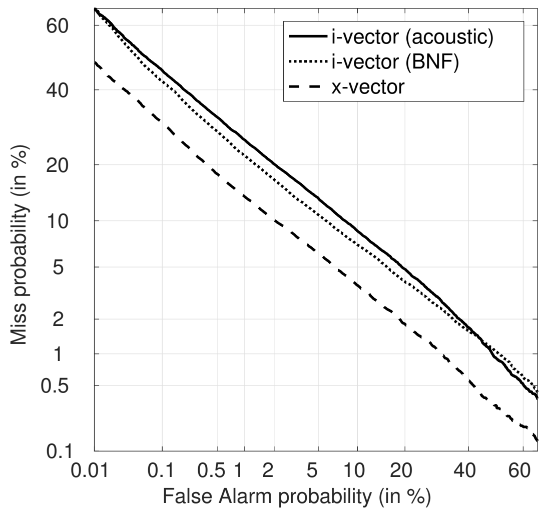


Figure 5.1: DET curve for the Cantonese portion of NIST SRE16 using Section 5.6.5 systems.

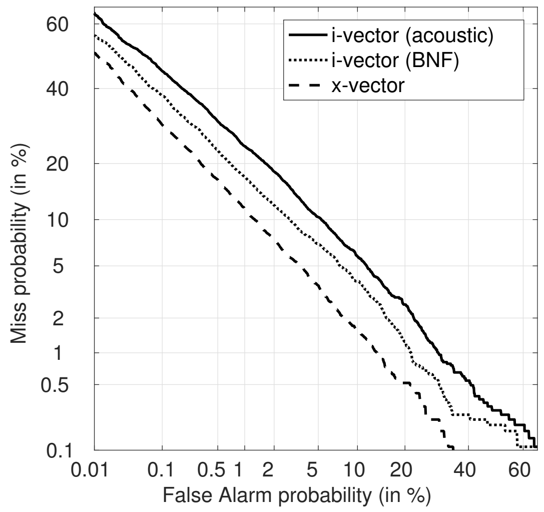


Figure 5.2: DET curve for the SITW Core using Section 5.6.5 systems.

5.6.5 Adding VoxCeleb with Data Augmentation

The training data in Sections 5.6.1–5.6.4 is dominated by telephone speech. In this experiment, we explore the effect of adding a large amount of microphone speech to the systems in Section 5.6.4. The VoxCeleb dataset (Nagrani, Chung, and Zisserman, 2017) is augmented, and added to both the extractor and PLDA lists. As noted in Section 5.5.1, we found 60 speakers who overlap with SITW; all speech from these speakers were removed from the training lists.

On SITW, both i-vector and x-vector systems improve significantly. However, the x-vector exploits the large increase in the amount of in-domain data better than the i-vector systems. Compared to i-vector (MFCC), the x-vectors are better by 44% in EER and 29% in DCF10^{-2}. Compared to the i-vector (+BNF) system, it is now better by 32% in EER and 17% in DCF10^{-2}. On SRE16, the i-vector systems remain roughly the same as they were in Section 5.6.4, but the x-vectors improve on all operating points by a small amount. These results are illustrated by detection error tradeoff (DET) plots in Figures 5.1 and 5.2.

5.7 Conclusions

In this section, we found that data augmentation is an easily implemented and effective strategy for improving x-vector performance. We also made the x-vector system – our implementation of DNN embeddings – available in the Kaldi toolkit. We found that the x-vector system significantly outperformed two standard i-vector baselines on SRE16 Cantonese. After including a large amount of augmented microphone speech, the x-vectors achieved much lower error rates than our best baseline on Speakers in the Wild. Bottleneck features from an ASR DNN are used in our best i-vector system, and so it requires transcribed data during training. On the other hand, the x-vector DNN needs only speaker labels to train, making it potentially ideal for domains with little transcribed speech. More generally, it appears that x-vector-based systems achieve state-of-the-art performance on standard evaluation datasets and are now a strong contender for the next-generation representation for speaker recognition.

In the next chapter, we will deviate from speaker recognition, and apply this work to the task of spoken language recognition.

Chapter 6

Spoken Language Recognition using X-vectors

The previous chapters introduced the problem of speaker recognition, and developed several neural network-based solutions to the task, which began with the end-to-end approach described in Chapter 3 and culminated in the x-vector architecture of Chapter 5. In this chapter, we adapt these techniques to the task of spoken language recognition, and demonstrate that they outperform state-of-the-art baselines based on i-vectors. The deep neural network is trained here to predict language labels, and maps sequences of speech features to fixed-dimensional embeddings, which capture the language characteristics of the audio. Once extracted, the embeddings (still called x-vectors) utilize the same classification technology developed for i-vectors. In the 2017 NIST language recognition evaluation, this x-vector-based system outperformed all others in the evaluation. In the post-evaluation analysis presented here, we experiment with several variations of the x-vector framework, and show that the best performing system uses multilingual bottleneck features, data augmentation, and a discriminative Gaussian classifier.

6.1 Introduction

The National Institute for Standards and Technology (NIST) organized it's 8th language recognition evaluation (LRE) in 2017, with an ongoing goal of promoting development in language recognition technology and measuring its performance. Like the previous evaluation in 2015, the NIST LRE 2017 focused on distinguishing between dialects and closely related languages. It was separated into an unconstrained *open* training condition and a *fixed* training condition, in which only a specified set of training resources were permitted. In addition to conversational telephone speech and broadcast speech data[1], this evaluation also included wideband speech extracted from videos.

[1] The broadcast news portion of LRE 2017 was downsampled to 8kHz before the data was released to the participants.

Most modern language recognition systems are based on i-vectors (Dehak et al., 2011), which was described in more detail in Section 2.4.1.1. Once extracted, i-vectors are commonly classified using Gaussian models, logistic regression, or cosine scoring. State-of-the-art i-vector systems incorporate deep neural networks (DNN) trained as acoustic models for automatic speech recognition (ASR). This is generally done in one of two ways: either posteriors from the ASR DNN replace those from a Gaussian mixture model (GMM) or the i-vector system is trained on bottleneck features extracted from the DNN (Song et al., 2013; Matejka et al., 2014; Richardson, Reynolds, and Dehak, 2015; Fér et al., 2015; McCree, Sell, and Garcia-Romero, 2016).

Methods to directly capture language characteristics using DNNs have also been proposed recently (Lopez-Moreno et al., 2014; Gonzalez-Dominguez et al., 2014; Gonzalez-Dominguez et al., 2015; Garcia-Romero and McCree, 2016). The approaches prior to this dissertation typically train a DNN to classify language at the frame-level, and utterance-level language scores are calculated by averaging frame-level log-posterior probabilities across time.

In this work, we adapt the x-vector framework with augmentation that we described in Chapter 5, to language recognition. This framework consists of a discriminatively trained DNN that maps variable-length speech segments to fixed-dimensional embeddings, that are still called x-vectors. Once extracted, x-vectors are used as a representation of speech like i-vectors. This allows for leveraging the classification and backend technology that has been developed for i-vector-based language recognition systems.

6.2 X-vector System

6.2.1 Overview

The x-vector system is based on a framework that we developed for speaker recognition that is described in Chapter 4 and 5. The system is comprised of a feed-forward DNN that maps variable-length speech segments to embeddings that we call *x-vectors*. Once extracted, the x-vectors are classified by a discriminatively trained Gaussian classifier, as described in Section 6.4.

The network is implemented using the nnet3 neural network library in the Kaldi Speech Recognition Toolkit (Povey et al., 2011). The recipe is based on the SRE16 v2 recipe available in the main branch of Kaldi as `https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2`. The training classes and features have been modified for language recognition.

Layer	**Layer context**	**Tot. context**	**In x out**
frame1	$[t-2, t+2]$	5	$5F$x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	$1500T$x3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xL

Table 6.1: The standard x-vector DNN architecture. X-vectors are extracted at layer *segment6*, before the nonlinearity. The input layer accepts F-dimensional features. The L in the softmax layer corresponds to the number of training languages.

6.2.2 Architecture

The DNN configuration is outlined in Table 6.1. The architecture is nearly the same as the one we used for speaker recognition in Section 5.5. The main differences are that the features are different (either MFCCs or bottleneck features from an ASR DNN) and the DNN is trained to classify languages, rather than speakers.

6.2.3 Training

The network is trained to classify languages using a multiclass cross entropy objective function:

$$E = -\sum_{n=1}^{N}\sum_{l=1}^{L} d_{nl} \log P(l \mid \mathbf{x}_{1:T}^{(n)}) \tag{6.1}$$

The quantity d_{nl} is 1 if the language label for segment n is l, otherwise it's 0.

The primary difference between the training here and in (Gonzalez-Dominguez et al., 2014; Gonzalez-Dominguez et al., 2015; Garcia-Romero and McCree, 2016) is that our system is trained to classify languages from variable-length segments, rather than at the frame level with a fixed context of frames. Suppose there are L languages in N training segments. Then $P(l \mid \mathbf{x}_{1:T}^{(n)})$ is the probability of language l given T input frames $\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ...\mathbf{x}_T^{(n)}$.

Training examples are constructed by picking speech chunks from the training data that are 2–4 seconds long. The corresponding target is the language label. The network is trained for several epochs using stochastic natural gradient descent (Povey, Zhang, and Khudanpur, 2015).

6.2.3.1 Data Augmentation

We use augmentation to increase the amount and diversity of the x-vector DNN training data. We employ the same strategy we used for speaker recognition, which is described in Section 5.5.3, but with the addition of *speed perturbation* as an augmentation type. Speed perturbation alters the speed of the speech signal using a specified speed factor (Ko et al., 2015). The MUSAN dataset is used for additive noises, and is described in Appendix A.

We use a 6-fold augmentation strategy that combines the original "clean" training list with up to 5 augmented copies[2]. To augment a recording, we randomly choose between one of the following:

- **speed perturbation**: Apply a speed factor of 0.9 or 1.1 to slow down or speed up the original recording.
- **music**: A single music file (without vocals) is randomly selected from MUSAN (see Appendix A), trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- **noise**: MUSAN noises are added at one second intervals throughout the recording (0-15dB SNR).
- **reverb**: The training recording is artificially reverberated via convolution

[2]Note that we may apply the same type of augmentation more than once to the same recording.

with simulated RIRs.

6.2.4 Embeddings

At test time, x-vectors are extracted at layer *segment6* of the network (see Table 6.1), before the nonlinearity. Since the pooling layer aggregates across the input frames, we are able to extract a single 512-dimensional x-vector from each variable-length speech segment.

Alternatively, the x-vector DNN could be used to classify languages directly. This possibility is explored in Section 6.6.5. However, we find that extracting embeddings is more flexible, as it facilitates integration with the backend technology developed for i-vectors. Also, it allows expanding to new languages without needing to retrain the x-vector extractor. The latter option is explored in Section 6.6.6.

6.2.5 Features

We experiment with acoustic features and two types of bottleneck features. Before being fed into the x-vector DNN, features are mean-normalized over a 3-second sliding window, and nonspeech frames are removed using an energy-based speech activity detection system.

6.2.5.1 Acoustic Features

The acoustic features are 23 MFCCs computed from a sampling window of 25ms.

6.2.5.2 Fisher English Bottleneck Features

For the fixed training condition of LRE17, 60-dimensional linear bottleneck features (BNF) are extracted from an ASR DNN trained on the Fisher English corpus, based on the network described in (Snyder, Garcia-Romero, and Povey, 2015). The DNN is a time-delay acoustic model built using the nnet2 library in Kaldi. It uses the same architecture and training recipe as the system described in Section 5.3.2.

