UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TECNOLOGÍA ELECTRÓNICA Y DE LAS COMUNICACIONES

BOTTLENECK AND EMBEDDING REPRESENTATION OF SPEECH FOR DNN-BASED LANGUAGE AND SPEAKER RECOGNITION

–TESIS DOCTORAL–

RECONOCIMIENTO DE IDIOMA Y LOCUTOR BASADO EN REPRESENTACIÓN BOTTLENECK Y EMBEDDING A PARTIR DE REDES NEURONALES PROFUNDAS

Author: Alicia Lozano Díez
(Ingeniera en Informática y Licenciada en Matemáticas, Universidad Autónoma de Madrid)

A Thesis submitted for the degree of:

Doctor of Philosophy

Madrid, March 2018

Department:	Tecnología Electrónica y de las Comunicaciones Escuela Politécnica Superior Universidad Autónoma de Madrid (UAM), SPAIN
PhD Thesis:	Bottleneck and Embedding Representation of Speech for DNN-based Language and Speaker Recognition
Author:	**Alicia Lozano Díez** Ingeniera en Informática y Licenciada en Matemáticas Universidad Autónoma de Madrid, SPAIN
Advisors:	**Joaquín González Rodríguez** Doctor Ingeniero de Telecomunicación Universidad Autónoma de Madrid, SPAIN
	Javier González Domínguez Doctor Ingeniero Informático y de Telecomunicación
Year:	2018
Committee:	President: **Luis A. Hernández Gómez** Universidad Politécnica de Madrid, SPAIN
	Secretary: **Daniel Ramos Castro** Universidad Autónoma de Madrid, SPAIN
	Vocal 1: **Ascensión Gallardo Antolín** Universidad Carlos III, SPAIN
	Vocal 2: **Mitchell McLaren** SRI International, U.S.A.
	Vocal 3: **Lukáš Burget** Brno University of Technology, CZECH REPUBLIC

The research described in this Thesis was carried out within the Audias Research Group at the Dept. of Tecnología Electrónica y de las Comunicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid (from 2014 to 2018). This work was funded by the PhD scholarship Ayudas para contratos predoctorales para la formación de doctores 2013 (BES-2013-064886), under the project CMC-V2: Caracterización, Modelado y Compensación de Variabilidad en la Señal de Voz (TEC2012-37585-C02-01), funded by Ministerio de Economía y Competitividad, Spain.

The author was awarded with a PhD scholarship "Ayudas para contratos predoctorales para la formación de doctores 2013 (BES-2013-064886)" from Ministerio de Economía y Competitividad from 2014 to 2017 which supported the research summarized in this Dissertation.

Derived from partial work of this dissertation, the author was granted with an intern scholarship "Ayuda a la Movilidad Predoctoral para la realización de Estancias Breves en Centros de I+D 2014 (EEBB-I-15-10381)" from Ministerio de Economía y Competitividad. This internship was completed at the speech processing group (Speech@FIT) at the Faculty of Information Technology of Brno University of Technology, Brno, Czech Republic, under the supervision of Dr. Jan "Honza" Černocký during the period March-July 2015.

Derived from partial work of this dissertation, the author was granted with an intern scholarship "Ayuda a la Movilidad Predoctoral para la realización de Estancias Breves en Centros de I+D 2015 (EEBB-I-16-10905)" from Ministerio de Economía y Competitividad. This internship was completed at the Speech Technology and Research (STAR) Laboratory at SRI International, Menlo Park, California, USA, under the supervision of Dr. Dimitra Vergyri during the period May-September 2016.

Derived from partial work of this dissertation, the author was granted with an intern scholarship "Ayuda a la Movilidad Predoctoral para la realización de Estancias Breves en Centros de I+D 2016 (EEBB-I-17-12491)" from Ministerio de Economía y Competitividad. This internship was completed at the speech processing group (Speech@FIT) at the Faculty of Information Technology of Brno University of Technology, Brno, Czech Republic, under the supervision of Dr. Jan "Honza" Černocký during the period September-November 2017.

Abstract

AUTOMATIC SPEECH RECOGNITION has experienced a breathtaking progress in the last few years, partially thanks to the introduction of deep neural networks into their approaches. This evolution in speech recognition systems has spread across related areas such as language and speaker recognition, where deep neural networks have noticeably improved their performance.

In this PhD thesis, we have explored different approaches to the tasks of speaker and language recognition, focusing on systems where deep neural networks become part of traditional pipelines, replacing some stages or the whole system itself.

Specifically, in the first experimental block, end-to-end language recognition systems based on deep neural networks are analyzed, where the neural network is used as classifier directly, without the use of any other backend but performing the language recognition task from the scores (posterior probabilities) provided by the network. Besides, these research works are focused on two architectures, convolutional neural networks and long short-term memory (LSTM) recurrent neural networks, which are less demanding in terms of computational resources due to the reduced amount of free parameters in comparison with other deep neural networks. Thus, these systems constitute an alternative to classical i-vectors, and achieve comparable results to them, especially when dealing with short utterances. In particular, we conducted experiments comparing a system based on convolutional neural networks with classical Factor Analysis GMM and i-vector reference systems, and evaluate them on two different tasks from the National Institute of Standards and Technology (NIST) Language Recognition Evaluation (LRE) 2009: one focused on language-pairs and the other, on multi-class language identification. Results shown comparable performance of the convolutional neural network based approaches and some improvements are achieved when fusing both classical and neural network approaches. We also present the experiments performed with LSTM recurrent neural networks, which have proven their ability to model time depending sequences. We evaluate our LSTM-based language recognition systems on different subsets of the NIST LRE 2009 and 2015, where LSTM systems are able to outperform the reference i-vector system, providing a model with less parameters, although more prone to overfitting and not able to generalize as well as i-vector in mismatched datasets.

In the second experimental block of this Dissertation, we explore one of the most prominent applications of deep neural networks in speech processing, which is their use as feature extractors. In this kind of systems, deep neural networks are used to obtain a frame-by-frame representation of the speech signal, the so-called *bottleneck feature* vector, which is learned directly by the network and is then used instead of traditional acoustic features as input in language and speaker recognition systems based on i-vectors. This approach revolutionized these two fields, since they highly outperformed classical systems which had been state-of-the-art for many years (i-vector based on acoustic features). Our analysis focuses on how different configurations of the

neural network used as bottleneck feature extractor, and which is trained for automatic speech recognition, influences performance of resulting features for language and speaker recognition. For the case of language recognition, we compare bottleneck features from networks that vary their depth in terms of number of hidden layers, the position of the bottleneck layer where it comprises the information and the number of units (size) of this layer, which would influence the representation obtained by the network. With the set of experiments performed on bottleneck features for speaker recognition, we analyzed the influence of the type of features used to feed the network, their pre-processing and, in general, the optimization of the network for the task of feature extraction for speaker recognition, which might not mean the optimal configuration for ASR.

Finally, the third experimental block of this Thesis proposes a novel approach for language recognition, in which the neural network is used to extract a fixed-length utterance-level representation of speech segments known as *embedding*, able to replace the classical i-vector, and overcoming the variable length sequence of feature provided by the bottleneck features. This embedding based approach has recently shown promising results for speaker verification tasks, and our proposed system was able to outperform a strong state-of-the-art reference i-vector system on the last challenging language recognition evaluations organized by NIST in 2015 and 2017. Thus, we analyze language recognition systems based on embeddings, and explore different deep neural network architectures and data augmentation techniques to improve results of our system. In general, these embeddings are a fair competitor to the well-established i-vector pipeline which allows replacing the whole i-vector model by a deep neural network. Furthermore, the network is able to extract complementary information to the one contained in the i-vectors, even from the same input features. All this makes us consider that this contribution is an interesting research line to explore in other fields.

A mis padres y a mi hermana.
A mis abuelas.
A todos los que están ahí, que forman mi familia extendida.

Acknowledgements

DURANTE EL DESARROLLO DE ESTA TESIS, han sido muchas las personas que me han ayudado en cada momento y sin las que no hubiera sido posible llegar a este *punto y aparte.* Son muchos, por tanto, los agradecimientos que se han ido acumulando a lo largo de estos años.

En primer lugar, quiero agradecer a mi tutor y director, Joaquín González Rodríguez, la oportunidad que me brindó para empezar este camino en el grupo de investigación, y por su apoyo y sabios consejos durante toda esta trayectoria. Por supuesto, agradecer esta guía en el camino a mi otro director, Javier González Domínguez, que desde cerca o desde la distancia, ha hecho posible empezar, continuar y finalizar esta etapa. Y extensivos estos agradecimientos a Doroteo Torre Toledano y Daniel Ramos Castro, que también me han acompañado y guiado durante todos estos años.

Quiero continuar agradeciendo su infinito apoyo a Javier Franco y Rubén Zazo, con quienes he compartido muchos y muy buenos momentos tanto en el laboratorio como fuera de él. En este mismo laboratorio (y en la planta de arriba), he tenido el placer de coincidir con unos estupendos compañeros, que han hecho este camino mucho más ameno. A todos vosotros, ¡gracias!

As part of this journey, I have had the incredible opportunity to visit two great research laboratories, Speech@FIT (BUT, Brno) and STAR Lab (SRI International, California), full of wonderful people. Thanks to all of you for hosting me, and give me the chance to learn from you! All those moments inside and outside the lab will be kept in my mind as one of the most enriching experiences in my life, without forgetting the rest of the people I met during those months abroad that made that experience even greater. Thank you very much!

Por supuesto, todo esto no sería posible ni la mitad de divertido sin todos mis amigos (*chikys y leyendas* de forma resumida, y todos los que han estado ahí a lo largo de tantos años), que se han convertido en mi familia. Con ellos disfruto de mi tiempo libre, fines de semana, escapadas y vacaciones, y juegan un papel muy importante en mi vida, por lo que este trabajo, también se lo dedico a ellos.

Y finalmente (y como ya se sabe, no por ello menos importante ni mucho menos), no hay palabras suficientes para agradecer todo a mis padres y mi hermana (y amiga, eterna compañera de vivencias). El infinito cariño y apoyo por parte de todos ellos me lleva acompañando toda la vida, y todo se lo debo a ellos, por lo que nunca podré expresar con palabras mi inmenso agradecimiento.

A todos, gracias.

Alicia Lozano Díez
Madrid, Abril 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine Learning and, in particular, the family of algorithms encompassed under the name of *deep learning*, is a hot topic in a number of research communities, and nowadays more and more frequently used in everyday applications and devices.

Broadly speaking, machine learning can be defined as a set of methods that allows machines to learn directly from a given set of data, without being necessary to provide them with some specific rules given by an expert in the target task. This process of learning from data is known as *training*.

Algorithms based on machine learning have been used for many years, and they include a wide variety of approaches. Among them, we can highlight Artificial Neural Networks (ANN) that, inspired by the structure of biological neural networks, are able to model data using complex transformations with an architecture based on layers of hidden units (or neurons). Following this line and further inspired by human brain behavior, deep neural networks (DNN) stack several layers of hidden units, which allows for even more complex models. Even though this kind of machine learning tools were studied many years ago, and theoretical training algorithms existed already, the lack of big databases and hardware limitations made them be outperformed by some other approaches in most of the applications. However, these two constraints have been overcome in the last couple of decades, allowing them to reappear with enough strength to almost conquer most fields of application such as computer vision and speech processing.

Furthermore, speech is the most natural way of communication among humans and everyday devices are using increasingly this type of interaction with humans. The amount of information contained in the speech signal is huge, including not just the message itself, but information about the person's identity, her/his age, the language being used, the emotions the person is expressing, and the subject the person is talking about, among others. All this information can be used by automatic systems in a wide range of applications and this results in a variety of research fields that specifically target each of them as their main objective, but at the same time they are closely related and are based on the same or similar underlying knowledge, even sharing some modules of the automatic systems developed for the task on which each field is focused. One of the largest research fields concerning speech processing is the one that aims to automatically extract the speech content, known as Automatic Speech Recognition (ASR), where deep neural networks broke into in the last years and were able to extract more efficiently the information contained in the rich voice signal, outperforming by far the best already existing techniques. Thereby, the application of these deep learning based approaches was quickly spread and adopted by related research fields such as language and speaker recognition on which this Dissertation is focused. In this case, systems aim to automatically identify the language or the speaker's identity in a given speech utterance, respectively. These two fields span also a wide range of applications such as security, access control or forensics, and deep learning algorithms have been able to widely outperform other approaches. In this Dissertation, we explore the application of neural networks in these two research areas in different ways: used as end-to-end systems or as feature extractors both at frame or utterance level. This Thesis might be considered an approach towards understanding a bit better the way these algorithms exploit information contained in the signal for different target tasks.

1.1. DNNs: a Breakthrough in Speech Processing

Artificial Neural Networks, or just Neural Networks (NNs), are machine learning architectures inspired by the human's brain behavior, which are able to learn directly from data. This learning process, known as *training*, was possible thanks to the development of the backpropagation algorithm [Rumelhart *et al.*, 1985], which allows estimation of gradients and constitutes the basis of training algorithms such as gradient descend. When neural networks stack more than one hidden layers in their structure, they are known as *deep* architectures (Deep Neural Networks, DNNs), and the main underlying idea is that each layer is able to learn a different pattern from the input data, extracting information at different levels of abstraction. Even though theoretical algorithms already existed before 1990, it was not until hardware and data limitations were overcome when these more complex architectures with stacked hidden layers started to spread through different research fields with stunning performance in comparison with traditional state-of-the-art systems. Several fields benefited from these powerful learning tools, such as for instance object recognition [Krizhevsky *et al.*, 2012], face recognition [Taigman *et al.*, 2014] and automatic translation [Bahdanau *et al.*, 2014], to mention a few.

In 2006, Deep Belief Networks (DBNs) were presented in [Hinton *et al.*, 2006]. This approach was able to overcome the poor performance of training algorithms obtained so far by including an unsupervised training stage for each hidden layer (consisting on Restricted Boltzmann Machines, RBMs) to initialize the network, usually known as *pre-training*. After this initialization method, supervised training was performed via backpropagation in the step known as *fine-tuning* in order to adjust the parameters to the target task.

In speech processing, it was in 2012 when these DBNs were presented in [Mohamed *et al.*, 2012] for acoustic modeling and improvements were obtained by replacing the GMM with these *pre-trained* neural networks. The *pre-training* and *fine-tuning* steps seemed to be beneficial, especially when limited amount of labeled data is available, and even though it was a more complex way of training, these impressive results showed by the use of DNNs for acoustic modeling led to a resurgence of interest in neural networks in the speech research community [Hinton *et al.*, 2012a].

Since then, DNNs are among the most popular methods used in many stages of speech recognition systems. They have been used to replace GMMs in the GMM-HMM classical pipeline [Hinton *et al.*, 2012a], and also as feature extractors [Bao *et al.*, 2013; Deng *et al.*, 2010; Grezl *et al.*, 2007], or even as end-to-end ASR systems [Hannun *et al.*, 2014], which has motivated as well their use in closely related fields such as speaker and language recognition.

1.2. Motivation

The outstanding results obtained by deep neural networks in many research fields drew the attention of researchers in speech signal processing, where, in fact, these machine learning tools became the state-of-the-art in the last few years. They were able to noticeably outperform classical speech recognition systems, which had experienced marginal gains for years by incremental improvements over the same GMM-HMM pipeline. These impressive results obtained by DNNs, which learned directly from the input data, showed their power as tools to extract the complex information contained in the speech signal.

Moreover, speech is becoming more and more popular nowadays as the way for humans to interact with machines, and this rises the interest of researchers in closely related fields such as speaker and language recognition as well. These tasks, on which this Thesis is focused, can help improving or making easier different daily used applications such as call centers that route their calls depending on the language, or control access systems that need accurate speaker identification methods.

This powerful tandem of DNNs and speaker/language recognition that was emerging seemed an interesting research line to explore due to the wide range of applications and approaches that surge from their combination.

Thus, different DNN-based approaches to language and speaker recognition, from end-to-end to frame or utterance level feature extractors, are presented in this Dissertation in order to explore their ability to leverage information from the speech signal for these two target tasks.

Even though we focused on these two research fields, related areas have shown improvements following the same lines we studied in this Thesis, which supports the idea of DNNs being one of the most powerful tools for signal processing that can learn useful information from complex signals and might be adjusted to another wide range of applications.

1.3. Goals of the Thesis

The main goal of this Thesis is to explore emerging approaches to language and speaker recognition based on deep neural networks. This goal can be split into the following ones:

- Reviewing existing approaches for speech, language and speaker recognition, both before and after the introduction of DNNs into their pipelines.
- Experimenting with end-to-end DNN-based systems for language recognition, exploring two different architectures with stunning results in related works while reducing the number of free parameters with respect to other DNN architectures: convolutional neural networks and long short-term memory recurrent neural networks.
- Exploring different configurations of DNNs used as frame-wise feature extractors (*bottleneck features*) trained for ASR and the influence of various designs of the DNN in bottleneck feature-based language and speaker recognition.
- Proposing an approach to language recognition that uses DNNs to extract utterance level representations (*embeddings*), as fixed length vectors (similar to traditional i-vectors).

1.4. Outline of the Dissertation

This Dissertation is organized according to the main goals described in Section 1.3, providing first some theoretical background and motivation of the techniques used in the experimental part of this Thesis.

In particular, the chapters structure and description is as follows:

- Chapter 1 reviews the successful applications of deep neural networks to speech processing, and presents the motivation, goals, organization and contributions of this Thesis.
- Chapter 2 provides a summary of existing approaches in state-of-the-art language and speaker recognition fields, including the basic concepts of traditional i-vector based systems and deep neural networks.
- Chapter 3 introduces end-to-end approaches to language recognition, and provides an study of convolutional neural networks and long short-term memory recurrent neural networks to tackle language recognition in short utterances.
- Chapter 4 presents an empirical analysis of DNNs used as bottleneck feature extractors (frame-by-frame feature vectors), exploring different designs and their influence in the final performance of language and speaker recognition.
- Chapter 5 describes DNN-based embeddings as fixed-length utterance level vector representation of the speech and proposes their use in language recognition systems, providing better performance than the classical i-vector representation.
- Chapter 6 summarizes the conclusions that might be drawn from this Dissertation and highlights some future research lines.

The organization and dependence between the mentioned Chapters is depicted in Figure 1.1.

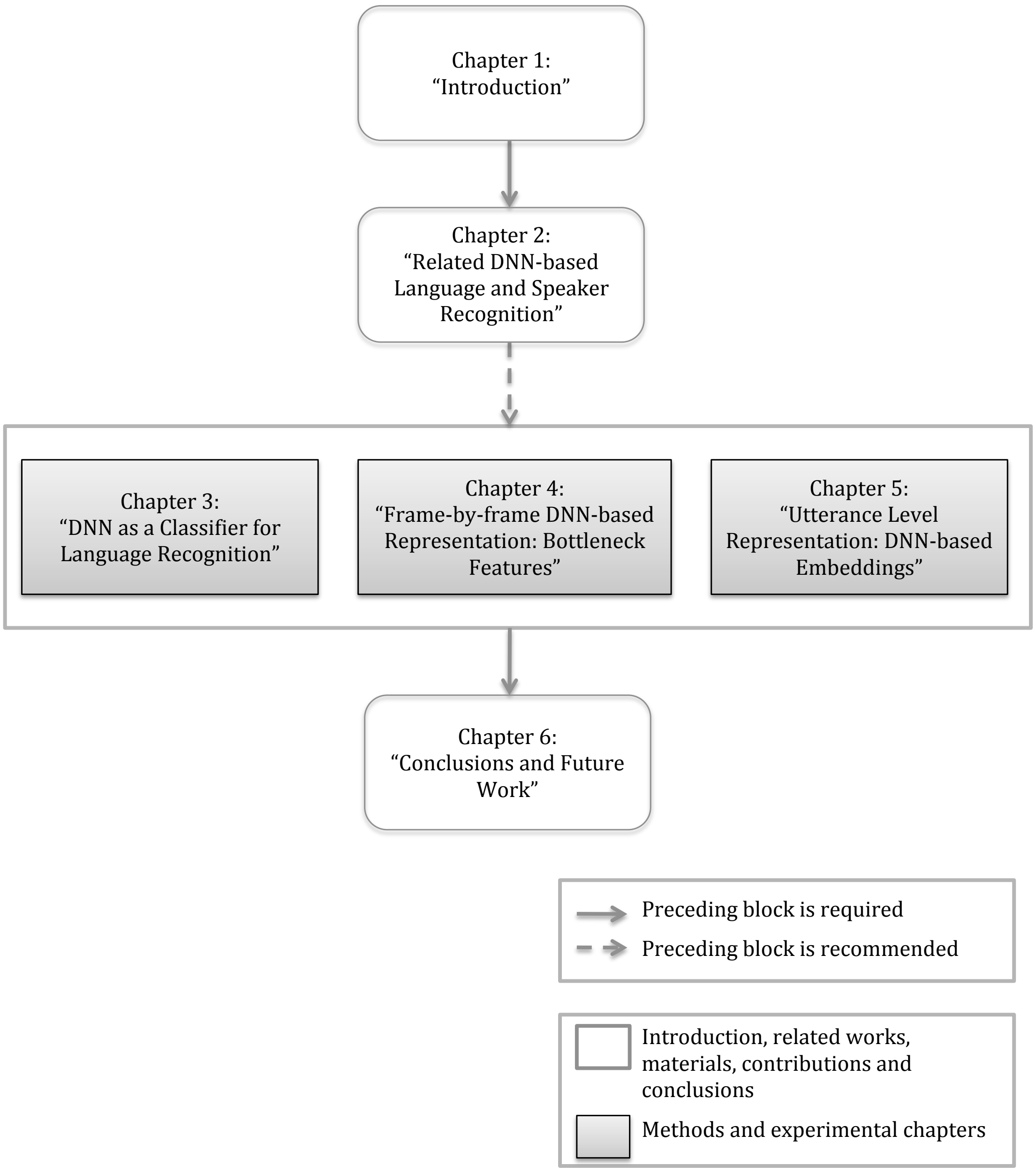

Chapter 1:
"Introduction"
Chapter 2:
"Related DNN-based Language and Speaker Recognition"
Chapter 3:
"DNN as a Classifier for Language Recognition"
Chapter 4:
"Frame-by-frame DNN-based Representation: Bottleneck Features"
Chapter 5:
"Utterance Level Representation: DNN-based Embeddings"
Chapter 6:
"Conclusions and Future Work"
Preceding block is required
Preceding block is recommended
Introduction, related works, materials, contributions and conclusions
Methods and experimental chapters

Figure 1.1: *Dependence among Dissertation chapters.*

1.5. Detailed Research Contributions

The research contributions of this PhD Thesis are the following (publications in each group -journal or conferences- are ordered by date):

- JOURNAL PUBLICATIONS WITH JCR

  A. Lozano-Diez, *R. Zazo, D. T. Toledano and J. Gonzalez-Rodriguez, "An analysis of the influence of deep neural network (DNN) topology in bottleneck feature based language recognition", PLoS ONE 12(8): e0182580, August 2017.*

  In this paper, we analyze the bottleneck DNN-based architecture for language recognition, and present a systematic study of different variations of the DNN architecture, including position and size of the bottleneck layer and the number of layers in the DNN, which influences the information comprised in the bottleneck features used as input for language recognition based on an i-vector model. This is done in order to obtain the best configuration of the DNN used as bottleneck feature extractor for this setup and to provide some insights into this widely used features in state-of-the-art language recognition systems. Our systems are evaluated on the challenging NIST LRE 2015 framework, providing advantages and disadvantages of bottleneck features vs. acoustic features (MFCCs) in our setup. Experimental results in this article are included in Chapter 4.

  R. Zazo, ***A. Lozano-Diez***, *J. Gonzalez-Dominguez, D. T. Toledano, J. Gonzalez-Rodriguez, "Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks", PLoS ONE 11(1): e0146917, January 2016.*

  This article includes a study about language recognition systems based in an end-to-end approach using long short-term memory (LSTM) recurrent neural networks. In particular, we aimed to replicate the previous work published by the co-advisor of this Thesis at Google Inc. (New York, U.S.A.) [Gonzalez-Dominguez *et al.*, 2014] but with modest resources reducing the computational power needed and using an open-source toolkit instead of the proprietary software used in the original work. Performance of this end-to-end LSTM-based approach is compared to an i-vector based system, which is outperformed by LSTMs, especially when dealing with short test utterances (less than 3 seconds of speech). Experimental results in this article are included in Chapter 3.

- PEER-REVIEWED INTERNATIONAL CONFERENCES

***A. Lozano-Diez**, O. Plchot, P. Matejka, O. Novotny, J. Gonzalez-Rodriguez, "Analysis of DNN-based Embeddings for Language Recognition on the NIST LRE 2017", in Proc. of Odyssey 2018 Speaker and Language Recognition Workshop, Les Sables d'Olonne, France, June 2018 (accepted).*

In this work, we analyze the recent successful application of DNNs as embedding (utterance level) representation extractors for speaker recognition in [Snyder *et al.*, 2017], and we adapt it for the problem of language recognition. We evaluate the performance of these DNN-based embeddings on the last NIST LRE 2017 setup, and analyze the influence of data augmentation by means of noise and reverberation addition and some other configurations of the DNN for the final task of language identification. This system, fully developed by the first author, was used as part of the Brno University of Technology (BUT) submission to the NIST LRE 2017, and improved afterwards. Obtained results are shown in Chapter 5 of this Dissertation.

*O. Plchot, P. Matejka, O. Novotny, S. Cumani, **A. Lozano-Diez**, J. Slavicek, M. Diez, F. Grezl, O. Glembek, M. Kamsali, A. Silnova, L. Burget, L. Ondel, S. Kesiraju, J. Rohdin, "Analysis of BUT-PT Submission for NIST LRE 2017", in Proc. of Odyssey 2018 Speaker and Language Recognition Workshop, Les Sables d'Olonne, France, June 2018 (accepted).*

This work includes the whole system description of the Brno University of Technology (BUT) submission to the NIST LRE 2017 (in collaboration with Politecnico di Torino, Universidad Autonoma de Madrid and Phonexia), and analyzes post-evaluation improvements, including some DNN-based embeddings results. These results have been summarized in Chapter 5 this Dissertation, although a more extended analysis is conducted for the DNN-embeddings subsystem, one of the main research lines of this Thesis.

***A. Lozano-Diez**, O. Plchot, P. Matejka and J. Gonzalez-Rodriguez, "DNN based Embeddings for Language Recognition", in Proc. of ICASSP, Calgary (Canada), April 2018 (accepted).*

In this research paper we present the novel application of DNN-embeddings for language recognition, successfully applied to speaker recognition in [Snyder *et al.*, 2017]. We modified the architecture with respect to this previous work, adjusting it to the language recognition problem and analyzing some techniques such as dimensionality reduction via Principal Components Analysis (PCA), which resulted in improvements and pointed to some other directions we explored in subsequent works. Our proposed systems are evaluated on the challenging NIST LRE 2015 dataset, yielding results competitive with a strong i-vector system in the state-of-the-art. This analysis is described and analyzed in Chapter 5.