6.2.5.3 Multilingual Bottleneck Features

For the open training condition, we use 60-dimensional BNFs extracted from an ASR DNN trained on multiple languages. Up to and including the bottleneck layer, the DNN has the exact same architecture as the one described in Section 6.2.5.2. It also uses the same MFCC features.

The DNN is trained on 23 languages from the IARPA Babel dataset (Amharic, Cebuano, Guarani, Javanese, Lao, Tagalog, Tokpisin, Zulu, Assamese, Dholuo, Haitian, Kazakh, Lithuanian, Pashto, Tamil, Turkish, Cantonese, Igbo, Kurmanji, Mongolian, Swahili, Telugu, and Vietnamese). The hidden layer parameters are shared across languages. Each language is given a separate output layer that computes posteriors over 4300 to 4900 subphonetic context dependent states, depending on the language.

6.3 I-vector Baseline

Our baselines consist of two state-of-the-art joint i-vector systems introduced by McCree, Sell, and Garcia-Romero (2016). These systems were part of our LRE17 submission, and are described in greater detail in (Mccree et al., 2018). The joint system differs from the classical i-vector system in that both the means and the *weights* of the GMM that comprises the UBM are adapted to each training or test recording. This allows the i-vector to model both acoustic and phonotactic variability.

The first joint i-vector system uses posteriors from an ASR acoustic model trained on Fisher English, and is very similar to the system described in (Snyder, Garcia-Romero, and Povey, 2015) for speaker recognition. We refer to systems built using this approach as "senone" systems. The second system uses BNFs extracted from a different acoustic model, which is also trained on

Fisher English, as described in Section 6.2.5.2.

6.4 Classifier

We use the same classifier architecture for x-vectors and i-vectors. Prior to classification, embeddings are whitened and length normalized (Garcia-Romero and Espy-Wilson, 2011), and their dimensionality is reduced using linear discriminant analysis (LDA). Classification is performed by a discriminatively-trained Gaussian classifier (McCree, 2014). When fusing the decisions made by multiple language recognition systems, we learn a weight for each system, and average the weighted scores.

The whitening matrix, LDA, and classifier were trained on an augmented version of the LRE17 training data, as described in Section 6.5.2.1. A held-out portion was used for estimating the fusion weights. Training recordings were segmented to durations of 3-60 seconds of speech. In addition to applying data augmentation similar to what is described in Section 6.2.3.1, we also apply additional augmentations we found helpful in previous evaluations: simulation of the GSM-AMR phone encoding, addition of babble noise from Fisher, and synthetic, low-frequency modulated noise.

Cluster	Languages
Arabic	Egyptian Arabic, Iraqi Arabic, Levantine Arabic, Maghrebi Arabic
Chinese	Mandarin, Min Nan
English	British English, General American English
Slavic	Polish, Russian
Iberian	Caribbean Spanish, European Spanish, Latin American Continental Spanish, Brazilian Portuguese

Table 6.2: The NIST LRE 2017 language clusters.

6.5 Corpora

6.5.1 NIST LRE 2017

We evaluate performance on the 2017 NIST language recognition evaluation (LRE) benchmark data set (*NIST 2017 Language Recognition Evaluation Plan* 2017). It consists of 14 languages in 5 language clusters, as described in Table 6.2. The data came from conversational telephone speech and broadcast narrowband speech data from the LDC MLS14 corpus, as well as wideband speech extracted from internet videos from the VAST corpus (Tracey and Strassel, 2018). The focus of the evaluation is on differentiating between closely related languages/dialects *within* each of the clusters listed above.

Training conditions were broken down into *fixed* and *open* training conditions. The fixed condition permits use of only past LREs, Fisher English, Switchboard corpora and the LRE17 development data, as well as publicly available noise corpora. The open condition lifts these restrictions.

6.5.2 Training Data

This section describes the training data used by both the i-vector baselines and x-vector systems. The ASR DNNs were trained on additional corpora, as described in Section 6.2.5.

6.5.2.1 NIST LRE 2017 Training and Development

This dataset consists of about 18,000 training segments and 4,000 development segments provided by NIST for the evaluation participants. It is comprised of narrowband and wideband data collected by LDC for the MLS14 and VAST corpora and contains speech from the 14 target languages/dialects in Table 6.2. Due to the small amount of wideband training data, it too was downsampled to 8kHz before training, to match the sampling rate of the majority of the training data.

6.5.2.2 2015 NIST IVC

This dataset consists of audio from the i-vector Machine Learning Challenge (IVC) (*The 2015 Language Recognition i-Vector Machine Learning Challenge* 2015). This dataset is composed primarily of previous NIST LREs. It consists of data collected over the telephone channel. In total, there are 177,000 speech segments with an average duration of 35 seconds from 57 languages. This dataset is used only in some experiments, as described below in Section 6.6.6.

6.6 Results

Results are reported in terms of the C_{primary} metric described in (*NIST 2017 Language Recognition Evaluation Plan* 2017). It is the average of the actual detection costs, described earlier in Section 2.3, computed at $P_{\text{target}} = 0.1$ and $P_{\text{target}} = 0.5$. In the following tables, the *Overall* results for each category of the evaluation (e.g., language or microphone type) have been equalized by the NIST scoring tool, as described in (*NIST 2017 Language Recognition Evaluation Plan* 2017). As a result, scores from each data source (VAST or MLS14) or language have been balanced and contribute equally to the metric. Note however, that the costs are computed using a single threshold, rather than a different threshold for each category.

6.6.1 Baseline

In this section, we compare performance of the two joint i-vector systems described in Section 6.3 with that of the x-vector system. The system labeled

System	MLS14	VAST	Overall
1 ivec_senone	0.193	0.217	0.205
2 ivec_bnf	0.170	0.206	0.189
3 xvec_bnf	0.148	0.178	0.163
1,2 fusion	0.151	0.183	0.167
1,3 fusion	0.130	0.168	0.150
2,3 fusion	0.125	0.164	0.145
1,2,3 fusion	0.124	0.163	0.144

Table 6.3: Comparison of the C_{primary} of two joint i-vectors systems with that of the x-vector system on the LRE17 evaluation set. Various fusions are reported in the second half of the table. All systems conform to the fixed training condition.

ivec_senone uses posteriors from an ASR DNN trained on Fisher English and ivec_bnf is trained on Fisher English BNFs (similar to the BNFs used in Section 6.2.5.2). The system xvec_bnf is an x-vector system trained on the Fisher English BNFs from Section 6.2.5.2.

In Table 6.3 we see that xvec_bnf achieves lower C_{primary} than either i-vector system on both the MLS14 and VAST parts of the evaluation. Overall, xvec_bnf outperforms ivec_senone by about 20% and ivec_bnf by about 14% relative.

The last 4 rows of Table 6.3 report score fusion results. By itself, xvec_bnf achieves a C_{primary} roughly equal to the fusion of the two i-vector systems. Moreover, the x-vector and i-vector systems appear to be very complementary when fused; the fusion of ivec_bnf and xvec_bnf is 23% better than our best baseline, and 13% better than the fusion of the two i-vector systems.

	MLS14			VAST
System	**3s**	**10s**	**30s**	
1 ivec_senone	0.216	0.178	0.084	0.217
2 ivec_bnf	0.194	0.156	0.064	0.206
3 xvec_bnf	0.169	0.135	0.061	0.178
1,2 fusion	0.171	0.140	0.062	0.183
1,3 fusion	0.148	0.125	0.055	0.168
2,3 fusion	0.143	0.119	0.050	0.164
1,2,3 fusion	0.142	0.117	0.052	0.163

Table 6.4: C_{primary} for i-vector and x-vector systems as well as their fusions. MLS14 results are broken own by test segment duration, as indicated by column labels. The VAST subset is 10 seconds or longer.

6.6.2 Duration Analysis

NIST LREs typically include test segments with durations clustered around 3, 10 and 30 seconds. This facilitates the analysis of performance as a function of utterance duration. For LRE17 only the MLS14 portion of the data was provided with these duration patterns. The VAST data had a wider variety of segment duration, with each segment being 10 seconds or longer.

Table 6.4 shows results on the MLS14 test set by duration. It is clear that the shorter segments are harder to classify by all systems. Also, even though the VAST data had longer duration segments, the domain shift with respect to the majority of our training data (telephone and broadcast speech versus internet videos) results in performance that is worse than the 3 second MLS14 subset. Since the x-vector system was trained on examples that are

System	MLS14	VAST	Overall
xvec_mfcc	0.209	0.203	0.206
xvec_bnf	0.148	0.178	0.163
xvec_mlbnf	0.130	0.149	0.140

Table 6.5: X-vector performance as a function of feature type. Results are reported in terms of C_{primary}, using MFCCs, Fisher English BNFs or multilingual BNFs. The first two systems conform to the fixed training condition, whereas the last system meets the requirements of the open training condition.

on average 3 seconds long, it was expected that it might achieve its largest gains (compared to the baselines) on the shortest test segments. However, Table 6.4 demonstrates no clear relationship between the x-vector's relative performance and duration: compared to the baselines, xvec_bnf is 13% and 22% better on the 3 second segments, 13% and 24% better on the 10 second segments, and 5% and 27% better on the 30 second segments.

6.6.3 X-vector Features

It has been well documented that i-vector-based language recognition systems improve greatly by incorporating ASR DNNs. In this section, we see how this observation generalizes to x-vectors, by comparing systems trained on MFCCs (xvec_mfcc), Fisher English BNFs (xvec_bnf), and multilingual BNFs (xvec_mlbnf). See Section 6.2.5 for a description of these these features.

In Table 6.5 we see that x-vector performance echoes the trend observed

System	MLS14	VAST	Overall
xvec_mlbnf_no_aug	0.152	0.179	0.166
xvec_mlbnf	0.130	0.149	0.140

Table 6.6: Comparison of performance with or without augmenting the x-vector DNN training list. In either case, the classifier (Section 6.4) uses the same augmented list.

in i-vectors: monolingual BNFs perform much better than acoustic features alone, but multilingual BNFs are the best choice (Fér et al., 2015). Using just MFCCs, xvec_mfcc achieves performance comparable to ivec_senone from the previous section. However, replacing MFCCs with Fisher English BNFs improves overally performance by 21% in C_{primary}. Finally, substituting MFCCs with multilingual BNFs reduces C_{primary} even further, by 32%.

6.6.4 Data Augmentation

Next, we test the importance of augmenting the x-vector DNN training data. In both the unaugmented and data augmented training setups, the features are multilingual BNFs and the Gaussian classifier still uses the same augmentation strategy as described in Section 6.4.

In Table 6.6 we observe that removing augmentation increases detection cost by over 10%. This is likely due to augmentation increasing the limited amount of training data, as well as making the system more robust against

System	MLS14	VAST	Overall
xvec_mlbnf_direct	0.155	0.256	0.206
xvec_mlbnf	0.130	0.149	0.140

Table 6.7: Comparison of performance using the x-vector DNN for direct classification, or as features for the Gaussian classifier from Section 6.4.

degraded audio. This result parallels our observations when training x-vectors for speaker recognition in Chapter 5.

6.6.5 Direct Classification vs. Embeddings

The x-vector framework is based on our speaker recognition work in Chapter 5, where the goal is to produce embeddings that generalize to unseen speakers. However, in a closed-set language recognition task, the x-vector DNN can be used directly for classification (without the need for a backend classifier), provided it is trained on the same language classes as required for deployment. In this section, direct classification is compared with using embeddings extracted from the same system.

In Table 6.7 we see that using embeddings to train the Gaussian classifier achieves much better performance than using the system directly for classification. In particular, the direct system appears to suffer from the limited amount of VAST training data. While xvec_mlbnf is only 16% better than xvec_mlbnf_direct on MLS14 it is 42% better on VAST.