*R. Zazo, **A. Lozano-Diez** and J. Gonzalez-Rodriguez, "Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks", in Proc. of Odyssey 2016 Speaker and Language Recognition Workshop, Bilbao, Spain, June 2016.*

In this paper, an LSTM end-to-end approach for language identification is evaluated in different datasets from NIST LRE 2009 and 2015, providing an analysis of this system performance in comparison with i-vector systems, which are outperformed by LSTM-based systems when short test segments are considered. This is in line with other DNN-based systems used in related works. Results of this analysis are presented in Chapter 3 of this Dissertation.

***A. Lozano-Diez**, A. Silnova, P. Matejka, O. Glembek, O. Plchot, J. Pesan, L. Burget and J. Gonzalez-Rodriguez, "Analysis and Optimization of Bottleneck Features for Speaker Recognition", in Proc. of Odyssey 2016 Speaker and Language Recognition Workshop, Bilbao, Spain, June 2016.*

In this work, bottleneck features are analyzed in the context of speaker recognition. In particular, an analysis of different processing of input features to the DNN used as bottleneck feature extractor is presented, including different features as well as normalization techniques. DNNs used as bottleneck feature extractors are trained for the task of phoneme classification (automatic speech recognition, ASR) and optimal DNNs for this task do not always implied optimal bottleneck features for the task of speaker recognition. This analysis is described in Chapter 4.

***A. Lozano-Diez**, R. Zazo-Candil, J. Gonzalez-Dominguez, D. T. Toledano and J. Gonzalez-Rodriguez, "An End-to-end Approach to Language Identification in Short Utterances using Convolutional Neural Networks", in Proc. of Interspeech 2015, Dresden, Germany, pp. 403-407, September 2015.*

An end-to-end DNN-based system for language recognition is presented in this work, in particular based on convolutional neural networks, previously successfully applied to other tasks such as computer vision. We focused on the task of short test utterances (about 3 seconds of speech) and obtained comparable results to an i-vector system used as reference while reducing the size of the model in terms of free parameters. The description of this system and results obtained for language recognition on a subset of eight languages (multi-class task) from the NIST LRE 2009 dataset are presented in Chapter 3.

***A. Lozano-Diez**, J. Gonzalez-Dominguez, R. Zazo, D. Ramos and J. Gonzalez-Rodriguez, "On the Use of Convolutional Neural Networks in Pairwise Language Recognition", in Proc. of IberSPEECH 2014, Advances in Speech and Language Technologies for Iberian Languages, Springer LNCS-8854, pp. 79-88, November 2014.*

Convolutional neural networks are used in this work for the case of challenging language-pair recognition (instead of the multi-class task addressed on the paper above), as end-to-end systems for classification between two languages. The proposed systems used convolutional neural networks as they provided a reduced model in terms of parameters with respect to other DNN approaches. The experiments conducted using this setup are presented in Chapter 3.

Other contributions related to the problems developed in this Thesis but not presented in this Dissertation include:

- PEER-REVIEWED INTERNATIONAL CONFERENCES

  D. Castan, M. McLaren, L. Ferrer, A. Lawson and ***A. Lozano-Diez****, "Improving Robustness of Speaker Recognition to New Conditions Using Unlabeled Data", in Proc. of Interspeech 2017, Stockholm (Sweden), August 2017.*

  In the context of the NIST SRE 2016, symmetric score normalization (Snorm) and calibration using unlabeled in-domain data were shown to be beneficial, but speaker labels are required for training calibration models, and therefore they can be estimated using clustering techniques when only unlabeled in-domain data was available. In this work, we evaluated these techniques in order to analyzed if they generalize for other frameworks apart from the NIST SRE 2016. The description of the systems submitted for the evaluation (a collaboration of SRI International in the U.S.A., CONICET-UBA in Argentina and Universidad Autonoma de Madrid in Spain) is also included in this work.

  A. Lozano-Diez*, I. Gomez-Piris, J. Franco-Pedroso, J. Gonzalez-Dominguez and J. Gonzalez-Rodriguez, "Speaker Clustering for Variability Subspace Estimation", in Proc. of IberSPEECH 2014, Las Palmas de Gran Canaria, Spain, November 2014.*

  In this work, we presented our approaches to speaker clustering in order to obtain an estimation of speaker identities in unlabeled datasets, further used to train Linear Discriminant Analysis (LDA) models for speaker verification. This was done on to of i-vectors from the NIST SRE 2012, following the protocol defined for the NIST i-vector speaker recognition challenge in 2014. These models trained on top of estimated labels did not show a significant loss in performance with respect to the same model trained on correct labels.

Chapter 2

Related Works in DNN-based Language and Speaker Recognition

TRADITIONALLY, LANGUAGE AND SPEAKER RECOGNITION FIELDS have progressed hand in hand since they are closely related and rely on many common underlying aspects, which allows these two approaches share many steps in their pipelines. Often, successful approaches in speaker recognition have been adapted to the language recognition field as it happened with the widely used GMM-UBM and i-vector approaches, and more recently, with deep learning algorithms.

In this chapter, we first introduce the language and speaker recognition tasks and review classical i-vector front-end, which has been and still is among state-of-the-art approaches for speaker and language recognition. Then, we introduce basic concepts of deep neural networks and some of their successful applications to the field of automatic speech recognition (ASR), which is closely related to our two areas of interest. ASR is also one of the main motivations of this Thesis due to the common speech processing stages with the tasks we address and the increasing interest in research and society in new speech driven technologies like Amazon Echo and Google Home devices. Finally, we review speaker and language recognition state-of-the-art systems based on DNNs, which run in parallel with the approaches proposed in this Thesis.

2.1. Introduction to Language and Speaker Recognition Tasks

The task of language recognition or language identification (LID) is defined as the task of identifying the language spoken in a given audio segment [Muthusamy *et al.*, 1994]. On the other hand, the speaker recognition (speaker detection or speaker verification) task consists of determining whether a specific speaker is speaking in a given utterance. Automatic systems aim to perform these tasks automatically, learning from a given dataset the necessary parameters to identify the language or speaker, respectively, contained in new spoken data.

Research in both fields has been driven to a large extent by the Language Recognition Evaluation (LRE) and Speaker Recognition Evaluation (SRE) series organized by NIST (U.S. National Institute of Standards and Technology) since 1996. These technology evaluations provide a common framework (making data available to all participants) to evaluate a given recognition task. Each participant sends blind results to the organization, which later provides comparative results, and final conclusions are shared during a workshop. Each evaluation differs in the specific tasks that participants have to address, such as dealing with different test duration, channel variability or noise conditions. Some of these evaluation frameworks (in particular, SRE 2010 for speaker and LRE 2009, 2015 and 2017) have been used in the experimental work of this Thesis.

Both types of technology have several applications. For example, the task of LID is useful for call centers to classify a call according to the language spoken, for speech processing systems that deal with multilingual inputs or for multimedia content indexing. In the case of speaker recognition systems, among the wide range of applications, we can highlight their use in security such as authentication using voice instead of other biometric traits, forensic scenarios or multimedia content classification by speakers.

Moreover, language and speaker recognition share important modules with many other systems from closely related research fields like speech recognition (whose aim is to transcribe audio segments), or in general, speech signal processing. Furthermore, not just the speech signal processing research area is involved, but also techniques from machine learning.

In fact, the successful application and adaptation of machine learning tools is one of the main lines of research in language and speaker recognition nowadays. The speech signal is a complex signal that contains a lot of information. Part of this information might be useful for the target task (for instance, language or speaker recognition), but there is also a large amount of information that automatic systems try to remove, which comes from different sources of variability. Furthermore, speech segments constitute temporal sequences which vary in duration, which poses a challenge for modeling the information and representation of this type of data. Generally speaking, there are several factors that make difficult disentangling the desired parts from the rest of unwanted information, and complex transformations such as those performed by deep neural networks and other machine learning algorithms seem perfectly suited to address the speech modeling.

2.2. GMM-UBM and i-vector Approaches to Language and Speaker Recognition

As first step in traditional speech, speaker and language recognition systems, the information in the input signal is processed to obtain some feature vectors which represent the signal in a frame-wise basis. Some of the most common features extracted have been acoustic features based on Mel-Filter banks, among which we can highlight Mel-Frequency Cepstral Coefficients (MFCCs) or Perceptual Linear Predictive (PLP) features.

These frame-wise feature vectors have been often modeled by a Gaussian Mixture Model

(GMM), normally adapted from an Universal Background Model (UBM) for robustness against data scarcity via Maximum a Posteriori (MAP), and the information of a given utterance or language/speaker model might be summarized then in a *supervector* of means of the Gaussian components of the model, which provides a fixed-length representation of the segment that simplifies posterior modeling. However, this *supervector* is embedded in a high-dimensional space: typically 1024 or 2048 Gaussian components were used, and about 20 MFCCs augmented with delta and double delta coefficients provided a 60-dimensional feature space, resulting in a *supervector* of more than 60k dimensions. This fact of high-dimensional vectors still posed a challenge for further modeling and comparison of utterance similarities or differences. In this way, Factor Analysis (FA) techniques started to be applied in speaker and language recognition, providing a channel subspace and a speaker/language variability subspace, where the information useful for the target class was represented by the latent factors of these two subspaces, which reduced the dimensionality of the representation. However, in order to properly estimate these two subspaces, large training datasets that gather as much variability as possible were required.

Thereby, Total Variability modeling emerged, putting together channel and speaker/language variability in just one subspace that comprises all kinds of variability sources, which required less training data to obtain a good estimation of the variability subspace.

Therefore, Total Variability and the corresponding i-vector approaches have been the state-of-the-art in speaker recognition for several years [Dehak *et al.*, 2011a]. Given the success this technique showed in that field, it was adapted and introduced in the language recognition research community [Martínez *et al.*, 2011], becoming the state-of-the-art in this area as well.

In a classical i-vector pipeline, we can split the system into different steps, common for both speaker and language recognition, usually up to the scoring stage:

- UBM-GMM modeling

  The first step of this approach consists in modeling the feature space with a Gaussian Mixture Model (GMM). This GMM is trained from feature vectors (as for example, MFCC or bottleneck features), with the Expectation Maximization (EM) algorithm, with data from a great number of utterances that belong to different languages or speakers. The resulting GMM is known as Universal Background Model (UBM), and is defined by its mean vector (μ, concatenation of mean vectors of each Gaussian component known as *supervector*) and its covariance matrix (Σ, covariance matrices of each Gaussian component).

- Statistics computation

  Given a trained UBM defined by its parameters $\lambda = \{\mu, \Sigma\}$, the next step is to compute the Baum-Welch statistics for a given utterance. These statistics represent each frame according to the GMM-UBM. Then, for each Gaussian component c, and each utterance frame u_t, the zero- and first-order sufficient statistics are obtained as follows:

$$N_c = \sum_t P(c|u_t, \lambda) \tag{2.1}$$

$$F_c = \sum_t P(c|u_t, \lambda)(u_t - \mu) \tag{2.2}$$

where $P(c|u_t, \lambda)$ is the posterior probability of component c generating the frame u_t.

- Total variability subspace training and i-vector extraction

  Generally speaking, the idea of the Total Variability (TV) approach is to project the supervector of means from a given utterance into a subspace T in which the variability (both channel and language or speaker) of a training dataset is represented [Dehak *et al.*, 2011a].

  The T projection matrix is trained via Expectation Maximization (EM), with a dataset which includes variability useful for the target task (language or speaker variability, for the two considered cases).

  Then, the total variability model can be represented as follows:

$$\mu_{UTT} = \mu_{UBM} + Tw \tag{2.3}$$

  where μ_{UTT} is the utterance-dependent supervector, μ_{UBM} is the UBM supervector of means (language or speaker-independent) and w is a latent variable, whose point estimated with Maximum A Posteriori (MAP) will be the i-vector representing each utterance.

  To extract the i-vector corresponding to a given utterance, the UBM is used to collect the Baum-Welch statistics from the utterance. Once these statistics and the T matrix are available, each i-vector can be extracted as shown in [Dehak *et al.*, 2011a].

  These i-vectors will have information of the language or speaker identity contained in the utterance they represent, together with information related to other sources of variability (channel variability). The information captured by the i-vectors will depend on the quality in terms of variability of the dataset for which the T matrix is trained.

- Classification / Verification (Scoring)

  Finally, once the i-vectors are computed, scoring step is performed.

  For the case of language recognition, one of the basic approaches is the cosine distance scoring, in which a score is extracted for each trial or comparison between test and train (language model) i-vectors and represents the cosine of the angle between them. Then, the higher the resulting score is, the higher is the probability to belong to the same class, since those data points are closer in the i-vector space. Moreover, some other simple generative backends such as Gaussian Linear Classifier (GLC) [Cumani *et al.*, 2015; Martínez *et al.*, 2011] showed their suitableness.