Although it's likely the direct results could be improved with hyper-parameter tuning and calibration in the backend, these results still underscore the flexibility of our standard x-vector approach. Once extracted, x-vectors can be fed into the same pipeline used for i-vectors, taking advantage of existing classifier and backend technology that assists in domain adaptation and calibration.

6.6.6 Adding New Languages without Retraining

Another advantage of extracting embeddings from a DNN, rather than using it for direct classification, is that it opens up the possibility of deploying the system with a different set of languages without having to retrain the x-vector extractor. Given a multilingual training list with sufficient diversity, it may be possible to train the DNN to produce embeddings that will extend to differentiating between languages or dialects not present in the training data. In this section, we simulate this by training an x-vector DNN called xvec_mlbnf_57lang using the 57 languages of the IVC dataset (see Section 6.5.2.2). Although different training data is used, the same augmentation strategy (see Section 6.2.3.1) we used for xvec_mlbnf is applied to xvec_mlbnf_57lang. It is important to note that the IVC dataset may contain a significant acoustic domain mismatch with the LRE17 evaluation data as it is

System	MLS14	VAST	Overall
xvec_mlbnf_57lang	0.153	0.183	0.168
xvec_mlbnf	0.130	0.149	0.140

Table 6.8: Performance with an x-vector DNN trained on the IVC dataset (Section 6.5.2.1) or on the LRE17 development data (Section 6.5.2).

from a different collection. In either system, we use the same in-domain data to train the backend classifier.

In Table 6.8 we see that xvec_mlbnf_57lang lags behind the system trained on completely in-domain data and matching languages. Nonetheless, it achieves similar performance as xvec_mlbnf_no_aug. In the future, we plan to expand the number of languages as well as the domains represented in the x-vector training data to further explore this approach.

6.7 Conclusions

In this chapter, we adapted the x-vector framework, which was originally developed for speaker recognition, to the task of language recognition. We demonstrated that x-vectors achieved excellent performance on the NIST LRE 2017, outperforming several state-of-the-art i-vector systems. We explored several variations to the basic x-vector framework. We found that, like in i-vector systems, ASR bottleneck features greatly improved performance over acoustic features. Echoing results in speaker recognition, our experiments showed that augmenting the x-vector DNN training data is highly beneficial. Finally, although the framework permits direct classification, we found that extracting x-vectors from the DNN and using them as features for a Gaussian classifier produced much better results.

Chapter 7

Speaker Diarization

In the previous chapter, we showed that x-vectors can be adapted to the task of spoken language recognition. In this chapter, we will show that they can be successfully applied to the problem of speaker diarization as well. Speaker diarization is an important front-end for speech technology that handles multiple speakers. It is the process of grouping segments of speech according to the speaker, and is sometimes referred to as the "who spoke when" task. Until recently, most speaker diarization systems used i-vectors as a core component. In a typical diarization system, i-vectors are extracted from short segments of speech, and clustered, to discover the individual speakers in a recording. However, diarization performance, even in relatively clean settings is still very poor. In this chapter, we show how i-vectors can be removed from the diarization process, and replaced with x-vectors. Performance is evaluated on several diarization benchmarks, and we demonstrate that the x-vector system exceeds the performance of a strong baseline system.

7.1 Introduction

Most research in speech processing assumes that there is only one speaker per recording, and the majority of standard evaluation datasets reflect this assumption. However, audio collected from many real-world environments, such as broadcast news or smart home devices (Amazon Echo or Apple HomePod, for instance) violate this single-speaker assumption. When ASR, for example, is performed on this multi-speaker audio, it is often necessary to both transcribe the speech and to identify the speakers that produced each utterance. Solutions employ a preprocessing step, known as speaker diarization, which precedes the application of any downstream speech technology, such as ASR.

In a typical diarization system, i-vectors are extracted from short segments of speech (Shum et al., 2011; Shum, Dehak, and Glass, 2012; Shum et al., 2013; Senoussaoui et al., 2014; Sell and Garcia-Romero, 2014) and are clustered. This is a sensible approach given the success of i-vectors for speaker recognition. In this chapter, we propose replacing i-vectors for diarization with x-vectors, which in Chapter 5, were shown to be highly effective for capturing speaker characteristics. Furthermore, the results in Chapters 3 and 4 suggest that these embeddings are more effective relative to i-vectors for shorter durations of speech. This is a highly desirable property, as the segment length used in diarization is typically between 1 and 2 seconds of speech.

After presenting a background on diarization technologies, we will describe the proposed system. Then, we will present the results on several evaluation datasets. In Section 7.4.1 we will present results on the popular CALLHOME telephony dataset. Section 7.4.2 presents results on the CHIME 5 dataset, which consists of challenging far-field speech, and finally, in Section 7.4.3 we evaluate performance on the DIHARD 2018 challenge dataset, which consists of speech from several languages and diverse recording conditions. Finally, in Section 7.5 we conclude the chapter.

7.2 Background

7.2.1 Diarization with I-vectors

Soon after their development for speaker recognition, Shum et al. (2011) and Shum, Dehak, and Glass (2012) adapted i-vectors to the task of speaker diarization. Early work utilized i-vectors extracted from short segments of speech, and used cosine scoring as a distance metric, and clustered using K-means or spectral-based techniques (Shum et al., 2011; Shum, Dehak, and Glass, 2012). Other clustering algorithms on i-vectors for diarization have included Variational Bayesian GMMs (Shum et al., 2013), mean shift (Senoussaoui et al., 2014), and agglomerative hierarchical clustering (AHC) (Kenny, Reynolds, and Castaldo, 2010; Sell and Garcia-Romero, 2014). Since it tends to achieve the best performance, the work that follows in this chapter will use AHC exclusively. In a study by Sell and Garcia-Romero (2014), it was shown that cosine scoring can be outperformed by PLDA. Followup work by Sell and Garcia-Romero (2015a), showed that additional improvements could be obtained by replacing the unsupervised GMM-UBM with the senone partitions from a DNN trained for ASR.

One shortcoming of the segmentation-based approaches described above is that the resulting diarization marks will be restricted to arbirary begin and end times according to the segmentation boundaries. To remedy this, a second stage of diarization, often called resegmentation, can be added. In resegmentation, the clustering results are used to initialize a frame-level diarization system that then iterates to refine the boundaries of speaker turns. Previously, most resegmentation was performed in the acoustic feature space with a Hidden Markov Model (HMM), but recent work has shown that resegmentation can be more effective using subspace techniques (Sell and Garcia-Romero, 2015b). In the remainder of this chapter, we focus solely on the initial clustering stage of the diarization system, and do not study performance with second-stage refinement.

7.2.2 Diarization with Neural Networks

Mirroring progress in speaker recognition, recent systems have replaced i-vectors with DNN-based embeddings for capturing speaker characteristics for diarization (Garcia-Romero et al., 2017a; Sell et al., 2018; Wang et al., 2018). In Chapter 3 we described an end-to-end DNN that jointly learns a fixed-dimensional embedding and a scoring metric. Garcia-Romero et al. (2017b), applied this method to diarization, and found that it achieves similar performance to a state-of-the-art diarization system. Sell et al. (2018) built on this work, by using the x-vector architecture introduced in Chapter 5. In that work, x-vectors simply replace i-vectors as the short segment-level embedding, and we continue to use PLDA to compare embeddings, and AHC to cluster. The work presented in this chapter uses the same paradigm.

Other work has also explored using DNNs to process multi-speaker audio, though in different contexts. Hershey et al. (2016) used neural embeddings to *separate* the speech of multiple overlapping speakers. In that work, a

DNN is trained to provide features for unsupervised clustering, resulting in a process called deep clustering. Deep clustering operates on each time-frequency bin and learns embeddings for those bins with desirable behavior in Euclidian space. Although this approach could be viewed a superset of diarization, in practice it is typically used to separate two overlapping speakers in short segments of speech, and may be used in conjunction with a standard diarization system.

7.2.3 Performance Metrics

Speaker diarization performance is commonly measured with the diarization error rate (DER). In its purest form, DER combines three types of error:

- **missed speech**: when speech activity detection (SAD) fails to detect speech
- **false alarm speech**: when SAD erroneously detects speech
- **speaker error**: when speech is assigned to the incorrect speaker

DER therefore includes both speaker labeling errors and errors due to audio being misclassified as speech or nonspeech. Therefore, it is sometimes useful to use the oracle SAD marks, to focus exclusively on speaker labeling errors. In Section 7.4 we will present results both with oracle SAD marks and using an automatic SAD system.

Tools for computing DER are quite flexible, and provide several options to control how strict the error metric is. On some corpora, such as CALLHOME, the convention is to ignore any errors within 250ms of a speaker transition, and errors due to overlapping speakers. However, in some datasets, such as the DIHARD challenge, emphasis is placed on correctly attributing all speech, including overlapping speech, and errors in overlapped speech regions are also included in the DER.

7.3 Experimental Setup

7.3.1 Diarization System

Figure 7.1 illustrates the schema of a typical diarization system. The speech segments are identified as grey rectangles. Before diarization is performed, the number of speakers and their location in the recording is unknown, which is indicated by question marks in the figure. The output of the diarization system is the input speech segments, but now (if we've performed diarization accurately) we've identified how many speakers are present in the recording, and their location. The steps of the diarization system are as described in the

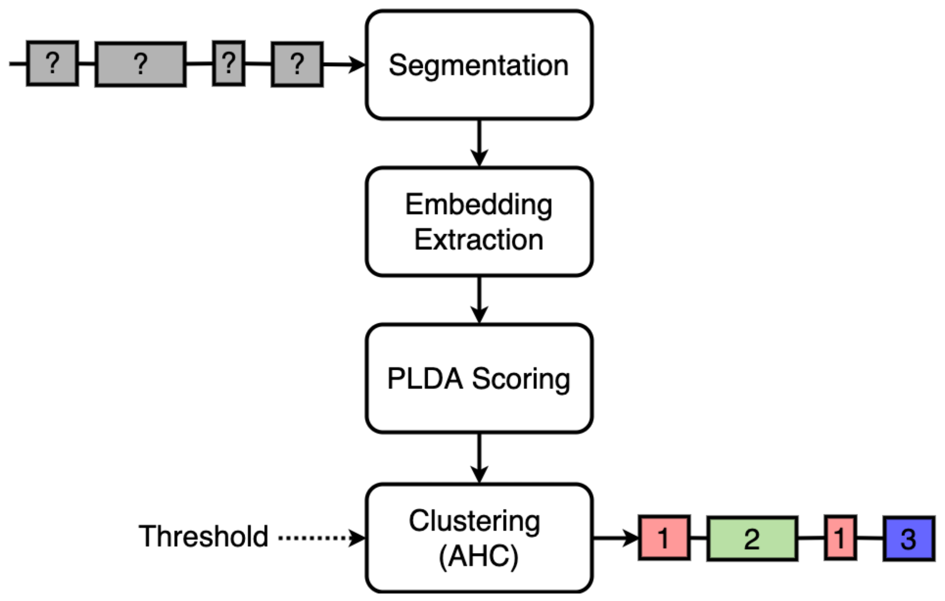


Figure 7.1: Diagram of a speaker diarization system. The labels 1, 2, and 3 denote speakers discovered during the diarization process.

following sections.

7.3.2 Speech Activity Detection

Diarization is typically performed on top of the output of a speech activity detection (SAD) system that identifies the speech and nonspeech regions. In this chapter, we will primarily make use of oracle SAD marks. This permits us to focus solely on speaker mislabeling errors. However, we include results using a Kaldi-based SAD system in Section 7.4.2. This system is a time-delay DNN trained on the Chime 6 training data to discriminate between speech and nonspeech classes.