  In the field of speaker recognition or verification, some other scoring approaches in the i-vector subspace are usually applied to compensate the variability still existing in the i-vector such as Linear Discriminant Analysis (LDA) or Probabilistic Linear Discriminant Analysis (PLDA) [Prince and Elder, 2007]. Broadly speaking, PLDA aims to separate the speaker identity related information contained in the i-vector from any other information coming from variability sources (channel variability), which is similar to what factor analysis modeling does with the GMM *supervectors* but on top of i-vectors instead, providing a model for comparison (scoring) of pairs of i-vectors. However, these PLDA approaches were not as successful for language recognition, due to the projection into a (N-1)-dimensional space (where N is the number of languages involved in the task) with the consequent loss of information for LID, where the number of classes is much smaller than in the case of speaker recognition [Martínez *et al.*, 2011].

2.3. Basic Concepts of Neural Networks

Artificial Neural Networks or simply Neural Networks (NN), are machine learning algorithms based on a collection of units (neurons) organized in layers (hidden layers) and connected among them (weights) that provides a transformation of the input data in order to perform a given task such as classification. The so-called Deep Neural Networks (DNNs) consist of several (more than one in opposition to shallow architectures, which are composed of just one) hidden layers, which makes them able to extract features at multiple levels of abstraction and learn complex non-linear functions directly from the input data in order to minimize an error cost. A graphical example of a standard deep neural network can be seen in Figure 2.1.

This way, a feedforward DNN used to perform a classification task might have the following general structure: an input layer, which is fed with input vectors representing the data; two or more hidden layers (in opposition to shallow architectures, which had just one hidden layer), where a transformation is applied to the output of the previous layer, obtaining a higher level representation as we move away from the input layer; and an output layer, which computes the output of the DNN. In this last layer, the output is compared (for the case of supervised learning) to the reference label (true value) and the error criterion is applied to compute the cost.

The model is defined by its parameters: weight matrices $W_{j,j-1}$ and bias vectors b_j, with j going from 1 to the number of hidden layers. These parameters are adjusted iteratively to minimize a cost function, typically with stochastic gradient descent.

Thus, given a training set $(x^{(i)}, y^{(i)})$, where $x^{(i)}$ is a given feature vector, and $y^{(i)}$ its corresponding class (true value), each hidden layer applies a non-linear transformation function g to the output of the previous layer. This transformation takes into account the parameters W and b which relate one layer to its previous one, and provides the activation values of neurons with the following equations:

$$h_j(x^{(i)}) = g(W_{j,j-1}h_{j-1}(x^{(i)}) + b_j), \quad j = 2, ..., N-1 \tag{2.4}$$

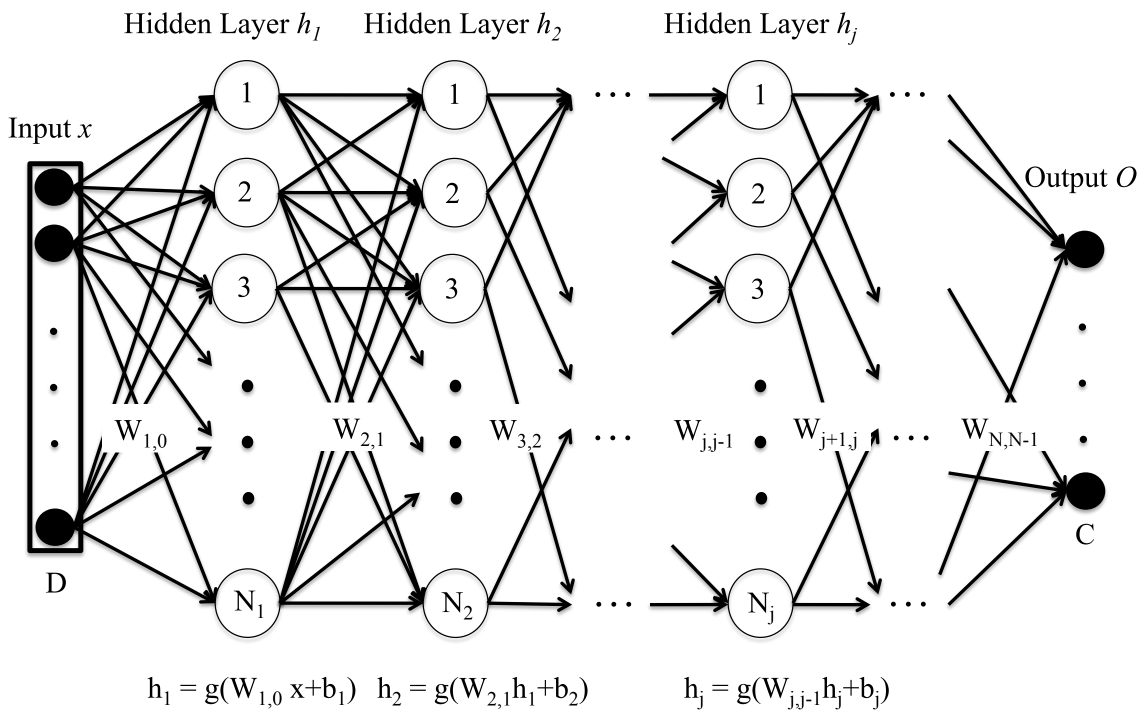


Figure 2.1: *This is a graphical representation of a standard feedforward DNN architecture. The DNN is fed with an input vector x of dimension D, which is transformed by the hidden layers h_j (composed of N_j hidden units) according to a function g (usually a non-linear transformation) and the parameters of the DNN (weights matrices W and bias vectors b). Finally, the output layer O provides the output of the DNN for the target task (e.g., for the case of classification, the probability of an input vector to belong to each class C).*

$$h_1(x^{(i)}) = g(W_{0,1}x^{(i)} + b_1) \tag{2.5}$$

Finally, for a classification task, the output layer computes a softmax function, which outputs the probability P of a given input x to belong to a certain class c:

$$P(c|h(x)) = \frac{\exp(W_l^c h_l(x) + b_l^c)}{\sum_{k=1}^{C} \exp(W_l^k h_l(x) + b_l^k)} \tag{2.6}$$

where $h_l(x)$ refers to the last hidden layer activation for input x, W_l^c and b_l^c denote the weights matrix and bias vector respectively, which connect the output unit for class c with the last hidden layer, and C is the total number of classes.

In order to adjust the parameters to the task, a cost function is considered, trying to minimize the error between the prediction (output by the network) and the true class, and parameters are modified step by step via backpropagation [Bishop, 2006].

In this Thesis Dissertation we focused on this type of DNNs, which belongs to the set of *supervised learning* algorithms which make use of labels in order to learn, as opposed to *unsupervised learning* techniques, which work without any prior knowledge about the labels of training data (also know as *clustering*). Among the different types of DNNs in the *supervised learning* set of algorithms, the general structure described in this section are usually referred to as *feed-forward* DNNs, which is the type of network used in Chapter 4 with a layer which aims to comprise the information learned by the DNN (*bottleneck layer*).

A variation of these feed-forward DNNs is the convolutional neural network [LeCun *et al.*, 2001], where some connections between units are shared and this way, they are able to extract the same features from different locations in the input matrix, among other things. This type of networks are explored for language recognition in Chapter 3.

We would like to highlight as well neural networks that allow connections between units in the same hidden layer, known as Recurrent Neural Networks (RNN), especially suitable to model temporal sequences. In particular, long short-term memory (LSTM) are a type of RNNs where each unit is replaced by a *cell* that can keep information (it has memory) and, thus, is able to take into account the information given by the context in an input sequence. A study of LSTM-based language identification systems is presented in Chapter 3.

2.4. DNN-based Systems for Speech Processing

Automatic Speech Recognition (ASR) aims to automatically obtain the transcription of a given speech segment. The ASR field has historically drawn the attention of researchers, and it is increasingly interesting motivated by the wide range of successful applications such as Apple's Siri, Amazon Echo or Google Home, which make use of speech to interact since it is the most natural way to communicate between humans.

Traditional technologies used to perform ASR have been based on the combination of Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) in order to model the acoustic and temporal information of the speech signal. Incremental solutions based on these HMM-GMM approaches made them difficult to be outperformed even by the first neural network (with just one hidden layer) based systems for many years [Bourlard and Morgan, 1993; Hinton *et al.*, 2012a]. However, advances both in algorithms and hardware, and the availability of large datasets to train properly DNNs with several hidden layers, made ASR systems experience a remarkable progress in the last decades.

One of the first successful applications was achieved with the use of DNN based acoustic models [Hinton *et al.*, 2012b], in which GMMs were replaced by DNNs in order to compute posterior probabilities of phonemes. Many other approaches based on DNNs showed impressive performance. For instance, in [Abdel-Hamid *et al.*, 2014], convolutional neural networks further improved this hybrid DNN-HMM approach for ASR; in [Hannun *et al.*, 2014] an end-to-end speech recognizer is developed based on recurrent neural networks, which is able to cope with noisy environments and reduces the dependency on hand-designed components of the pipeline; multilingual speech recognition in a real time end-to-end system for ASR is presented in [Gonzalez-Dominguez *et al.*, 2015]; and in [Bao *et al.*, 2013; Deng *et al.*, 2010; Grezl *et al.*, 2007], where neural networks are used as feature extractors (bottleneck features) for continuous speech recognition.

This tandem between machine learning and ASR allows progress in both fields as ASR provides a large-scale real application for machine learning researchers, which can test their algorithms properly and then gradually improve ASR systems performance [Deng and Li, 2013].

Moreover, ASR is closely related to the main lines of research explored in the experimental part of this Dissertation: language and speaker recognition. Thereby, improvements in ASR usually reflect in similar ideas in these other two areas, as it happened extensively with the use of bottleneck features (from DNNs trained for ASR) instead of traditional acoustic features.

2.5. DNN-based Approaches to Language and Speaker Recognition

Motivated by the outstanding results in ASR, DNNs were introduced in language and speaker recognition systems, following the same idea: replacing parts of the (or the whole) system with DNN-based models.

In this section, we review different approaches to the tasks of language and speaker recognition which include DNNs in different ways.

2.5.1. DNN-based Language Recognition

Different DNN-based strategies have been explored by language recognition researchers in recent years.

Deep neural networks have been also used as end-to-end systems for language identification, where the DNN performs classification from input features directly without i-vector modeling involved in the system. This use as classifiers for language recognition showed their success in [Lopez-Moreno *et al.*, 2014], and it has been explored in other research works [Gonzalez-Dominguez *et al.*, 2014; Montavon, 2009]. This approach is explored in Chapter 3 of this Dissertation.

However, many of the successful DNN approaches are based on the idea of replacing GMMs for modeling the acoustic features. Some of them use posteriors elicited by the DNN (trained for ASR) [Kenny *et al.*, 2014; Lei *et al.*, 2014a,b] instead of the ones obtained through a classical UBM-GMM. Also, some works use these posteriors to create feature vectors [Ferrer *et al.*, 2014, 2016], modeled then by different backends.

One of the most prominent uses of DNNs in today's language recognition systems are the bottleneck features, originally developed for speech recognition [Fontaine *et al.*, 1997; Grézl *et al.*, 2009a] but later very successfully applied also to language recognition [Fér *et al.*, 2015; Jiang *et al.*, 2014; Lozano-Diez *et al.*, 2017; Matějka *et al.*, 2014a; Richardson *et al.*, 2015a; Song *et al.*, 2013] , where they are the basis of state-of-the-art systems and where they gradually replaced traditional acoustic features like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptual Linear Prediction coefficients (PLP). This approach is explored in Chapter 4.

The fact that these feature vectors are extracted for every frame of an utterance (as well as senone posteriors are used in [Ferrer *et al.*, 2014]), producing a variable-length sequence, poses a challenge in modeling.

Some ideas to replace GMM and i-vectors in order to obtain an utterance level representation (usually referred to as *embedding*) have been applied for language recognition. For instance, outputs of hidden layers are averaged over time and stacked together in [Li *et al.*, 2016], forming a representation for the utterance in a high-dimensional space; or in [Pešán *et al.*, 2016], where one of the layers in the DNN averages frame-by-frame activations, and posteriors produced by the network are used for LID as end-to-end approach. Also, [Gelly and Gauvain, 2017] presents a DNN-based system which learns language dependent vectors with angular proximity loss function.

Motivated by all this, in Chapter 5 we present a DNN-based language recognition system with embeddings, following the approach used for speaker verification in [Snyder *et al.*, 2017]. In particular, we use bidirectional long short-term memory (BLSTM) recurrent layers in order to exploit the temporal information of the signal (whose frames are represented by bottleneck features), before the sequence summarizing layer. Then, we pool together the mean and standard deviation statistics from the output of this frame-by-frame part of the DNN (over all frames of an input sequence). The pooled mean and standard deviation are then forwarded through two additional fully connected hidden layers, whose outputs will be used to extract the utterance level embeddings. Finally, a softmax layer is used as output layer, and the network is trained with multi-class cross-entropy objective to discriminate between languages.

Unlike [Pešán *et al.*, 2016], we used the extracted embeddings (instead of posteriors) to train a generative model for classification (GLC). Our architecture is also simpler than the one presented in [Li *et al.*, 2016], with smaller embedding layers, and yields better results.

We compare the performance of the system based on embeddings with a corresponding i-vector system that is trained using the same features. We report the performance on the NIST LRE 2015 and achieve comparable results which suggests that systems based on embeddings are a viable approach to be explored for LID. Finally, we further improve the results of the strong baseline by means of a score-level fusion which suggests that the DNN modeling has been able to extract complementary information out of the same bottleneck features.

2.5.2. DNN-based Speaker Recognition

Evolution of research in speaker recognition has usually progressed in parallel with language recognition, introducing similar techniques, and adapting the approaches from one area to the other. However, the application of DNNs as end-to-end systems for speaker recognition is not as straightforward as in language identification [Lei *et al.*, 2014b], since the task is usually speaker verification (determine whether two utterances are spoken by the same speaker) and thus, the amount of training data for each speaker is much more reduced, and speakers identities used for DNN training are usually different from the target ones.