7.3.3 Embedding Extraction

Once speech activity detection is performed, the speech segments are subsegmented into 1.5 second segments, which overlap with neighboring segments by 50%. In (Sell and Garcia-Romero, 2014) it was found that this denser sampling improves clustering. These segments are then projected into fixed-dimensional embeddings (in this chapter, either i-vectors or x-vectors).

7.3.4 PLDA Scoring

Next, as is commonly done in i-vector-based systems, the embeddings and the PLDA parameters are projected into a conversation-dependent space using principle component analysis (PCA). The conversation-dependent PCA adapts the scoring metric to the unique characteristics of the conversation. Then, we use this adapted PLDA model to compare the embeddings of all pairs of 1.5 second segments in the recording to produce a pair-wise similarity matrix.

7.3.5 Agglomerative Hierarchical Clustering

Given the similarity matrix, we perform agglomerative hierarchical clustering (AHC) with average linkage clustering (Maimon and Rokach, 2005). In AHC, we iteratively merge together clusters of speech segments, until some stopping

Table 7.1: X-vector DNN architecture, where N is the size of the embedding layer (between 128 and 512 depending on the experiment).

Layer	Layer Type	Context	Size
1	TDNN-ReLU	t-2:t+2	512
2	TDNN-ReLU	t-2, t, t+2	512
3	TDNN-ReLU	t-3, t, t+3	512
4	Dense-ReLU	t	512
5	Dense-ReLU	t	1500
6	Pooling (mean+stddev)	Full-seq	2x1500
7	Dense(Embedding)-ReLU		N
8	Dense-Softmax		# training spkrs

criterion is met. If we don't know the number of speakers in a recording, the stopping criterion is a threshold. If the similarity between the closest clusters is less than this threshold, we stop the clustering process. Typically, this threshold is tuned on some in-domain dataset; its removal is a topic of Section 8.3.1 in the next chapter.

7.3.6 X-vector System

The x-vector system is closely based on the system described in Chapter 5. However, here the DNN is trained on MFCCs, between 20 and 40 dimensions, depending on the application and audio sampling rate. The features are mean normalization over a 3 second window. The x-vector architecture is similar to the one used in previous work, in Chapter 4. The embedding layer ranges between 128 and 512 depending on the application.

7.3.7 I-vector Baseline

The i-vector system is also quite similar to those described in earlier chapters for speaker recognition. However, here we append only the first-order deltas to the feature inputs (rather than including the second order deltas) and the mean normalization is applied over a 3 second window. The i-vector system uses a UBM that ranges from 512 to 2048 and the i-vector subspace dimension ranges 100 to 400, depending on the experiment. These configurations are described in following experimental sections.

7.4 Experimental Results

7.4.1 CALLHOME Telephony

In this section, we report results on the CALLHOME dataset, which consists of 500 telephone recordings from 6 different languages: Arabic, English, German, Japanese, Mandarin, and Spanish. Each recording contains multiple speakers, with 2 to 7 speakers per recording. For tuning purposes, the CALLHOME dataset was split into two equally sized parts, CALLHOME1 and CALLHOME2. For example, CALLHOME1 is used for centering and whitening in the embedding space (for both types of embeddings), and for tuning the AHC stopping threshold. Then, diaization is performed on CALLHOME2. This

Table 7.2: DER(%) on the CALLHOME diarization dataset

	threshold	**oracle #speakers**
i-vector	10.36	8.69
x-vector	8.39	7.12

process is then repeated with the partitions reversed.

For experiments on the CALLHOME corpus, the x-vector and i-vector extractors are trained on audio from past NIST SREs prior to 2010, Switchboard Cellular 1 and 2, and Switchboard 2 phase 1–3. The x-vector DNN training data is further enlarged using data augmentation, as described in Section 5.5.3. As we saw in Section 5.6.3, data augmentation does not improve the quality of the i-vector system (since it is trained without supervision) and so we do not use data augmentation in this system. Both systems use 20 dimensional MFCCs. The x-vector DNN uses the architecture described in Table 7.1, with an embedding layer of size 128. The i-vector system uses a 2048-component UBM with a 128-dimensional i-vector subspace.

In Table 7.2 we compare the performance of the i-vector system and the x-vector system. In the column “threshold” we report the DER using the AHC threshold tuned on half of CALLHOME applied to the other half (and vice versa). Using this clustering method, we see that the x-vector system achieves a 19% reduction in diarization error rate. To get a better sense of the performance under optimal clustering, the column “oracle #speakers” assesses performance using the actual number of speakers per recording. Here we see that both systems improve given oracle SAD marks, and the x-vector system again achieves an 18% reduction diarization error rate.

7.4.2 CHIME 5

The CHIME 5 corpus consists of far-field audio collected from dinner parties using several microphone arrays. The dataset consists of a training, dev and eval portion. The partitions are equally sized, each containing audio from two parties recorded from five different microphone arrays. Each recording is about two and half hours long and contains speech from exactly four speakers.

The extractors (x-vector and i-vector) are trained on the VoxCeleb corpus. The x-vector DNN training list is further enlarged by artificially reverberating the training data, and combining it with the clean training list. Both systems use the CHIME 5 train set to train the PLDA models.

The x-vector DNN uses the architecture in Table 7.1, but here the x-vector dimension is 128. For the i-vector extractor, we use a 768-component UBM and a 100-dimensional i-vector extractor. The i-vector system is trained on 30 dimensional MFCCs, whereas the x-vector system is trained on 40 dimensional MFCCs (to match the features used by an ASR DNN). For both systems, we

Table 7.3: DER(%) on the CHIME 5 eval dataset

	oracle SAD	**Kaldi SAD**
i-vector	53.48	63.15
x-vector	32.58	40.48

perform beamforming on the microphones of each array using a simple delay-and-sum beamformer named Beamformit (Anguera, Wooters, and Hernando, 2007).

In the CHIME 5 dataset, nonspeech events provide a significant challenge. Therefore, we present results using both the oracle SAD and results using a TDNN SAD trained on the CHIME 5 train set using the Kaldi toolkit. In either case, we use the knowledge that there are exactly 4 speakers per recording in our clustering. In Table 7.3 we see that using the oracle SAD, the x-vector system provides a relative improvement of 39% in DER, compared to the i-vector baseline. Using the Kaldi SAD instead of the oracle SAD raises the DER for both systems, as the performance metric now includes errors due to missed and false alarm speech (see Section 7.2.3). Nonetheless, we see a reduction of 36% in DER of the x-vector system relative to the baseline. Therefore, we see that the substantial performance gains made by moving to a DNN embedding system are not lost after replacing the oracle SAD with a real SAD.

Table 7.4: DER(%) on the DIHARD 2018 eval dataset

	threshold	**oracle #speakers**
i-vector	28.51	24.42
x-vector	26.30	23.42

7.4.3 DIHARD 2018

The 2018 DIHARD challenge (Ryant et al., 2018) was intended to provide a standard dataset, drawn from diverse and challenging conditions, to evaluate diarization system performance. The development (dev) data included audio from ten diverse domains ranging from monologues to interviews with children to meetings and internet videos. An additional three sources not present in the training data were included in the evaluation (eval) data as well. This resulted in a highly diverse and challenging dataset for diarization.

As in Section 7.4.2, we use VoxCeleb to train both embedding extractors (i-vector or x-vector). For the x-vector DNN, we also add additional training data with augmentations, using the recipe described in Section 5.

The x-vector DNN follows the same architecture as Table 7.1, but uses 512-dimensional x-vectors. The i-vector system uses a 2048-dimensional UBM and 400-dimensional i-vectors.

In Table 7.4 we report DER results on the DIHARD 2018 eval dataset. In accordance with the DIHARD challenge, the DER error metric (see Section 7.2.3) includes errors around segment collars and due to overlapping speech. We see that unlike Sections 7.4.1 and 7.4.2, the performance of the i-vector and x-vector systems is quite similar. Nonetheless, the x-vector results using the tuned threshold are still about 8% better, and the results using the oracle number of speakers is about 4% better. Perhaps one reason for the more similar results here is that a significant portion of the error comes from overlapping speech, and neither system is capable of assigning a speech segment to multiple speakers.

7.5 Conclusion

In this work, we have presented a diarization system that use discriminateively trained embeddings (x-vectors) to replace i-vectors for diarization. We reported results on three diarization datasets, and found that x-vectors outperform a competitive i-vector baseline on all three datasets. In Chapter 4 we found that x-vectors tend to achieve their largest improvement over i-vectors on short speech segments. Since the standard diarization framework is based on clustering embeddings extracted from short speech segments, it is not surprising that they would perform well for this task. In Section 7.4.3 we presented results on the DIHARD 2018 evaluation dataset, where we included errors due to overlapping speech. An important limitation of this diarization framework is that it assumes that only one speaker can be speaking at a time. In Section 7.4.2, we reported results on the CHIME 5 dataset. Although the x-vector system achieved significantly better results than the i-vector system, we observe that this far-field domain is extremely challenging, DER remains unsatisfactorily high, and that existing data augmentation techniques are not sufficient. Effectively addressing overlapping speakers and improving the robustness of diarization to unconstrained far-field audio would have a significant impact on this field.

Chapter 8

Speaker Recognition with Diarization

Our prior work introduced x-vectors, an embedding that is very effective for both speaker recognition and diarization. These embeddings are now replacing i-vectors, which have been the state-of-the-art in both tasks for almost ten years. This chapter combines this previous work, described in Chapters 5 and 7, and applies it to the problem of speaker recognition in multi-speaker conversations. We measure performance on Speakers in the Wild and report what we believe are the best published error rates on this dataset. Moreover, we find that diarization substantially reduces speaker recognition error rates when there are multiple speakers, while maintaining excellent performance on single-speaker recordings. Finally, we introduce an easily implemented method to mitigate the need for the domain-sensitive threshold typically used in the clustering stage of a diarization system. The proposed method is more robust to domain shifts, and achieves similar speaker recognition results to those obtained using a diarization system with a well-tuned threshold.

8.1 Introduction

Most speaker recognition research assumes that there is only one speaker per recording. In particular, all models for capturing speaker characteristics in fixed-dimensional embeddings assume that the input speech was generated from a single speaker, and violating this assumption reduces the effectiveness of the representation (Kenny, Reynolds, and Castaldo, 2010; Martin and Przybocki, 2001). Research on the topic of speaker recognition in multi-speaker conversations has increased with the release of the 2016 Speakers in the Wild (SITW) challenge (McLaren et al., 2016) and the recent NIST 2018 Speaker Recognition Evaluation (*NIST Speaker Recognition Evaluation 2018* 2018) due to the presence of multi-speaker enrollment and test recordings. This encourages diarization to be performed in conjunction with speaker recognition. Participants in the SITW challenge showed that diarization can significantly improve speaker recognition rates (Novotnỳ et al., 2016; Liu et al., 2016). Our study underscores the value of diarization for speaker recognition in the multi-speaker environment.

8.2 X-vector DNN

This section describes the x-vector DNN. The architecture is based on the DNN embedding system described in Chapter 5, but uses a larger network architecture and additional training data augmentations, to improve performance.

8.2.1 Architecture

Table 8.1 summarizes the architecture used in this work. The first 10 layers of the x-vector DNN consists of layers that operate on speech frames, with a small temporal context centered around the current frame t. The pooling layer receives the output of layer 10 as input, aggregates over the input segment, and computes its mean and standard deviation. These segment-level statistics are concatenated together and passed through the remaining layers of the network. The output layer computes posterior probabilities for the training speakers. Compared to the architecture described in Chapter 5, we use a

Table 8.1: X-vector DNN architecture

Layer	Layer Type	Context	Size
1	TDNN-ReLU	t-2:t+2	512
2	Dense-ReLU	t	512
3	TDNN-ReLU	t-2, t, t+2	512
4	Dense-ReLU	t	512
5	TDNN-ReLU	t-3, t, t+3	512
6	Dense-ReLU	t	512
7	TDNN-ReLU	t-4, t, t+4	512
8	Dense-ReLU	t	512
9	Dense-ReLU	t	512
10	Dense-ReLU	t	1500
11	Pooling (mean+stddev)	Full-seq	2x1500
12	Dense(Embedding)-ReLU		512
13	Dense-ReLU		512
14	Dense-Softmax		7185 (# spkrs)

slightly wider temporal context in the TDNN layers, and interleave dense layers between the TDNN layers. We found that this architecture significantly outperforms our prior work in Chapter 5.

8.2.2 Features

The features are 30 dimensional MFCCs with a sampling window of 25 ms, mean-normalized over a sliding window of up to 3 seconds. Audio files are sampled at 16 kHz. The Kaldi energy SAD is used to filter out nonspeech frames.

8.2.3 Training

The DNN is trained to classify the 7,185 speakers in the training data using a multi-class cross entropy objective function. A training example consists of a 2–4 second speech segment (about 3 seconds average), along with the corresponding speaker label. Following a study by McLaren et al. in (McLaren et al., 2018), we use much more aggressive data augmentation than in Chapter 5 (see Section 8.5.1), train the DNN for 6 epochs (instead of 3) and use a minibatch size of 128 (instead of 64).

8.2.4 Embedding Extraction

Once the network is trained, x-vectors are extracted from the affine component of layer 12 (see Table 8.1). The x-vectors are used as features for two different PLDA backends (one for the diarization system described in Section 8.3 and one for the speaker recognition system described in Section 8.4).

8.3 Speaker Diarization

The diarization system is based on the system described in Chapter 7. The general problem of diarization is described in more detail there. The system uses x-vectors extracted from the DNN in Section 8.2 with PLDA, and agglomerative hierarchical clustering (AHC). The PLDA backend consists of centering, whitening and length normalization, followed by scoring. All components of the backend are trained on 3 second segments extracted from the augmented VoxCeleb data described in Section 8.5.1.

For either an enrollment recording or a test recording, x-vectors are extracted from 1.5 second segments with a 0.75 second overlap. PLDA scores are computed between all pairs of x-vectors. This is followed by AHC with average linkage clustering. In our primary system, the number of clusters is controlled by a stopping threshold which was tuned on the held-out SITW *DEV* set. The most similar clusters are iteratively merged, until the average PLDA similarity score between clusters is less than a threshold. Diarization results in some indeterminant number of clusters, N, which, ideally should correspond to the true number of speakers.

8.3.1 Removing the AHC Threshold

AHC-based diarization typically requires a well-chosen cluster stopping threshold to achieve good performance. This threshold is often dataset dependent, and a poorly chosen threshold will result in bad diarization performance. This is particularly a significant concern when a reliable development/tuning set is not available.

To improve robustness in the speaker recognition task, we propose a simple alternative that eliminates the need for the AHC threshold. Instead of relying on a tuned AHC threshold, we begin with an estimate of the maximum number of speakers K that might appear in the recordings. We assume that there are never more than K speakers in an utterance, and perform clustering K times, with exactly $k \in \{1, 2, \ldots, K\}$ clusters. Taking the union of each of the individual diarizations results in a set of $N = \frac{K(K+1)}{2}$ ways to partition a recording that has at most K speakers. The N putative single-speaker segments are then treated exactly the same as the speakers discovered by clustering with an AHC threshold, as described in Section 8.4. Note that although the putative single-speaker test segments from a single diarization output are disjoint (nonoverlapping), there is the possibility of significant overlap among the pairs of N segments we consider. For example, if the goal is to determine whether a given enrolled speaker is present in a multi-speaker test recording, then the k^{th} diarization output suggests k different "single-speaker" test segments. The K diarization outputs together therefore suggest $1 + 2 + \cdots + K = \frac{K(K+1)}{2}$ different test segments. This strategy works, because if the enrolled speaker is present in the test recording, then at least one of the N segments is likely to test positive, whereas if the speaker is not present, then none should.

Looking at the SITW *DEV* set, we found that the performance isn't very sensitive to different values of $K \geq 3$. We use $K = 5$ for the experiments in the results section.

8.3.2 Diarizing Enrollment Recordings

When processing a multi-speaker enrollment recording it is generally impossible to know which of the speakers we wish to enroll without additional information about the enrollment speaker (such as existing speech produced by the speaker). However, this task becomes feasible if we are able to identify a segment from the recording which is known to contain the speaker we wish to enroll. In practice, such segments may be obtained by manual annotation. In SITW, these are known as "assist" segments, and typically contain about 5 seconds of speech. Although the assist segment could be used directly to enroll the speaker, it typically does not provide enough speech to obtain a high quality representation of the speaker. To improve the quality of the representations, we use the assist segment to identify any other speech in the recording which belongs to the speaker we wish to enroll, while also removing any speech belonging to other speakers.

The speech corresponding to the assist segment is treated as an "auxiliary enrollment" and the entire enrollment recording is treated as an "auxiliary test” recording. After segmenting the multi-speaker enrollment recording into 1.5 second chunks and clustering them, we obtain N single-speaker segments in the auxiliary test recording. We then perform the procedure described in Section 8.4, which involves computing PLDA scores between the auxiliary enrollment (the assist segment) and each of the N speakers discovered automatically in the auxiliary test. All the speech segments belonging to the speaker in the auxiliary test that maximizes the PLDA score (as in Equation 8.1) are identified, and used by the speaker recognition system to extract a single enrollment x-vector.

8.3.3 Diarizing Test Recordings

Handling the test recordings is straightforward once AHC is performed. The speech segments are grouped according to the N hypothesized speakers discovered in the conversation, and are passed directly to the speaker recognition system, where they are then used to perform recognition as described in the next section.

8.4 Speaker Recognition

Recognition is performed using x-vectors extracted from the DNN in Section 8.2 and a PLDA backend. The x-vectors are centered, projected from 512 to 225 dimension via LDA, and are length-normalized. All parameters in the backend are estimated on the augmented VoxCeleb data, as described in Section 8.5.1.

If diarization was performed on a test recording (see Section 8.3.3), then, instead of extracting a single x-vector for the entire test recording, we extract N x-vectors, one for each of the N speakers identified in the recording. Suppose $R(\mathbf{u}, \mathbf{v}_i)$ is the PLDA log-likelihood ratio score between the x-vector $\mathbf{u}$ for the enrolled speaker and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ are the x-vectors for each of the N speakers in the test recording. To perform speaker recognition, we compute the PLDA score as in Equation 8.1, which is the maximum of the PLDA scores between the enrollment x-vector and all N test x-vectors.

$$R(\text{enroll}, \text{test}) = \max\{R(\mathbf{u}, \mathbf{v}_1), \ldots, R(\mathbf{u}, \mathbf{v}_N)\} \tag{8.1}$$

Handling a diarized enrollment recording is simpler, since there can only be one speaker of interest at a time. We simply extract the enrollment x-vector from all speech frames identified as belonging to the speaker of interest (as described in Section 8.3.2), and ignore the remaining frames.

8.5 Experimental Setup

Next, we describe the datasets used for training and evaluating the methods presented earlier in the chapter.

8.5.1 Training Data

The speaker recognition system is trained on a large subset of the combined VoxCeleb 1 (Nagrani, Chung, and Zisserman, 2017) and VoxCeleb 2 (Chung, Nagrani, and Zisserman, 2018) corpora sampled at 16 kHz. The test portion of VoxCeleb 2 as well as 60 speakers from VoxCeleb 1 overlap with the evaluation dataset, and so we removed them before training [1]. This leaves a total of over 150,000 recordings from 7,185 speakers. Using the target speaker marks provided in the corpora, the recordings are split into over 1.2 million single-speaker segments.

We apply a data augmentation strategy based on Chapter 5 that consists of adding noises, music, babble, and reverberation. The x-vector DNN was trained on 7.2 million segments, comprised of the 1.2 million "raw" segments

[1] See `http://www.openslr.org/resources/49/voxceleb1_sitw_overlap.txt` for a list of speakers from VoxCeleb 1 which are known to overlap with SITW.

extracted directly from VoxCeleb, plus an additional 6 million segments obtained by data augmentation.

The PLDA backend for speaker recognition (Section 8.4) was trained on the full-length recordings of VoxCeleb, but we only keep the speech belonging to the speakers of interest (as provided by the segment boundaries specified in the distributed corpora). For the backend, we apply augmentation to double the amount of training data, which increases the number of recordings from about 150,000 to 300,000. Finally, the diarization backend (Section 8.3) was trained on 256,000 three second segments extracted randomly from the full-length augmented recordings.

8.5.2 Speakers in the Wild

We perform experiments on the Speakers in the Wild (SITW) dataset developed by SRI International (McLaren et al., 2016). The dataset consists of challenging audio collected from diverse conditions in the video audio domain. One of the challenges is the presence of multiple speakers in some of the recordings. The recordings vary in length, from 6 to 240 seconds.

The dataset is divided into a development set *DEV* (which we use only for tuning) and an evaluation set *EVAL*. The *EVAL* set contains 180 speakers divided into 4,170 models and a total of 2,883 audio files.

Enrollment conditions

- *CORE:* Enrollment recordings, with duration between 6 and 240 seconds, that each contain exactly one speaker
- *ASSIST:* One or more speakers are present in the enrollment recording, along with an "assist" mark, which is a short segment (typically 5 seconds) of the recording that is known to contain only the speaker of interest

Test conditions

- *CORE:* Test recordings of duration between 6 and 240 seconds that contain exactly one speaker
- *MULTI:* One or more speakers may be present in the test recordings

A trial is considered a target trial if the enrollment speaker of interest is *any* of the potentially multiple speakers in the test recording.

8.6 Experimental Results

In Table 8.2 we report results on the *EVAL* portion of the Speakers in the Wild (SITW) dataset. The four evaluation conditions are formed by pairing an enrollment condition with a test condition described in Section 8.5.2. Performance on these conditions is examined in Sections 8.6.1–8.6.4. The results

Table 8.2: Results on the SITW evaluation set

		CORE-CORE			CORE-MULTI			ASSIST-CORE			ASSIST-MULTI		
	Diarize	**EER**	**DCF1**	**DCF2**	**EER**	**DCF1**	**DCF2**	**EER**	**DCF1**	**DCF2**	**EER**	**DCF1**	**DCF2**
	NO DIAR	1.7	0.20	0.34	3.5	0.28	0.44	3.2	0.24	0.38	4.3	0.28	0.43
Threshold	ENROLL	1.8	0.21	0.35	2.1	0.22	0.41	1.6	0.20	0.35	3.0	0.26	0.41
	TEST							3.3	0.24	0.39	3.8	0.26	0.41
	BOTH							1.7	0.21	0.36	2.1	0.21	0.37
No threshold	ENROLL	1.8	0.23	0.36	2.0	0.22	0.40	1.6	0.20	0.36	3.0	0.26	0.42
	TEST							3.8	0.26	0.40	3.9	0.26	0.41
	BOTH							2.2	0.23	0.38	2.2	0.22	0.38

are further broken down by whether or not the enroll or test recordings are diarized. The diarization system and its interaction with speaker recognition is the subject of Sections 8.3–8.4. We report results in terms of equal error rate (EER) and the minimum of the normalized detection cost function (DCF). DCF1 uses $P_{\text{Target}}=10^{-2}$ and DCF2 uses $P_{\text{Target}}=10^{-3}$. These metrics were described in greater detail in Section 2.3.