Thereby, some of the first approaches to speaker recognition based on DNNs used the network to replace the GMM-UBM for the statistics estimation [Garcia-Romero *et al.*, 2014; Lei *et al.*, 2014b], using the DNN output posteriors instead of the UBM to compute frame alignments. In [Lei *et al.*, 2014b], features used for frame alignments can be different than the ones used for statistics estimation, allowing flexibility in optimal features selection [McLaren *et al.*, 2015].

Also, as it happened with language recognition systems, DNN used as bottleneck feature extractors were explored for speaker recognition [Garcia-Romero and McCree, 2015; Lozano-Diez *et al.*, 2016; Yaman *et al.*, 2012], in the same fashion, using a DNN trained for ASR. Performance of standalone bottleneck features in this type of systems is usually improved by the use of joint acoustic (MFCCs, PLPs) and bottleneck features in the i-vector/PLDA pipeline [Yaman *et al.*, 2012]. This approach is further explored in Chapter 4.

Recently, the use of DNNs have been explored in text-dependent speaker recognition to replace both GMM and i-vectors in order to obtain an utterance level representation *embedding*. The embedding is obtained by computing the mean over the framewise outputs of one or more layers in the DNN in [Variani *et al.*, 2014] or by the use of a recurrent neural network in [Heigold *et al.*, 2016]. A fairly simple architecture was also developed for text-independent speaker verification in [Snyder *et al.*, 2017], where embeddings are obtained as outputs of two hidden layers (after pooling mean and standard deviation over time). The DNN in the mentioned work is trained for speaker classification with a given set of speakers, and subsequent modeling of embeddings with Probabilistic Linear Discriminant Analysis [Prince, 2007] (PLDA) achieves a performance comparable to i-vectors. The adaptation of this setup to language recognition is explored in Chapter 5.

Finally, despite the challenging scenario that speaker verification implies, some successful end-to-end DNN-based approaches for speaker verification have been explored in text-dependent frameworks [Heigold *et al.*, 2016; Variani *et al.*, 2014], but still using a DNN previously trained for speaker classification in order to extract embeddings (*d-vectors*) and performing the backend with another DNN that outputs the scoring for each trial. Some more recent works are showing promising results for text-independent speaker recognition in this line [Rohdin *et al.*, 2017; Snyder *et al.*, 2016], pushing research towards this challenging task.

Chapter 3

DNN as a Classifier for Language Recognition

THE USE OF DEEP NEURAL NETWORKS FOR LANGUAGE RECOGNITION has appeared in a wide range of forms. Many successful applications include DNNs in their pipeline, replacing parts of it. Some of these approaches will be reviewed and explored in the following chapters, meanwhile in this chapter, we present end-to-end DNN approaches understood as systems where the DNN performs modeling and classification stages and therefore, are meant to obtain a representation of the signal and a good classifier as a whole for the task they are trained to perform.

Thus, in this chapter, we introduce the main ideas and background about this end-to-end use of DNNs for the task of language recognition, followed by our contributions in this line by means of convolutional deep neural networks (CDNNs) and long short-term memory (LSTM) recurrent neural networks (RNN) architectures, which will be described as well in each corresponding section.

3.1. Introduction to End-to-end DNN Language Recognition

Deep neural networks are known in many tasks of signal processing by their ability of achieving a good representation of the input signal at different abstraction levels, but also by being good classifiers. Thus, we will refer as an *end-to-end* DNN-based system to the system that takes some inputs and performs the target task without any other backend afterwards. The DNN is trained as a whole to perform a target task such as classification, where it would output the probability of each input to belong to each target class.

These end-to-end approaches have proven to be very successful in language recognition [Lopez-Moreno *et al.*, 2014], outperforming the i-vector scheme that has been the state-of-the-art approach to language recognition for several years. An example of a generic architecture of end-to-end DNN-based language recognition system is depicted in Figure 3.1.

Generally for LID, this type of systems usually takes some input features (normally including

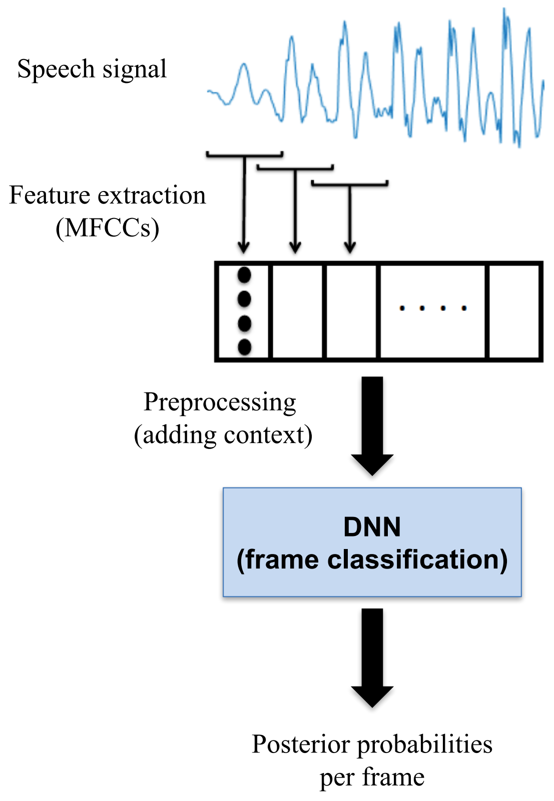

Speech signal
Feature extraction (MFCCs)
Preprocessing (adding context)
DNN (frame classification)
Posterior probabilities per frame

Figure 3.1: *Generic scheme of an end-to-end language recognition system based on deep neural networks. This is a graphical representation of a generic scheme of a DNN used as end-to-end system for language recognition. The DNN is fed with an input vector, which is usually composed of a feature vector for a frame concatenated with some context frames, and this is then transformed by the DNN to obtain a vector of scores per frame that contains the posterior probabilities for each of the languages involved in the setup.*

some context), and it is trained to classify each input frame into one of the target languages involved in the given dataset. Thus, once the DNN is trained, the scoring step consists of forwarding each input frame through the network, which outputs a vector of values usually indicating the probability that the given frame belongs to each class or language.

These DNN-based systems have noticeably outperformed i-vectors, especially when dealing with short test utterances [Lopez-Moreno *et al.*, 2014]. However, they present some drawbacks: DNN-based systems need huge training datasets in order to be successful [Lopez-Moreno *et al.*, 2014] and have a large number of parameters to be trained (and thus, their training is computationally expensive). Then, in Section 3.2 we propose the use of convolutional DNNs since their specific architecture reduces drastically the number of parameters to tune. Furthermore, in general, DNN-based approaches rely on stacking several acoustic frames as an input in order to model longer time context than a frame [Lopez-Moreno *et al.*, 2014]. Thus, in Section 3.3 we explore long short-term memory (LSTM) recurrent neural networks, a more proper model when copying with time depending sequences [Mikolov *et al.*, 2011], which have shown their success in the field [Gonzalez-Dominguez *et al.*, 2014] with a reduced number of parameters as well.

3.2. Convolutional DNN for Language Recognition

In this section, we present an end-to-end approach to the language identification (LID) problem based on Convolutional Deep Neural Networks (CDNNs). The use of CDNNs is mainly motivated by the ability they have shown when modeling speech signals [Abdel-Hamid *et al.*, 2014; McLaren *et al.*, 2014], and their relatively low-cost with respect to other deep architectures in terms of number of free parameters thanks to their structure based on sharing weights among hidden units in order to extract the same features from different locations.

This type of networks was already applied to language recognition in [Lei *et al.*, 2014a], where a CDNN trained for automatic speech recognition was used to replace the UBM in an i-vector based approach. However, our CDNN-based systems are trained to discriminate among a set of given languages and, thus, there is no need of a previous speech recognition stage, providing an end-to-end scheme.

In particular, in this section, we explore CDNN systems for the problem of language recognition over two different tasks extracted from the NIST LRE 2009. First, we applied them to pair-wise language recognition between similar languages, which has been one of the tasks addressed for several NIST LREs. Furthermore, we explore this approach using a balanced subset of 8 languages within the NIST LRE 2009. Both sections will be focused on the task of short test durations (segments up to 3 seconds of speech).

The proposed CDNN-based systems achieve comparable performances to our reference systems, while reducing drastically the number of parameters to tune (at least 100 times fewer parameters), and a simple fusion at score level outperforms our best standalone system for most of the cases.

3.2.1. Convolutional DNN System Description

The general scheme of our convolutional DNN language recognition system is depicted in Figure 3.2.

Convolutional neural networks are a type of neural network where each hidden layer is usually split into two parts: convolutional layer and subsampling layer [LeCun *et al.*, 2001]. The convolutional layer aims to perform feature extraction. Each unit in this layer is connected to a local subset of units in the hidden layer below, according to a given filter shape. It computes its activation by convolving the input with a linear filter (weights), adding a bias term and applying a non-linear transformation (in our case, tanh).

Moreover, groups of these units spatially related share their parameters and form what is called a *feature map*, since their objective is then to extract the same features from different locations in the input. This sharing of parameters also decreases the number of free parameters of the whole network.

On the other hand, the subsampling layer reduces the size of the representations obtained by the previous convolutional layer. In our case, this phase is based on partitioning the input into non-overlapping regions (according to a given pool shape) and choosing the maximum

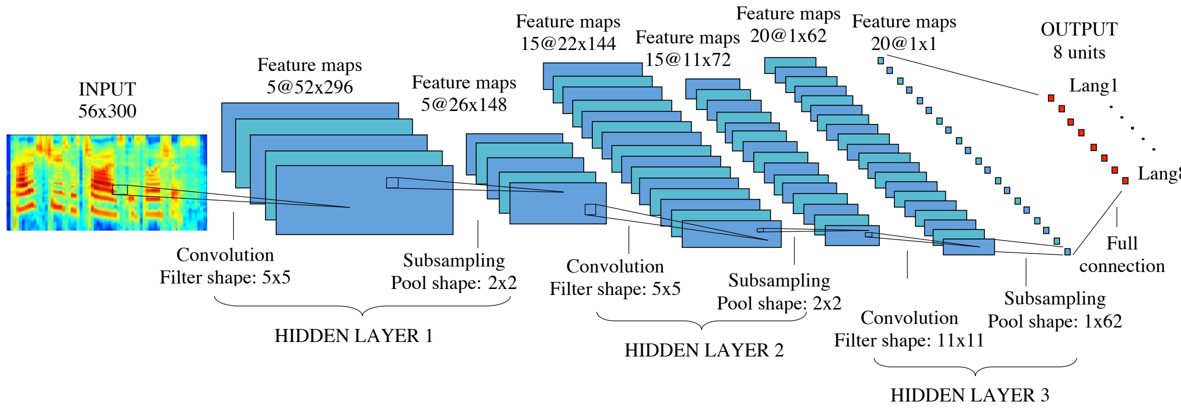


Figure 3.2: *Representation of a convolutional deep neural network architecture used in the experimental part of this section. It consists of three hidden layers of 5, 15 and 20 filters respectively, and aims to discriminate among 8 languages. Other models have the same structure but varying the input features, the number of filters in each layer and the number of output units to adjust to the specific task.*

activation of each region (*max-pooling*). This subsampling also makes the network invariant to small translations and rotations [Bengio, 2009].

All these aspects make convolutional networks easier to train than other DNNs, by using well-known supervised algorithms such as gradient descent [LeCun *et al.*, 2001]. Moreover, they are also smaller in terms of number of free parameters than the i-vector systems used typically for the problem of language identification.

3.2.2. Analysis on Language-pairs

In this section, we apply the CDNN approach for language recognition described in the previous Section 3.2.1 to the specific task of pair-wise language recognition. Thus, the proposed systems are evaluated on challenging pairs of languages selected from the NIST LRE 2009 dataset. Results are compared with two spectral systems based on Factor Analysis and Total Variability (i-vector) strategies, respectively. Moreover, a simple fusion of the developed approaches and the reference systems is performed. Some individual and fusion systems outperform the reference systems, obtaining up to approximately 17% of relative improvement in terms of $minC_{DET}$ for one of the challenging pairs.

3.2.2.1. CDNN-based System Description

The details of the CDNN-based system are as follows. The input of the network consists of a 2-dimensional *time-frequency* representation of the speech signal. In our case, 23 Mel-scale filter-bank outputs have been used to feed the network for each segment of 3 seconds of speech, normalized to have zero mean and unit variance for each coefficient over the whole training set. Those 3 seconds correspond with 300 frames, since windows of 20 ms of duration have been applied with 10 ms of overlap. Moreover, in order to suppress silences, a voice activity detector

Conf. Parameter	Model 1	Model 2
# Layers	3	3
# Filters/layer	[12, 12, 12]	[20, 50, 30]
Filter shapes	[(5, 5), (5, 5), (2, 2)]	[(5, 5), (5, 5), (2, 2)]
Pool shapes	[(2, 2), (2, 2), (1, 71)]	[(2, 2), (2, 2), (1, 71)]

Table 3.1: *Configuration parameters for the developed models.*

based on energy has been used. This last filtering process makes usually test segments contain less than 3 seconds of actual speech, fact that we had to cope with to obtain a fixed shape for the input of the neural network as it is expected in our architecture. Thereby, our approach to deal with this was simply applying a *right padding* by taking the first frames and replicating them at the end of the segment to fit this length requirement.