The *Threshold* system uses an AHC threshold tuned on the *DEV* set to control the number of speakers, whereas *No threshold* uses the alternative method described in Section 8.3.1 to eliminate the need for a threshold. In Section 8.6.5, we discuss performance using the proposed alternative system that eliminates the AHC threshold.

8.6.1 CORE-CORE

In the simplest SITW evaluation condition, there is exactly one speaker present in both the test and enrollment recordings. In the first row of results in Table 8.2 (NO DIAR), we do not apply any diarization and achieve very low error rates (1.7% EER). In the next row of results (TEST), we apply diarization to the test recordings. Using the standard approach, diarizing single-speaker recordings degrades performance as expected, but by a very small amount–less than half a percent relative on all performance metrics. The degradation is expected because diarization can only cause a reduction in the amount of test speech.

CORE-CORE is the most commonly used condition from SITW. Our best performance on this condition is EER=1.7% DCF1=0.20, which comfortably outperforms what were the previously best reported numbers by Silnova et al. (2018), which are EER=2.7% and DCF1=0.33. The x-vector DNN architecture used in this chapter is similar to that of Silnova et al. (2018), so the improvements reported here are mostly due to a better training recipe, which consists of more aggressive data augmentation than previously used, and the addition of a substantial amount of in-domain data from the VoxCeleb 2 Corpus (Chung, Nagrani, and Zisserman, 2018).

The study by Silnova et al. (2018), however, only reported performance on the CORE-CORE condition, which does not require diarization. We therefore cannot compare their results with the other test conditions we investigate next.

8.6.2 CORE-MULTI

CORE-MULTI extends the previous condition with test recordings that contain one or more speakers. We still use single-speaker enrollment recordings in this condition.

Diarizing the multi-speaker test conversations (TEST) results in a clear improvement over performing no diarization (NO DIAR), with an EER reduction from 3.5% to 2.1%. Using a tuned AHC threshold, diarization reduces EER by 38%, and by 20% in DCF1 and 8% in DCF2. The results that eliminate the AHC threshold are even slightly better. Note that we do not consider the effect of diarizing the enrollment recordings yet, as we do not consider that meaningful unless the assist segments are provided.

8.6.3 ASSIST-CORE

This condition introduces our systems to the assist segments. These segments provide a few seconds of speech of the speaker we wish to enroll. As described in Section 8.3.2, we use the assist marks to discover additional speech (in the enrollment recording) that belongs to the speaker of interest, while discarding any speech from other speakers. Although the enrollment recordings may have multiple speakers, the test recordings are single-speaker in this condition.

Diarizing the enrollment recordings (ENROLL) reduces EER by 50% relative to NO DIAR (from 3.2% to 1.6% in EER). The DCF numbers also improve, but by a smaller amount. As expected, unnecessarily diarizing the test recordings (but not enrollment) results in the worst performance. Nonetheless, the *Threshold* results are only slightly worse than the results without diarization. This indicates that the diarization system is not significantly reducing the amount of test speech in these single-speaker recordings. In the last row (BOTH), we diarize both the enrollment and the test recordings. For the *Threshold* system, this degrades performance by 2–8% relative to the ENROLL results, but still maintains an improvement over NO DIAR.

8.6.4 ASSIST-MULTI

This condition combines the challenge of potential multi-speaker enrollment recordings with multi-speaker test recordings. As in the previous section, diarizing the enrollment recordings is enabled by the assist segments.

Diarizing either enroll or test recordings individually (but not together) results in moderate improvements in EER, and smaller improvements in DCF1 and DCF2. Although both have multiple speakers, we observe a larger improvement by diarizing just the enrollment recordings as opposed to diarizing just the test recordings. Perhaps the enrollment recordings tend to contain more speech from the speaker of interest, and as a result, diarization adds a substantial amount of additional speech beyond what is contained in the initial 5 second assist segment. This would be consistent with our observations in previous chapters (see Table 3.2), that found that performance rapidly improves as we increase the amount of speech used for embedding extraction. Fortunately, the benefit of combining enroll and test diarization results in more dramatic improvements. Looking at the *Threshold* system, we observe a 50% EER reduction (from 4.3% to 2.1%) over no diarization and a 14–23% reduction in DCF.

8.6.5 Removing the Threshold

The previous sections showed that the *Threshold* system achieves excellent results. It relies on an AHC threshold tuned on labeled in-domain data. Although this is not an obstacle for the work in this chapter, as we are able to tune on the well-matched *DEV* set, it cannot be assumed that an in-domain development set is always available. The *No threshold* system uses the method described in Section 8.3.1 to address the problem of performing diarizing when no development set is available to tune on.

In Table 8.2 we see that, under most conditions, the alternative *No threshold* system performs similarly to *Threshold*. When diarizing is required for multi-speaker conversations, the results of this system are very similar to the standard approach. The system performs worst on ASSIST-CORE when we needlessly diarize the test recordings. However, the BOTH results are nonetheless better than the results without diarization.

8.7 Conclusions

This chapter investigated speaker recognition with multi-speaker recordings. We used a diarization system based on x-vectors, PLDA, and agglomerative hierarchical clustering (AHC) as a front-end for a speaker recognition system. We evaluated performance on the Speakers in the Wild dataset, and found that diarization significantly improved speaker recognition performance on multi-speaker conversations, and retained strong performance on single-speaker recordings as well. Finally, we showed that the AHC threshold, which controls the number of clusters, can be replaced with an alternative method that achieves similar performance under most conditions, but eliminates the need for a in-domain development set for tuning the diarization system.

Chapter 9

Conclusion

In this dissertation, we developed a DNN architecture that maps variable-length speech recordings to fixed-dimensional embeddings that we call x-vectors. These embeddings achieve state-of-the-art performance in several areas of speech technology, including speaker recognition, spoken language recognition, and speaker diarization. In the following sections, we will summarize the contributions and findings of the dissertation, and conclude with a brief discussion of several open problems.

9.1 Contributions

9.1.1 Speaker Recognition

In Chapter 3 we developed a end-to-end speaker verification system on a large proprietary dataset. The end-to-end approach jointly learns embeddings and a similarity metric to compare pairs of embeddings. This study provided early evidence that DNNs trained to capture speaker characteristics could be competitive with the best of the previously proposed systems. We showed, however, that the end-to-end approach required a very large amount of data to be effective, which is not always readily available.

In Chapter 4 we proposed splitting the end-to-end approach into two parts: a DNN to extract embeddings, which we call x-vectors, and a separately trained backend to compare them. We found that the x-vector system greatly improved over the end-to-end approach when trained and tested on publicly available datasets, and rivaled the performance of a traditional i-vector system. In Chapter 5 we greatly improved on this approach using data augmentation, and showed that the proposed system significantly outperformed the i-vector systems, to become the current state-of-the-art in speaker recognition. An additional contribution was the release of several x-vector-based recipes in the open source Kaldi toolkt, which has enabled scores of other researchers in academia and in industry to develop new techniques and to field novel applications.

9.1.2 Spoken Language Recognition

Mirroring our progress in speaker recognition, we adapted x-vectors to the task of spoken language recognition in Chapter 6. In this work, the x-vector DNN was trained with language labels instead of speaker labels. We contributed a study of several variations of the x-vector framework, and examined a variety of feature types, classifiers, and the augmentation strategy. We found that the best performing system uses multilingual bottleneck features from an ASR DNN, in addition to the usual data augmentation techniques and a discriminative Gaussian classifier. We believe that this is because languages differ in their phonetic inventories and that speech features optimal for speech recognition are therefore more effective in language recognition than, say, MFCC features. This system outperformed several state-of-the-art i-vector baselines, and achieved excellent results on the NIST 2017 Language Recognition Evaluation.

9.1.3 Speaker Diarization

In Chapter 7 we studied a diarization framework based on clustering fixed-dimensional embeddings extracted from short segments of speech using PLDA as a similarity metric, and agglomerative hierarchical clustering. Until recently, the embeddings used in state-of-the-art systems were i-vectors. In this chapter, we introduced a diarization system that replaced i-vectors with x-vectors. We found that x-vectors achieve superior performance for this task.

In general, diarization is not a stand-a-lone task, but rather a preprocessing step for down stream tasks, such as ASR or speaker recognition. In Chapter 8 we performed a study of diarization for speaker recognition on recordings with multiple speakers using our state-of-the-art x-vector system. We showed that our proposed diarization system significantly improves speaker recognition performance in the presence of multiple speakers while retaining excellent performance on single-speaker recordings as well. In addition, we showed that the domain-sensitive stopping threshold, commonly used in the diarization component, can be eliminated for the purpose of speaker recognition without significant degradation to speaker recognition performance.

9.1.4 Data Augmentation in Speech Processing

Augmentation is an inexpensive method to increase the amount and diversity of the existing training data. In Chapter 5 we performed an extensive study on data augmentation in speaker recognition. We studied the effect of data augmentation on several components of i-vector and x-vector systems. We found that although augmentation is beneficial in the PLDA classifier of both systems, it is not helpful in the i-vector extractor. However, the x-vector DNN effectively exploits data augmentation, due to its supervised training. The results of Chapter 6 echo these findings, and show that data augmentation significantly improves an x-vector system for spoken language recognition.

As part of this dissertation, we developed and released a music, speech, and noise corpus, called MUSAN. MUSAN, described in Appendix A, has been used for augmenting x-vector training datasets used in Chapters 5– 8. MUSAN is also used widely in other areas of speech processing not studied in this dissertation, including in ASR acoustic model training.

9.2 Extensions

In this section, we will briefly describe several extensions to the problems and solutions presented in this dissertation.

9.2.1 Alternative X-vector Architectures

A key attribute of the x-vector system is the temporal pooling mechanism, which is responsible for mapping speech frames to fixed-dimensional embeddings. In the work presented here, pooling consists of a mean and a standard deviation vector (see Section 3.3). However, one potential limitation of this approach is that all frames are treated as equally important in the statistics computation. However, some frames contain more speaker discriminative information than others. For example, it is easy to see that if a SAD system mislabels a *nonspeech* frame as *speech* we will inevitably include some input features to the x-vector DNN that contain no speaker discriminative information. To address this concern, an attention component was introduced in (Okabe, Koshinaka, and Shinoda, 2018; Cai, Chen, and Li, 2018), to assign weights to each frame of the input. The weights are in turn used in the computation of the statistics (mean and variance) in the pooling layer. Weighted statistics are now commonly used in recent work, and sometimes achieve better results than the standard statistics pooling layer.

9.2.2 X-vector Training Improvements

Most implementations of the x-vector paradigm use short speech segments (e.g., a few seconds) to train the DNN. This introduces a mismatch between training and inference, where we typically need to extract emeddings from longer duration recordings (e.g., a minute or more). To address this, Garcia-Romero et al. (2019) presented a DNN refinement approach that updates a subset of the DNN parameters with full recordings to reduce this mismatch. The engineering compromise of using 2 to 4 second segments to train the embedding can be relaxed by freezing the DNN components that are responsible for most of the memory, as well as controlling the network capacity so that it does not overfit to longer sequences. In this way, we can use a network pretrained on short segments as an initialization for a refinement step that then fine-tunes a subset of the network parameters on the full recordings. Also, the work in (Garcia-Romero et al., 2019) seeks to simplify the architecture by removing the need of PLDA for scoring, and replacing it with the simpler cosine similarity metric. To that end, the authors replaced the standard softmax output layer described here, with the angular softmax.

9.3 Future Work

In this section, we will explore several limitations of the work in this dissertation and how they may be addressed in the future.