Depending on the configuration of the network, two different models have been considered. Both of them have 3 hidden *convolutional-maxpooling* layers. Each of these layers are composed of two stages: 1) computation of the activation for each hidden unit in each feature map by convolving the input with a linear filter (*weights*), adding a bias term and applying the non-linear transformation *tanh* ($h = tanh(W * x + b)$); and 2) application of a sub-sampling phase based on partitioning the input into non-overlapping regions and choosing the maximum activation of each region. For both models, the shape of the linear filters is 5×5 for the first two hidden layers, and 2×2 for the third one. Regarding the max-pooling regions, they have a shape of 2×2 in the first two hidden layers, and 1×71 in the third one in order to have a single value as output of the last hidden layer. Then, the difference between the two mentioned models relies on the number of filters or *feature maps* considered for each hidden layer, which is related to the idea of how many different features we want the network to extract in each layer. The first model (referred to as *Model 1*) has 12 filters in each layer and the second one (referred to as *Model 2*), has 20, 50 and 30 in each of the three mentioned hidden layers, respectively. All this information is summarized in Table 3.1.

As far as the output layer is concerned, it consists of a *fully-connected* layer that computes a $softmax$ function according to the following expression:

$$P(Y = i|x, W, b) = softmax_i(Wx + b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}}$$

where i is a certain class, and W and b are the parameters of the model (weights and bias, respectively).

The output value is considered as a score or likelihood measure of belonging to a certain language, between the two languages involved, since the experiments conducted in this section are based on *language-pairs*. The final score for a test segment is computed as the difference between the logarithms of each likelihood.

Regarding the training of the network, the algorithm that has been used is the stochastic gradient descent with a learning rate of 0.1 and based on *minibatches* of 500 samples each one. The cost function that the algorithm tries to optimize (minimize in this case) is the negative log-likelihood, defined as follows:

$$NLL(\theta, D) = -\sum_{i=1}^{|D|} \log P(Y = y^{(i)}|x^{(i)}, \theta)$$

where D is the dataset, θ represents the parameters of the model ($\theta = W, b$, weights and bias respectively), $x^{(i)}$ is an example, $y^{(i)}$ is the label corresponding to example $x^{(i)}$, and P is defined as the output of the $softmax$ function defined above.

Also, an *early stopping* technique has been used during the training in order to reduce the possibility of *overfiting*, so the performance of the model is evaluated in a validation set, and if the improvements over that set are not considered relevant, the training of the network is stopped.

All this development has been done by using Python and, specifically, *Theano* [Bergstra *et al.*, 2010], following the ideas of [LISA].

3.2.2.2. Reference Systems: FA-GMM and i-vector

In order to have a baseline to compare with, two different systems have been taken as reference and have been evaluated on the same datasets that the proposed method based on CDNNs.

The first one consists of a Factor Analysis GMM Linear Scoring (FA-GMM-LS) [Gonzalez-Dominguez *et al.*, 2010b], which is a GMM system with linear scoring and session variability compensation applied in the statistic domain. The speech signal is represented by a parameterization consisting of seven MFCCs with CMN-Rasta-Warping concatenated to 7-1-3-7 SDC-MFCCs. Two Universal Background Models (UBMs) with 1024 Gaussian components were trained. One of them (UBM_{CTS}) was trained with Conversational Telephone Speech (hereafter, CTS). The other one (UBM_{VOA}) was train with data from VOA (Voice of America radio broadcasts through Internet), provided by NIST. Thereby, two different systems were developed, one for each UBM. Two session variability subspaces matrices were obtained (U_{CTS} and U_{VOA}). The subspaces were initialized with PCA (Principal Component Analysis) based on [Kenny *et al.*, 2005; Vogt and Sridharan, 2008], taking into account just top-50 eigenchannels, and trained by using the EM algorithm.

The second reference system, the i-vector system, is based on GMMs where a Total Variability modeling strategy [Dehak *et al.*, 2009] is employed in order to model both language and session variability. Unlike FA, a *total space* represented by a low-rank T matrix jointly includes language and session variability. Moreover, a session variability compensation stage is applied directly to the low dimensional space driven by T by means of Linear Discriminant Analysis (LDA) and Within-Class Covariance Normalization (WCCN) [Gonzalez-Dominguez *et al.*, 2010a].

The speech signal is represented as in the first reference system and the T matrix has been trained with CTS and broadcast data as well.

Both systems output a score for each test segment computed as the difference between the scores given for each of the two language models involved in each pair.

Moreover, as scores from reference and CDNN-based systems are in the same domain (real numbers), a simple sum fusion has been performed to check if both approaches are able to extract complementary information from the same input features.

3.2.2.3. Dataset Description

The database used to perform the experiments in this section has been that provided by NIST for the LRE 2009 [NIST, 2009].

The NIST LRE 2009 database includes data coming from different audio sources: conversational telephone speech (CTS), used in previous evaluations, and broadcast data that contain telephone and non-telephone speech. That broadcast data consist of two corpora from past Voice of America (VOA) broadcast in multiple languages (VOA2 and VOA3). Some language labels of VOA2 might be erroneous since they have not been audited, as it is mentioned by the organizers in the evaluation plan [NIST, 2009], where more details can be found as well.

Regarding evaluation data, segments of 3, 10 and 30 second of duration from CTS and broadcast speech data are available to test the developed systems. However, the experiments shown in this section are only based on segments of 3 seconds (*short duration*).

We have selected five challenging pairs of languages for the experiments presented in this section: Bosnian-Croatian (BC), Farsi-Dari (FD), Hindi-Urdu (HU), Portuguese-Spanish (PS) and Russian-Ukrainian (RU). These pairs are among the proposed tasks of the language-pair evaluation in the NIST LRE 2009, since they are considered of particular interest due to their similarities. Indeed, all of them except Portuguese-Spanish are considered mutually intelligible.

The available datasets have been split into three disjoint subsets: training, validation and test. The first two datasets include just broadcast data (VOA2 and VOA3) from the development data provided by the organizers for this evaluation. However, test segments come from CTS and VOA datasets and are the actual evaluation data of NIST LRE 2009. The specific amount of data (in hours) per language used in the experiments is shown in Table 3.2.

3.2.2.4. Performance Metrics

The performance of the systems has been evaluated according to the cost measure (C_{DET}) defined in the NIST LRE 2009 evaluation plan [NIST, 2009]. This metric takes into account the false alarm and false rejection probabilities and the cost of a bad classification of the segment of speech. As this measure shows the cost with the optimal threshold, it corresponds with the minimum cost operating point, so we will refer to it as $minC_{DET}$ [Van Leeuwen and Brummer, 2006].

Furthermore, Detection Error Tradeoff (DET) curves have been used in order to evaluate the performance of the systems in different operating points. In the legend of the DET curves shown in Section 3.2.3.5 the Equal Error Rate (EER), in %, is also shown.

	Amount of data (# Hours)		
	Training	**Validation**	**Test**
Bosnian	12.27	5.26	0.28
Croatian	9.16	3.92	0.29
Dari	25	10.72	1.07
Farsi	25	10.72	0.28
Hindi	25.9	11.10	0.51
Portuguese	11.79	5.05	0.32
Russian	20.27	8.69	0.66
Spanish	13.85	5.94	0.31
Ukrainian	15.89	6.81	0.31
Urdu	26.63	11.41	0.29

Table 3.2: *Amount of data used per language (in hours) for the experiments on the selected language-pairs from NIST LRE 2009.*

As it has been already mentioned, apart from the performance evaluation of the individual systems considered in this set of experiments, the performance of fusion systems has been also included. Those fusion schemes consist of a score level fusion where both mentioned reference systems (FA-GMM and i-vector) and the corresponding CDNN-based model are involved. A simple sum of the scores elicited by each system involved in the fusion scheme has been used to obtain the final score for a certain segment of speech.

3.2.2.5. Experiments and Results

The experiments shown in this section are based on the five challenging language pairs mentioned in Section 3.2.2.3. For each of these pairs, two different models (according to the configurations shown in Table 3.1) and the two reference systems described in Section 3.2.2.2 have been evaluated on the same test samples. Furthermore, the amount of data used for training the CDNN-based system (see Table 3.2) is approximately the same that the used for training the reference systems, although some languages datasets have been reduced in order to partially compensate the big differences between the amount of hours per language in the training datasets for each pair of languages involved in each experiment.

The performance of each individual system can be seen in the left side of Table 3.3. According to these results, CDNN-based models outperform the best reference systems in the case of Hindi-Urdu, with a relative improvement of ∼14% in terms of $minC_{DET}$. As it is shown in the right side of Table 3.3, by performing a simple sum-fusion of the reference systems and the CDNNs systems, the relative improvement yields up to $\sim$ 17% for the Hindi-Urdu pair. For the Bosnian-Croatian experiment, the fusion system gives $\sim$ 7% of relative improvement, and the performances of all individual systems are pretty similar for this language-pair.

	Individual Systems				**Fusion Systems**	
	Reference		**CDNNs**		**Ref. Systems +**	**Ref. Systems +**
	FA-GMM	**i-vector**	**Model 1**	**Model 2**	**Model 1**	**Model 2**
BC	34.45	37.24	34.89	37.76	**32.13**	35.48
FD	**33.81**	45.51	49.92	49.88	49.46	49.79
HU	43.30	41.93	36.09	37.72	**35.16**	36.90
PS	11.51	**9.15**	17.08	14.71	10.07	9.53
RU	35.29	**35.06**	45.69	44.53	41.72	42.50

Table 3.3: *Performance of individual (left) and fusion (right) systems ($minC_{DET} \times 100$) in language-pairs from NIST LRE 2009.*

By way of contrast, the models obtained for the case of the language-pairs Farsi-Dari, Portuguese-Spanish and Russian-Ukrainian, even the fusion ones, worsen results than those yielded by the reference systems. Possible reasons might be that the configuration parameters used are not adequate for the available data or that the development dataset has not been adequately selected (with little variability among utterances).

Regarding the comparison between the two CDNN-models, although *Model 2* has more filters (*feature maps*) and, thereby, its capability to extract a better abstract representation of the input signal is bigger, just in three pairs it gives better results than *Model 1*. This might be caused by a lack of data or variability within them that leads to the problem of *overfitting*. More evidence of occurrence of that problem is that we have observed a big *gap* between validation and test errors.

Finally, Figure 3.3 shows the DET curves obtained for each language-pair, comparing the performance at different operating points of both reference systems, the best CDNN system and the best fusion model. As it was observed with the $minC_{DET}$ performance measure, our individual approach outperforms the reference systems in the experiments with Hindi-Urdu, and the fusion one, in the Bosnian-Croatian pair. Relative improvements and general behavior of the systems are similar to the ones observed with $minC_{DET}$ measure.

3.2.3. Analysis on Multiple Languages

In this section, we evaluate different configurations of CDNN-based language recognition systems by for the task of identification between multiple languages instead of pairs. In particular, we selected a balanced subset of 8 languages within the NIST LRE 2009 Voice of America (VOA) dataset, for the task of short test durations (segments up to 3 seconds of speech).

The proposed CDNN-based systems achieve comparable performance to our baseline i-vector system, while reducing drastically the number of parameters to tune (at least 100 times fewer parameters than the i-vector model). Then, we combine these CDNN-based systems and the i-vector baseline with a simple fusion at score level. This combination outperforms our best

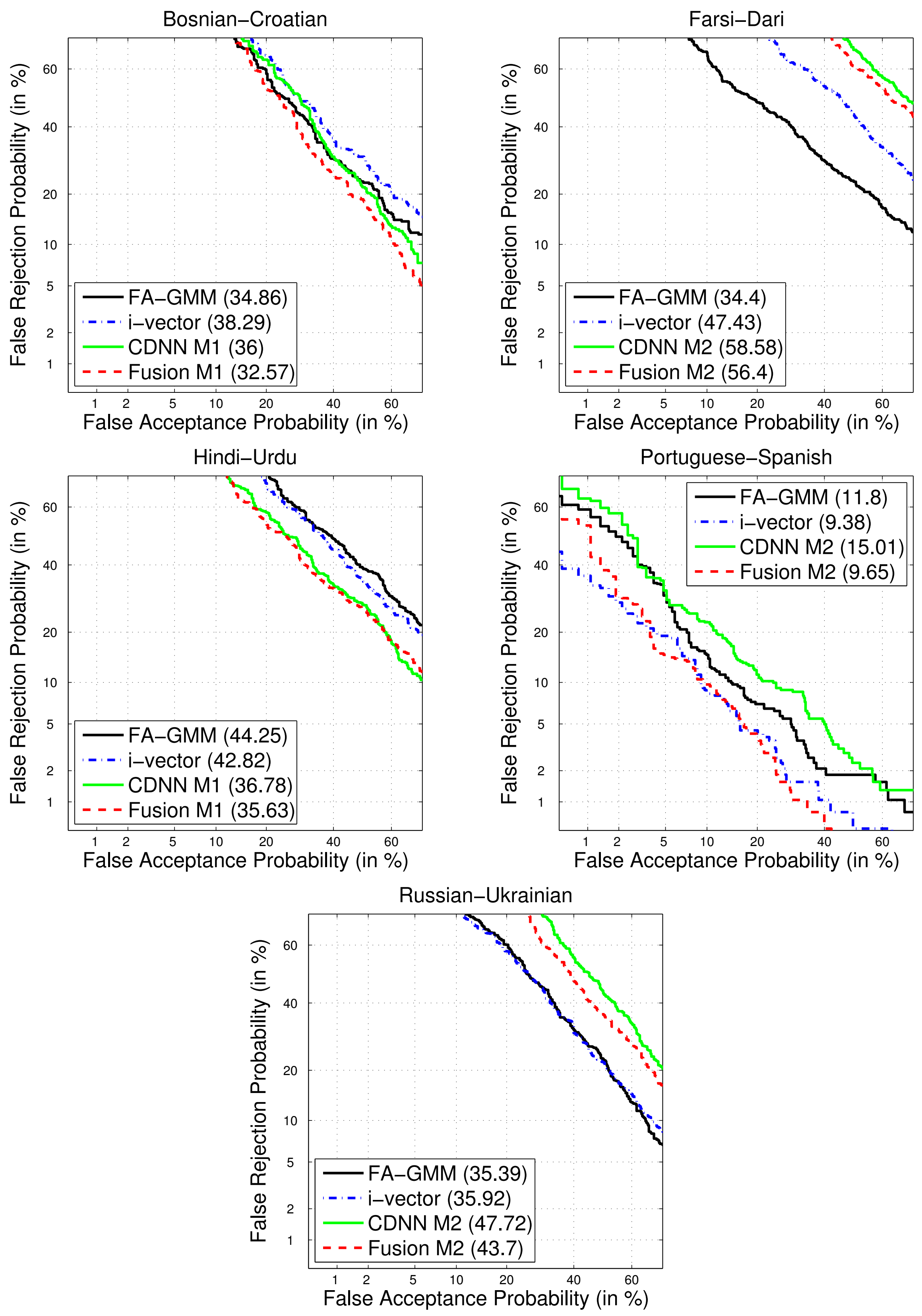


Figure 3.3: *DET curves corresponding to reference systems, the best CDNN system and the best fusion according to the EER for each language pair. The EER (in %) is shown in brackets.*

standalone system (up to 11% of relative improvement in terms of EER$_{avg}$).

3.2.3.1. CDNN-based System Description

For this set of experiments, we used again speech segments of 3 seconds, which correspond with 300 frames, with windows of 20 ms of duration and 50% of overlap. For this set of experiments, we represented each frame with a vector of 56 MFCC-SDCs (Shifted Delta Coefficients) [Torres-Carrasquillo *et al.*, 2002], with the configuration 7-1-3-7. These vectors are created by stacking delta cepstral coefficients computed across multiple speech frames. In particular, we use 7 MFCCs, with one frame advance and delay for the delta computation, and we stack 7 blocks with a time shift of 3 frames between them, which results in 56-dimensional feature vectors representing each frame of the utterance. Thus, we fed the network with the resulting 2-dimensional matrix of dimensions 56×300, which corresponds with a given speech segment of 3 seconds long. Finally, we normalized the input to have zero mean and unit variance for each coefficient over the whole training set. Moreover, in order to suppress silences, we used a voice activity detector based on energy. As it was explained in Section 3.2.1, when discarding non-speech frames, final test segments contain usually less than 3 seconds of actual speech, which did not allow for a fixed dimensional input matrix as it is expected by the network. To solve that, we applied a *right padding* by using the first frames of the segment to fit this requirement.

We built different networks depending on the number of filters (feature maps) considered for each hidden layer, which is related to the idea of how many different features are to be extracted in each layer. However, we used for all of them an architecture consisting in 3 hidden layers that perform the two stages described in Section 3.2.1: convolution and subsampling. All the architectures used in this section have also in common the shape of the linear filters (5×5 for the first two hidden layers, and 11×11 for the third one) and the max-pooling regions (with a shape of 2×2 in the first two hidden layers, and 1×62 in the third one in order to have a single value as output of the last hidden layer).

We conducted experiments to evaluate different amounts of data to train each network in order to study its influence in the performance, considering training sets of 178 h, 356 h or 534 h depending on the experiment. These training sets are balanced to contain approximately the same number of hours for each language involved in the given experiment.

The differences among the structures are summarized in Table 3.4.

The output layer consists of a fully-connected layer that computes a softmax function as it was defined in Section 3.2.2.1. Then, the network outputs the probability that the test segment belongs to a certain language, among the languages involved in the experiment.

For training, we used the stochastic gradient descent algorithm to minimize the negative log-likelihood with a learning rate of 0.1 and based on minibatches of 500 samples. We used *early stopping* to finish training when performance did not improve enough over the validation set. We used the same tools as for the case of language-pairs described in Section 3.2.2.1, implementing our systems with the *Theano* library [Bergstra *et al.*, 2010] from Python, following the ideas of [LISA].

	Configuration	**Development Data**	
ID	# Filters/Layer	Train	Validation
ConvNet 1	[20, 30, 50]	∼178h	∼31h
ConvNet 2	[5, 15, 20]	∼178h	∼31h
ConvNet 3	[5, 15, 20]	∼356h	∼63h
ConvNet 4	[5, 15, 20]	∼534h	∼63h
ConvNet 5	[10, 20, 30]	∼356h	∼63h
ConvNet 6	[10, 20, 30]	∼534h	∼63h

Table 3.4: *Configuration parameters for the developed models.*

3.2.3.2. Reference i-vector System

In order to have a baseline to compare with, an i-vector based system was evaluated on the same test dataset.

The i-vector system is based on GMMs where a Total Variability (TV) modeling strategy is employed in order to model both language and session variability [Dehak *et al.*, 2011b]. First, an Universal Background Model (UBM) composed of 1024 Gaussian components is trained from MFCC-SDC parameterization of the audio, with the configuration 7-1-3-7. Then, Baum-Welch statistics are computed over this UBM, and a TV space of 400 dimension is derived from them by using PCA followed by 10 EM iterations. All the process, from the parameterization of the audio to the i-vector computation has been done using Kaldi [Povey *et al.*, 2011a].

Regarding the classification stage, we used the classical cosine scoring scheme. Thus, given a test utterance i-vector w, and the i-vector model w_L (computed as the mean i-vector from all the utterances of the language L), the cosine similarity is computed as follows:

$$S_{w,w_L} = \frac{\langle w, w_L \rangle}{||w||||w_L||}$$

The classical Linear Discriminant Analysis (LDA) classification scheme gave us slightly lower performance than just the cosine scoring scheme in our experiments. This could be explained because we just used 8 languages in our experiments. Thus, if we used LDA, just 7 dimensions would remain from the i-vectors, and we could be losing information useful for discrimination.

3.2.3.3. Dataset Description

The database used to perform the experiments was that provided by NIST in the Language Recognition Evaluation 2009 (NIST LRE 2009).

As it was described in Section 3.2.2.3, the NIST LRE 2009 database includes data coming from different audio sources: conversational telephone speech (CTS), used in previous evaluations, and broadcast data containing telephone and non-telephone speech. That broadcast data consist of two corpora from Voice of America (VOA) broadcast in multiple languages (VOA2 and

VOA3). Some language labels of VOA2 might be erroneous since they have not been audited. More details can be found in [NIST, 2009].

Both language and audio source labels were distributed to participants. In this section, just data belonging to VOA were considered in order to avoid unbalanced data from different sources (CTS and VOA).

The whole database includes data from 40 languages (23 target and 17 out of set). From them, we selected 8 languages (as in [Gonzalez-Dominguez *et al.*, 2014]) for which up to 200 hours of audio are available: US English (Eng), Spanish (Spa), Dari (Dar), French (Fre), Pashto (Pas), Russian (Rus), Urdu (Urd) and Chinese Mandarin (Chi).

As in the previous experiments on language-pairs, for evaluation we focused on short duration task, selecting the test segments of 3 seconds of speech, where i-vector systems obtain lower performances. Our test dataset includes then 2942 test segments from the 8 languages mentioned before. Thus, we perform a closed-set task, without out of set test utterances.

3.2.3.4. Performance Metrics

The performance of the systems was evaluated according to two different metrics.

The first one is the average cost measure C_{avg}, as defined in the NIST LRE 2009 evaluation plan [NIST, 2009]. This measure takes into account the false alarm and false rejection probabilities and the cost of a bad classification of the speech segment. Therefore, it evaluates the ability of the system for discrimination and calibration (i.e., the capacity of setting optimal thresholds).

Secondly, the classical Equal Error Rate (EER, in %) was considered. As we deal with a multi-class task in this section, we compute the EER for each individual system, and average them to obtain an EER_{avg} as a metric for the performance of the whole system.

Furthermore, we present the confusion matrix of our best system, typically used when assessing the performance in a multi-class classification task. With this matrix, we show the discriminative capacity of the system and the confusion among all the languages involved in our experiments.

3.2.3.5. Experiments and Results

The experiments presented in this section are based on the 8 languages mentioned in Section 3.2.3.3. For the experiments based on convolutional DNNs (CDNNs), we split the development data into two disjoint datasets (training and validation) in order to perform training and model selection with different data. The amount of data used for each CDNN-based system can be seen in Table 3.4.

The performance of standalone systems and combined systems are summarized in Table 3.5. In this table we can also see the size (in terms of number of parameters to be trained) of each system.

In order to calibrate and combine the systems, we use multi-class logistic regression from FoCal toolkit [Brümmer, 2007], which parameters were estimated by using the evaluation scores

		Performance	
ID	Size	EER$_{avg}$ (%)	C_{avg}
i-vector	∼23M	**16.94**	**0.1535**
ConvNet 1	∼198k	22.14	0.2406
ConvNet 2	∼39k	25.90	0.2700
ConvNet 3	∼39k	24.69	0.2616
ConvNet 4	∼39k	23.48	0.2461
ConvNet 5	∼78k	21.60	0.2282
ConvNet 6	∼78k	21.11	0.2293
AllConvNets	-	17.93	**0.1836**
ConvNet 6+i-vector	-	**15.96**	**0.1433**
AllConvNets+i-vector	-	**15.04**	**0.1360**

Table 3.5: *Individual and combined systems performance of the experiments on the eight languages subset from NIST LRE 2009.*

themselves.

- Individual Systems

  As we can see in Table 3.5, the performance of the i-vector system is better than the one obtained by the standalone CDNN-based systems. However, the size of these models is between ∼100 and ∼600 times smaller with respect to the number of parameters that need to be tuned in the i-vector system. In our case, the baseline i-vector system presented in Section 3.2.3.2 has ∼23M of parameters, which is given by the number of Gaussian components of the UBM (1024), the feature space dimensionality (56 MFCC-SDCs) and the i-vector dimensions (400). In contrast, our biggest CDNN-based model is ∼100 times smaller (∼198k parameters). Moreover, the datasets used to train the CDNN systems are actually smaller than the one composed of 200 h per language that we used to train the i-vector system. Therefore, the CDNN systems are able to extract useful discriminant information even with less data and much less parameters. If we compare the different CDNN-based systems, we can see that the more data we introduce, the better the performance (see Figure 3.4).

  Furthermore, increasing the number of filters per layer (and, thus, the size of the model) yields better performance even if the amount of data used to train remains constant (compare ConvNet 3 and 5 or ConvNet 4 and 6 in Table 3.4).

  It should be highlighted that increasing the number of parameters or the amount of data means higher cost in terms of time and memory needed to train the CDNN-based systems, although it is slightly noticeable in the testing stage. Nevertheless, their size is much smaller than the i-vector approach and they need less resources to be stored and tested.

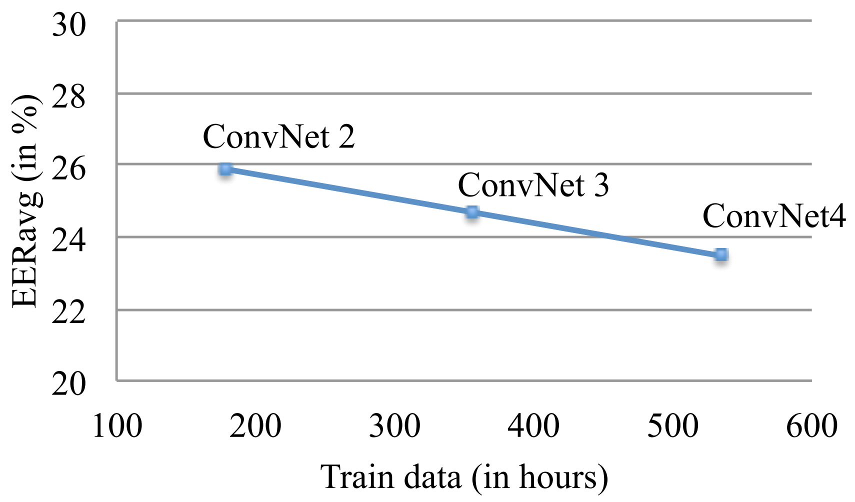


Figure 3.4: *Influence of the amount of data used for training in the performance of the system (in terms of EER$_{avg}$). Note that the three convolutional networks shown in this graphic have the same topology (ConvNet 2, 3 and 4, with [5, 15, 20] filters per layer).*

- Fusion Systems

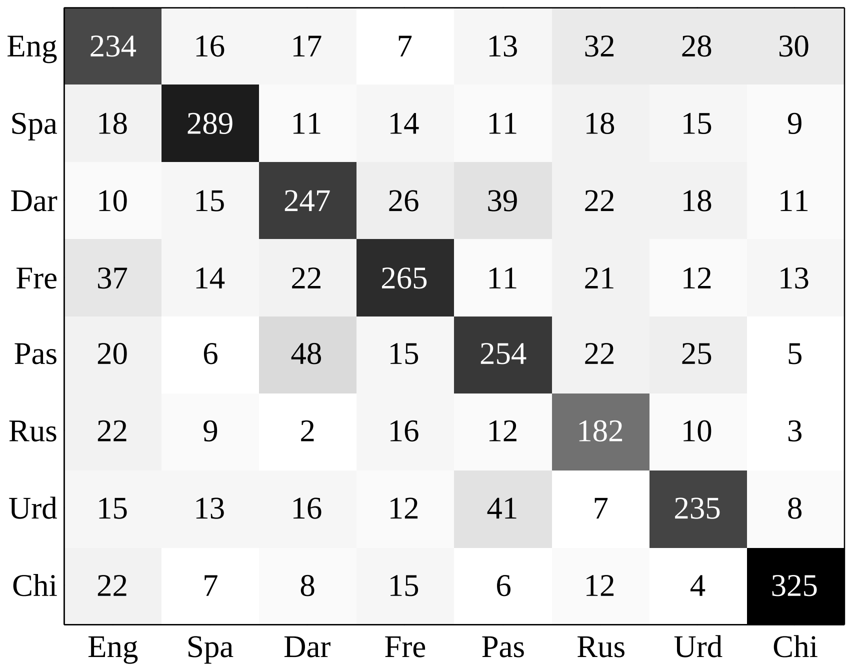
  The first kind of fusion we present in this section is the combination of the CDNN models. As it is shown in Table 3.5 (see AllConvNets row), with this fusion we obtain comparable performance to i-vector system (17.93% of EER$_{avg}$ versus 16.94%), and a ∼15% of relative improvement (in EER$_{avg}$) with respect to the best standalone CDNN system (ConvNet 6). From this result, we can draw the conclusion that even using the same type of architecture, varying just the number of filters per layer and the training dataset, CDNNs are able to extract complementary information.