9.3.1 Probabilistic Embeddings

Although x-vectors achieve excellent results, generative models, such as i-vectors continue to have some appealing properties that are absent in our proposed approach. In particular, notions of *uncertainty* are easily extracted from a generative model. Capturing uncertainty may be useful for domains where the quality of embedding varies significantly, such as in speaker diarization, where embeddings are extracted from very short segments. According to Brummer et al. (2018), an x-vector can be interpreted as a point estimate for a hidden speaker factor, but it does not allow for the extraction of uncertainty. As a solution, Brümmer proposed a framework for probabilistic embeddings, in which the output of the DNN is the likelihood function of a hidden speaker factor, rather than its point estimate. As proposed in (Brummer et al., 2018), the probabilistic embedding framework can be combined with the x-vector framework proposed here, and trained in an end-to-end manner.

9.3.2 ASR Acoustic Model Adaptation

In this dissertation, we found that x-vectors have become the state-of-the-art representation for several areas of speech technology that have historically relied on i-vectors: speaker recognition, diarization, and spoken language recognition. ASR, however, is a notable exception to this trend. In ASR, it is often beneficial to append i-vectors to the input features, which helps the acoustic model adapt to the characteristics of the speaker and recording environment. Unfortunately, in particularly challenging acoustic conditions (e.g., with significant noise or reverberation), i-vectors do not yield significant accuracy improvements, and therefore it is of great interest to replace i-vectors with a more robust alternative. Despite their proven ability to capture speaker characteristics, x-vectors have had limited success for this purpose. However, there has recently been some progress in a related direction. Khokhlov et al. (2019) proposed modifying the x-vector framework to capture information about the room impulse response (rather than speakers or languages), and the embeddings are then used like i-vectors for ASR acoustic model adaptation.

9.3.3 Overlapping Speakers

Speaker diarization is a multi-faceted and extremely difficult problem in speech processing. Although good results can be achieved on clean, summed channel telephone speech, there are many domains where the performance is not satisfactory. Recently, the DIHARD competition highlighted challenges due to diverse recording conditions, children's speech, and overlapping speakers. Most diarization systems make no attempt to handle overlapping speech. In fact, the system described in Chapter 7 makes the assumption that one and only one speaker may speak at a given time. However, having a reliable way to detect and handle overlap has the potential to improve overall diarization accuracy and will likely benefit technologies such as ASR or speech separation, that are applied on top of the diarization results. A solution may involve incorporating a stand-alone speech separation system, or perhaps rethinking the current diarization approach entirely, and replacing it with a more comprehensive approach that combines speech separation and diarization together.

Appendix A

MUSAN: A Music, Speech, and Noise Corpus

A.1 Introduction

The MUSAN corpus was released under a flexible Creative Commons license (Snyder, Chen, and Povey, 2015). It consists of music from several genres, speech from twelve languages, and a wide assortment of man-made and natural noises. Although originally designed for training models for speech activity detection, this dataset has proven to be more valuable for data augmentation purposes, as demonstrated in Chapters 5 and 6.

Most publicly available corpora for music and speech discrimination that provide raw audio do not address the copyright of the data sources nor appear to have permission to redistribute the data. For instance, the GTZAN Music/Speech dataset (Tzanetakis and Cook, 2002) is widely used, but it appears that permission was not given by the copyright holders to redistribute the work. Other corpra, such as the Million Song database (Bertin-Mahieux et al., 2011) circumvent intellectual property issues by providing only features. However, this limits the user to building systems based only on the provided feature type. In contrast, our corpus is compiled from Creative Commons and US Public Domain sources, so we are free to redistribute the original audio.

The corpus consists of approximately 109 hours of audio that is in the US Public Domain or under a Creative Commons license. The dataset is partitioned into speech, music, and noise as described in the following sections.

A.1.1 Speech

This portion of the corpus consists of about 60 hours of speech. It contains 20 hours and 21 minutes of read speech from Librivox, all of which are in the Public Domain. Each WAV file is an entire chapter of a book, read by one speaker. Approximately half of the Librivox recordings are in English, and the remainder are from eleven other languages. Annotations for speaker and language are provided. The rest of the speech portion consists of 40 hours and

1 minute of US government (federal and various states) hearings, committees and debates that are generally understood to be in the US Public Domain. These files have been obtained from the Internet Archive and the Missouri Channel senate archives. These recordings are entirely in English.

A.1.2 Music

The music portion of the corpus has been downloaded from Jamendo, Free Music Archive, Incompetech, and HD Classical Music. It is 42 hours and 31 minutes. The music is divided into Western art music (e.g., Baroque, Romantic, and Classical) and popular genres (e.g., jazz, bluegrass, hiphop, etc). Annotations for genre, artist, and the presence or absence of vocals are provided. For Western art music, the composer is also provided. These files have all been released under some form of the Creative Commons license. Care was taken to ensure that any of these files can be used for commercial purposes.

A.1.3 Noise

This portion of the corpus contains 929 files of assorted noises, with a total duration of about 6 hours. These range from man-made noises, such as DTMF tones, dialtones, fax machine noises, and more, as well as ambient sounds, such as car idling, thunder, wind, footsteps, paper rustling, rain, animal noises, etc. We do not include recordings with intelligible speech. However, some recordings are crowd noises with indistinct voices. The files were downloaded from Free Sound and Sound Bible. The Free Sound part of the corpus is Public Domain material and the Sound Bible part is CC licensed.

References

Anguera, Xavier, Chuck Wooters, and Javier Hernando (2007). “Acoustic beamforming for speaker diarization of meetings”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.7, pp. 2011–2022.

Bertin-Mahieux, Thierry, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere (2011). “The Million Song Dataset”. In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Bromley, Jane, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah (1994). “Signature verification using a" siamese" time delay neural network”. In: *Advances in neural information processing systems*, pp. 737–744.

Brummer, Niko, Anna Silnova, Lukas Burget, and Themos Stafylakis (2018). “Gaussian meta-embeddings for efficient scoring of a heavy-tailed PLDA model”. In: *arXiv preprint arXiv:1802.09777*.

Burget, Lukáš, Oldřich Plchot, Sandro Cumani, Ondřej Glembek, Pavel Matějka, and Niko Brümmer (2011). “Discriminatively Trained Probabilistic Linear

Discriminant Analysis for Speaker Verification". In: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Prague, Czech Republic.

Cai, Weicheng, Jinkun Chen, and Ming Li (2018). "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System". In: *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pp. 74–81.

Chung, Joon Son, Arsha Nagrani, and Andrew Zisserman (2018). "VoxCeleb2: Deep Speaker Recognition". In: *INTERSPEECH*.

Cumani, Sandro, Pier Domenico Batzu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis (2011). "Comparison of speaker recognition approaches for real applications". In: *Interspeech*. ISCA.

Dehak, Najim, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet (2011). "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, pp. 788–798.

Fér, Radek, Pavel Matějka, František Grézl, Oldřich Plchot, and Jan Černockỳ (2015). "Multilingual bottleneck features for language recognition". In: *Sixteenth Annual Conference of the International Speech Communication Association*.

Garcia-Romero, Daniel and Carol Y Espy-Wilson (2011). “Analysis of i-vector Length Normalization in Speaker Recognition Systems.” In: *Interspeech*, pp. 249–252.

Garcia-Romero, Daniel and Alan McCree (2016). “Stacked Long-Term TDNN for Spoken Language Recognition.” In: *INTERSPEECH*, pp. 3226–3230.

Garcia-Romero, Daniel, Xinhui Zhou, and Carol Y Espy-Wilson (2012). “Multi-condition training of Gaussian PLDA models in i-vector space for noise and reverberation robust speaker recognition”. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4257–4260.

Garcia-Romero, Daniel, Xiaohui Zhang, Alan McCree, and Daniel Povey (2014). “Improving speaker recognition performance in the domain adaptation challenge using deep neural networks”. In: *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pp. 378–383.

Garcia-Romero, Daniel, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree (2017a). “Speaker diarization using deep neural network embeddings”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4930–4934.

Garcia-Romero, Daniel, David Snyder, Gregory Sell, Daniel Povey, and Alan McCree (2017b). “Speaker diarization using deep neural network embeddings”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4930–4934.

Garcia-Romero, Daniel, David Snyder, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur (2019). “X-vector DNN Refinement with Full-length Recordings for Speaker Recognition”. In: *Proc. Interspeech*, pp. 1493–1496.

Ghahremani, Pegah, Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur (2016). “Acoustic Modelling from the Signal Domain Using CNNs.” In: *Interspeech*, pp. 3434–3438.

Ghalehjegh, Sina Hamidi and Richard C Rose (2015). “Deep bottleneck features for i-vector based text-independent speaker verification”. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pp. 555–560.

Glembek, Ondřej, Lukáš Burget, Niko Brümmer, Oldřich Plchot, and Pavel Matějka (2011). “Discriminatively Trained i-vector Extractor for Speaker Verification”. In: *Interspeech*.

Gonzalez-Dominguez, Javier, Ignacio Lopez-Moreno, Haşim Sak, Joaquin Gonzalez-Rodriguez, and Pedro J Moreno (2014). “Automatic language identification using long short-term memory recurrent neural networks". In: *Fifteenth Annual Conference of the International Speech Communication Association*.

Gonzalez-Dominguez, Javier, Ignacio Lopez-Moreno, Pedro J Moreno, and Joaquin Gonzalez-Rodriguez (2015). "Frame-by-frame language identification in short utterances using deep neural networks". In: *Neural Networks* 64, pp. 49–58.

Hasan, Taufiq, Rahim Saeidi, John HL Hansen, and David A Van Leeuwen (2013). "Duration mismatch compensation for i-vector based speaker recognition systems". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 7663–7667.

Heck, Larry P, Yochai Konig, M Kemal Sönmez, and Mitch Weintraub (2000). "Robustness to telephone handset distortion in speaker recognition by discriminative feature design". In: *Speech Communication*. Vol. 31. 2, pp. 181–192.

Heigold, Georg, Ignacio Moreno, Samy Bengio, and Noam Shazeer (2016). "End-to-end text-dependent speaker verification". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5115–5119.

Hershey, John R, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe (2016). “Deep clustering: Discriminative embeddings for segmentation and separation”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 31–35.

Ioffe, Sergey (2006). “Probabilistic linear discriminant analysis”. In: *Computer Vision–ECCV 2006*, pp. 531–542.

Kenny, Patrick, Douglas Reynolds, and Fabio Castaldo (2010). “Diarization of telephone conversations using factor analysis”. In: *IEEE Journal of Selected Topics in Signal Processing* 4.6, p. 1059.

Kenny, Patrick, Themos Stafylakis, Pierre Ouellet, Vishwa Gupta, and Md Jahangir Alam (2014). “Deep neural networks for extracting Baum-Welch statistics for speaker recognition”. In: *Proc. Odyssey*.

Khokhlov, Yuri, Alexander Zatvornitskiy, Ivan Medennikov, Ivan Sorokin, Tatiana Prisyach, Aleksei Romanenko, Anton Mitrofanov, Vladimir Bataev, Andrei Andrusenko, Mariya Korenevskaya, et al. (2019). “R-vectors: New technique for adaptation to room acoustics”. In: *Proc. Interspeech*.

Ko, Tom, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur (2015). “Audio augmentation for speech recognition.” In: *Interspeech*, pp. 3586–3589.

Ko, Tom, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur (2017). “A study on data augmentation of reverberant speech for robust speech recognition”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, pp. 5220–5224.

Konig, Yochai, Larry Heck, Mitch Weintraub, Kemal Sonmez, et al. (1998). “Nonlinear Discriminant Feature Extraction for robust Text-independent Speaker Recognition”. In: *Proc. RLA2C, ESCA workshop on Speaker Recognition and its Commercial and Forensic Applications*.

Lei, Yun, Lukas Burget, Luciana Ferrer, Martin Graciarena, and Nicolas Scheffer (2012). “Towards noise-robust speaker recognition using probabilistic linear discriminant analysis”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, pp. 4253–4256.

Lei, Yun, Nicolas Scheffer, Luciana Ferrer, and Mitchell McLaren (2014). “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, pp. 1695–1699.

Lin, Min, Qiang Chen, and Shuicheng Yan (2013). “Network in network”. In: *arXiv preprint arXiv:1312.4400*.

Liu, Yi, Yao Tian, Liang He, and Jia Liu (2016). "Investigating Various Diarization Algorithms for Speaker in the Wild (SITW) Speaker Recognition Challenge." In: *Interspeech*, pp. 853–857.

Lopez-Moreno, Ignacio, Javier Gonzalez-Dominguez, Oldrich Plchot, David Martinez, Joaquin Gonzalez-Rodriguez, and Pedro Moreno (2014). "Automatic language identification using deep neural networks". In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pp. 5337–5341.

Maimon, Oded and Lior Rokach (2005). *Data mining and knowledge discovery handbook*. Springer. Chap. Clustering methods, pp. 321–352.

Martin, Alvin F and Mark A Przybocki (2001). "Speaker recognition in a multi-speaker environment". In: *Seventh European Conference on Speech Communication and Technology*.

Matejka, Pavel, Le Zhang, Tim Ng, Ondrej Glembek, Jeff Z Ma, Bing Zhang, and Sri Harish Mallidi (2014). "Neural network bottleneck features for language identification". In: *Proc. Odyssey*, pp. 299–304.

McCree, Alan (2014). "Multiclass discriminative training of i-vector language recognition". In: *Proc. Odyssey*, pp. 166–172.

McCree, Alan, Gregory Sell, and Daniel Garcia-Romero (2016). “Augmented data training of joint acoustic/phonotactic DNN i-vectors for NIST LRE15”. In: *Proc. Odyssey: Speaker Lang. Recognit. Workshop*, pp. 204–209.

Mccree, Alan, David Snyder, Gregory Sell, and Daniel Garcia-Romero (2018). “Language Recognition for Telephone and Video Speech: The JHU HLTCOE Submission for NIST LRE17”. In: *Odyssey*.

McLaren, Mitchell, Yun Lei, and Luciana Ferrer (2015). “Advances in deep neural network approaches to speaker recognition”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pp. 4814–4818.

McLaren, Mitchell, Luciana Ferrer, Diego Castan, and Aaron Lawson (2016). “The 2016 Speakers in the Wild Speaker Recognition Evaluation.” In: *Interspeech*, pp. 823–827.

McLaren, ML, Diego Castan, Mahesh Kumar Nandwana, Luciana Ferrer, and Emre Yilmaz (2018). “How to train your speaker embeddings extractor”. In: *Odyssey: The Speaker and Language Recognition Workshop, Les Sables d’Olonne*.

Nagrani, Arsha, Joon Son Chung, and Andrew Zisserman (2017). “VoxCeleb: a large-scale speaker identification dataset”. In: *Interspeech*.

NIST 2017 Language Recognition Evaluation Plan (2017). https://www.nist.gov/sites/default/files/documents/2017/06/01/lre17_eval_plan-2017-05-31_v2.pdf.

NIST Speaker Recognition Evaluation 2016 (2016). https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2016/.

NIST Speaker Recognition Evaluation 2018 (2018). https://www.nist.gov/sites/default/files/documents/2018/08/17/sre18_eval_plan_2018-05-31_v6.pdf.

Novotný, O., P. Matějka, O. Glembeck, O Plchot, F. Grézl, L. Burget, and J. Černocký (2016). "Analysis of the dnn-based SRE systems in multi-language conditions". In: *Spoken Language Technology Workshop (SLT)*. IEEE.

Novotnỳ, Ondrej, Pavel Matejka, Oldrich Plchot, Ondrej Glembek, Lukás Burget, and Jan Cernockỳ (2016). "Analysis of Speaker Recognition Systems in Realistic Scenarios of the SITW 2016 Challenge." In: *Interspeech*, pp. 828–832.

Okabe, Koji, Takafumi Koshinaka, and Koichi Shinoda (2018). "Attentive Statistics Pooling for Deep Speaker Embedding". In: *Proc. Interspeech 2018*, pp. 2252–2256.

Peddinti, Vijayaditya, Daniel Povey, and Sanjeev Khudanpur (2015). “A time delay neural network architecture for efficient modeling of long temporal contexts.” In: *Interspeech*, pp. 3214–3218.

Povey, D., A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely (2011). “The Kaldi speech recognition toolkit”. In: *Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop*.

Povey, Daniel, Xiaohui Zhang, and Sanjeev Khudanpur (2015). “Parallel training of Deep Neural Networks with Natural Gradient and Parameter Averaging”. In: *CoRR* abs/1410.7455. URL: http://arxiv.org/abs/1410.7455. published.

Prince, Simon JD and James H Elder (2007). “Probabilistic Linear Discriminant Analysis for Inferences About Identity”. In: *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007.* Pp. 1–8.

Reynolds, Douglas A, Thomas F Quatieri, and Robert B Dunn (2000). “Speaker verification using adapted Gaussian mixture models”. In: *Digital signal processing* 10.1-3, pp. 19–41.

Richardson, Fred, Douglas Reynolds, and Najim Dehak (2015). “Deep neural network approaches to speaker and language recognition”. In: *Signal Processing Letters, IEEE* 22.10, pp. 1671–1675.

Ryant, N, K Church, C Cieri, A Cristia, J Du, S Ganapathy, and M Liberman (2018). *First DIHARD challenge evaluation plan*.

Sadjadi, Seyed Omid, Jason Pelecanos, and Sriram Ganapathy (2016). “The IBM Speaker Recognition System: Recent Advances and Error Analysis”. In: *Interspeech*, pp. 3633–3637.

Salman, Ahmad (2012). “Learning speaker-specific characteristics with deep neural architecture”. PhD thesis. University of Manchester.

Sell, G., D. Snyder, A. Mccree, D. Garcia-Romero, J. Villalba, M. Maciejewski, V. Manohar, N. Dehak, D. Povey, S. Watanabe, and S. Khudanpur (2018). “Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge”. In: *Proceedings of the 19th Annual Conference of the International Speech Communication Association, INTERSPEECH 2018*. Hyderabad, India, pp. 2808–2812. URL: http://www.danielpovey.com/files/2018{_}interspeech{_}dihard.pdfhttp://dx.doi.org/10.21437/Interspeech.2018-1893.

Sell, Gregory and Daniel Garcia-Romero (2014). “Speaker diarization with PLDA i-vector scoring and unsupervised calibration”. In: *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pp. 413–417.

Sell, Gregory and Daniel Garcia-Romero (2015a). “Diarization resegmentation in the factor analysis subspace”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4794–4798.

Sell, Gregory and Daniel Garcia-Romero (2015b). “Diarization resegmentation in the factor analysis subspace”. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4794–4798.

Senoussaoui, Mohammed, Patrick Kenny, Themos Stafylakis, and Pierre Dumouchel (2014). “A study of the cosine distance-based mean shift for telephone speech diarization”. In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 22.1, pp. 217–227.

Shum, Stephen, Najim Dehak, and James Glass (2012). “On the use of spectral and iterative methods for speaker diarization”. In: *Thirteenth Annual Conference of the International Speech Communication Association*.

Shum, Stephen, Najim Dehak, Ekapol Chuangsuwanich, Douglas Reynolds, and James Glass (2011). “Exploiting intra-conversation variability for speaker diarization”. In: *Twelfth Annual Conference of the International Speech Communication Association*.

Shum, Stephen H, Najim Dehak, Réda Dehak, and James R Glass (2013). “Unsupervised methods for speaker diarization: An integrated and iterative approach”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.10, pp. 2015–2028.

Silnova, Anna, Niko Brümmer, Daniel Garcia-Romero, David Snyder, and Lukáš Burget (2018). “Fast Variational Bayes for Heavy-tailed PLDA Applied to i-vectors and x-vectors”. In: *Proc. Interspeech 2018*, pp. 72–76.

Snyder, David, Guoguo Chen, and Daniel Povey (2015). *MUSAN: A Music, Speech, and Noise Corpus*. arXiv:1510.08484v1. eprint: 1510.08484.

Snyder, David, Daniel Garcia-Romero, and Daniel Povey (2015). “Time delay deep neural network-based universal background models for speaker recognition”. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pp. 92–97.

Snyder, David, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur (2017). “Deep Neural Network Embeddings for Text-Independent Speaker Verification”. In: *Proc. Interspeech*, pp. 999–1003.

Snyder, David, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur (2018a). “Spoken Language Recognition using X-vectors”. In: *Odyssey*.

Snyder, David, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur (2018b). “X-vectors: Robust DNN Embeddings for Speaker Recognition”. In: *2018 IEEE International Conference on Acoustics, Speech* *and Signal Processing (ICASSP)*. IEEE. URL: http://www.danielpovey.com/files/2018_icassp_xvectors.pdf.

Song, Yan, Bing Jiang, YeBo Bao, Si Wei, and Li-Rong Dai (2013). "I-vector representation based on bottleneck features for language identification". In: *Electronics Letters* 49.24, pp. 1569–1570.

Sturim, Douglas E and Douglas A Reynolds (2005). "Speaker adaptive cohort selection for Tnorm in text-independent speaker verification". In: *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*. Vol. 1. IEEE, pp. I–741.

The 2010 NIST Speaker Recognition Evaluation (2010). https://www.nist.gov/system/files/documents/itl/iad/mig/SRE10_maineval_workshop_public_brief.pdf.

The 2015 Language Recognition i-Vector Machine Learning Challenge (2015). https://www.nist.gov/sites/default/files/documents/itl/iad/mig/lre_ivectorchallenge_rel_v1-1.pdf.

Tracey, Jennifer and Stephanie Strassel (2018). "Vast: A corpus of video annotation for speech technologies". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Tzanetakis, George and Perry Cook (2002). “Musical genre classification of audio signals”. In: *IEEE Transactions on speech and audio processing* 10.5, pp. 293–302.

Variani, Ehsan, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez (2014). “Deep neural networks for small footprint text-dependent speaker verification”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4052–4056.

Wang, Quan, Carlton Downey, Li Wan, Philip Andrew Mansfield, and Ignacio Lopz Moreno (2018). “Speaker diarization with lstm”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5239–5243.

Yamada, Takanori, Longbiao Wang, and Atsuhiko Kai (2013). “Improvement of distant-talking speaker identification using bottleneck features of DNN.” In: *Interspeech*, pp. 3661–3664.

Vita

David Snyder received his BS in Computer Science from Portland State University in 2012. He began a PhD in the Computer Science department at Johns Hopkins University in 2013. Daniel Povey and Sanjeev Khudanpur co-advised his research, which focused on discriminatively trained representations for speaker recognition, spoken language recognition, and speaker diarization. As a contributor to the Kaldi toolkit, he developed and maintained the speaker recognition and diarization recipes. David has published over 20 papers in speech processing, which have been recognized by best paper and student grant awards. In 2016, he interned at Amazon Alexa, where he worked on wake word detection. David is a recipient of a 2013 NSF Graduate Research Fellowship and the 2018 Frederick Jelinek Fellowship. In February 2020, David joined the Siri speech group at Apple, where he currently works as a machine learning scientist in Seattle, Washington.