  On the other hand, when fusing the best CDNN system with the i-vector system, the combination outperforms our baseline by ∼6% in terms of EER$_{avg}$. Moreover, when fusing all the CDNN models with the baseline i-vector system, this relative improvement reaches up to a 11%. The confusion matrix for this last combination can be seen in Figure 3.5.

3.3. LSTM RNN for Language Identification

Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNNs) have recently outperformed other state-of-the-art approaches, such as i-vector and Deep Neural Networks (DNNs), in automatic Language Identification (LID), particularly when dealing with very short utterances ($\leq$3 s).

In this section, our contribution is to present an open-source, end-to-end, LSTM RNN language recognition system which runs on limited computational resources (a single GPU), comparing it against a classical i-vector system. We analyze this end-to-end approach on different environments based on data from two different NIST Language Recognition Evaluations (LRE).



	Eng	Spa	Dar	Fre	Pas	Rus	Urd	Chi
Eng	234	16	17	7	13	32	28	30
Spa	18	289	11	14	11	18	15	9
Dar	10	15	247	26	39	22	18	11
Fre	37	14	22	265	11	21	12	13
Pas	20	6	48	15	254	22	25	5
Rus	22	9	2	16	12	182	10	3
Urd	15	13	16	12	41	7	235	8
Chi	22	7	8	15	6	12	4	325

Figure 3.5: *Confusion matrix corresponding to the fusion of all CDNN-based systems and our baseline i-vector system.*

First, we train and test our system on a balanced and controlled environment selected from a subset of the NIST LRE 2009, where the LSTM RNN LID system clearly outperforms our reference i-vector system, especially when the system has to cope with short test utterances. This result is in line with previously published research using proprietary LSTM implementations and huge computational resources, which made these former results hardly reproducible. Further, we extend those previous experiments modeling unseen languages (out of set, OOS, modeling), which is crucial in real applications. Results show that an LSTM RNN with OOS modeling is able to detect these languages and generalizes robustly to unseen OOS languages. Furthermore, we also analyze the effect of even more limited test data (from 2.25 s to 0.1 s) proving that with as little as 0.5 s an accuracy of over 50% can be achieved.

Moreover, we test this approach on the development and evaluation data of the NIST LRE 2015. Here, results show that our deep learning approach based on LSTM RNNs is more sensitive to unbalanced datasets, channel variability and, especially, to the mismatch between development and test datasets.

3.3.1. LSTM RNN System Description

As we have already mentioned, while DNN-based approaches have proven to perform great in a variety of scenarios, they rely on stacking several acoustic frames as an input in order to model longer time context than a frame [Lopez-Moreno *et al.*, 2014]. On the other hand, recurrent neural networks (RNNs), a special type of DNNs where connections between units form directed

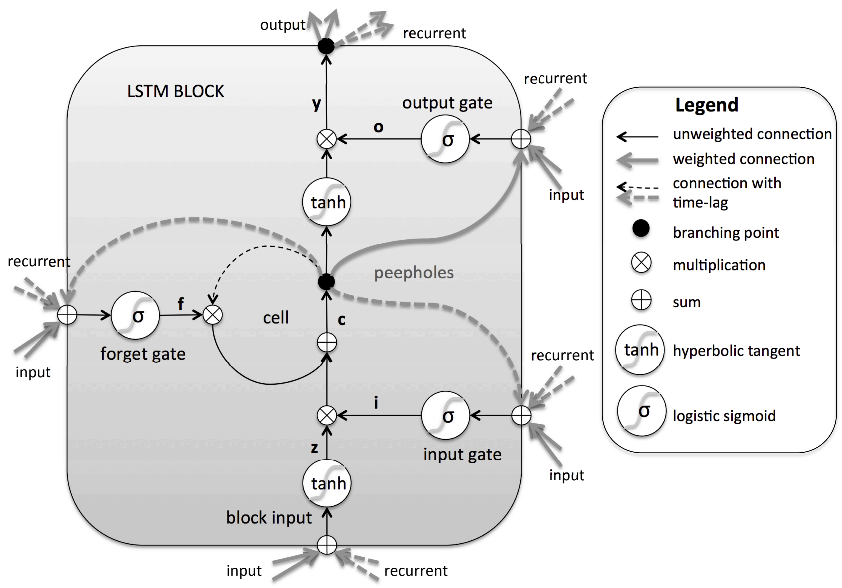


Figure 3.6: *Long Short-Term Memory recurrent neural network architecture. A single memory block is shown for clarity. The output goes to every unit in the next layer. The recurrent output goes to this memory block and every other memory block in this layer. All inputs and recurrent inputs shown are the same signals (same input goes to the memory block and to the three gates). Adapted from [Greff* et al.*, 2015].*

cycles, seem to be a more proper model when coping with time depending sequences [Mikolov *et al.*, 2011]. Even though RNNs are, in theory, a good model to fit temporal sequences such as speech, its training process has issues that makes its performance not as good as expected [Bengio *et al.*, 2013; Pascanu *et al.*, 2013].

Long Short-Term Memory (LSTM) recurrent neural networks (RNNs) have the ability to store information from previous inputs during long time periods [Gers *et al.*, 2000, 2003; Graves, 2012], which makes them much more suitable to model sequential data than deep feed forward neural networks. LSTM RNNs have recently been shown to outperform the state-of-the-art DNN systems in tasks involving time-depending signals including acoustic modeling in large vocabulary speech recognition [Graves *et al.*, 2013] or handwriting recognition [Frinken *et al.*, 2012].

A LID system based on LSTM RNNs with an outstanding performance is presented in [Gonzalez-Dominguez *et al.*, 2014], where the solution implemented runs over a large machine infrastructure and includes a proprietary LSTM RNNs implementation. This fact makes its use hardly reproducible or simply inaccessible for many research groups. Thus, we adapt that approach in order to build an efficient system using open source software and explore different aspects that affect LSTM RNNs performance, implementing several configurations.

The underlying idea of a LSTM neural network is to replace the hidden units in a traditional DNN with *memory blocks* [Greff *et al.*, 2015], like the one is depicted in Figure 3.6.

These blocks can be seen as a memory chip in a digital computer, where each one contains one or more memory cells recurrently connected and three multiplicative units (the input, output and forget gates) that work similar to the write, read and reset signals in that chip. More precisely, the input and output gates control respectively the flow of input activations into the memory cell and the output flow of cell activations into the rest of the network. The forget gate allows the flow of information from the memory block to the cell as an additive input, therefore adaptively forgetting or resetting the cell's memory. These features make them easier to train properly than conventional RNNs. The input and output gates help solving the vanishing error problem in the traditional RNN [Bengio *et al.*, 2013]: in the absence of a new input or error signals to the cell, the local error remains constant. In addition, the forget gate allows the network to have an adaptive and limited memory buffer avoiding infinite loops. The LSTM architecture described in [Gers *et al.*, 2003] also has the ability to learn precise timing of the outputs using *peephole* connections. These connections allow communication between gates of the same memory block, outperforming the traditional architecture specially when precise timing of the outputs is important.

3.3.1.1. Architecture

A schematic of a unit of the LSTM RNN used in this section can be seen in Figure 3.6. It features a single memory cell, three gates (input, forget and output), an output activation function and peephole connections. The output of the block is recurrently connected to the block input and all of the gates.

The vector formulas for a LSTM layer forward pass are given below. More details and the corresponding Back Propagation Through Time (BPTT) formulae can be found in [Graves and Schmidhuber, 2005]. Here x^t is the input vector at time t, the W are rectangular input weight matrices, the R are square recurrent weight matrices, the p are peepholes weight vectors and b are bias vectors. σ is the logistic sigmoid function, $tanh$ is the hyperbolic tangent activation function and $\odot$ is the element-wise product of the vectors.

$$z^t = tanh(W_z x^t + R_z y^{t-1} + b_z) \qquad \text{block input} \tag{3.1}$$
$$i^t = \sigma(W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i) \qquad \text{input gate} \tag{3.2}$$
$$f^t = \sigma(W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f) \qquad \text{forget gate} \tag{3.3}$$
$$c^t = i^t \odot z^t + f^t \odot c^{t-1} \qquad \text{cell state} \tag{3.4}$$
$$o^t = \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o) \qquad \text{output gate} \tag{3.5}$$
$$y^t = o^t \odot tanh(c^t) \qquad \text{block output} \tag{3.6}$$

The output layer is then configured as a *softmax*, where hidden units map input x_j to a class probability p_j in the form

$$p_j = \frac{exp(x_j)}{\sum_l exp(x_l)} \tag{3.7}$$

where l is an index over all the classes.

As a cost function for backpropagating gradients in the training stage, we use the cross-entropy function defined as

$$C = -\sum_j t_j \log p_j \tag{3.8}$$

where t_j represents the target probability of the class j for the current evaluated example, taking a value of either 1 (true class) or 0 (other class).

3.3.1.2. System Description

In all our experiments the training dataset is split into random chunks of 2 seconds from which MFCC-SDC (Shifted Delta Coefficients) with the configuration 7-1-3-7 are computed using Kaldi [Povey *et al.*, 2011a]. The network is fed with these MFCC-SDC with no stacking of acoustic frames: a single MFCC-SDC is given as an input at each time step.

Our system consists of one or two hidden layers followed by an output layer and are implemented using CURRENNT [Weninger *et al.*, 2014] running over a single GPU. The hidden layers are uni-directional LSTM layers with forget gates and peepholes while the output layer is a softmax layer with the same number of units as languages we have in our experiments. The softmax layer utilizes a cross entropy error function for the back propagation and returns a probability for each input frame and language.

In order to deal with unbalanced data and make the training process faster we train each iteration with a different subset of the training data, which consists of picking random chunks of 2 seconds until we have about 6 hours of audio per language.

The memory blocks in the LSTM hidden layers store the temporal state of the network which changes with the input to the neural network at each time step. When the system gives a probability for a given frame of belonging to one of the languages as an output, it relies not only on the frame input but on every previous frame in that sequence or file. Therefore, the last outputs are computed when the system has information of almost the whole file so they are the most reliable. For scoring, we compute an utterance level score for each target language by averaging the log of the softmax output for that language but taking into account just the last frame scores for every file (details are given in the first part of Section 3.3.3.3).

Finally, multi-class linear logistic regression calibration using FoCal Multiclass toolkit [Brümmer, 2007] was applied to the outputs of every neural network and the reference i-vector system. Moreover, to analyze whether the information learned by the reference and proposed systems is complementary, linear logistic regression fusion of the two individual systems described (LSTM and i-vector) was performed and evaluated.

3.3.2. Reference i-vector System Description

The i-vector system follows again the standard procedure described in [Dehak *et al.*, 2011b]. It is based on an Universal Background Model (UBM) consisting of 1024 Gaussian components, trained on the same MFCC-SDC with the configuration 7-1-3-7 used for the LSTM-based system. Then, we derive a Total Variability (TV) subspace of 400 dimensions using PCA followed by 10 EM iterations. Both MFCC-SDC and TV matrix were obtained using Kaldi [Povey *et al.*, 2011a].

Having at maximum 8 different classes in our experiments, a standard classification scheme based on Linear Discriminant Analysis (LDA) would have projected our data into a space with 7 or less dimensions, loosing relevant information for LID. Therefore Cosine Distance scoring without LDA has been used for this task. Thus, the similarity or a given test utterance i-vector and the mean i-vector of the language is given by the cosine score as defined in the Section 3.2.3.2.

The total number of parameters of the i-vector system accounts for the TV matrix. It is given by NxFxD, being N, F and D the number of Gaussians components (1024), the feature dimension (56) and the i-vector dimensions (400). In our model, this makes a total of $\sim$23M of parameters.

3.3.3. Analysis on NIST LRE 2009

In this section, we explore the performance of our proposed LSTM-based language recognition system on a subset of the NIST LRE 2009 database that consists of a controlled environment in which the amount of data per language is balanced and large. We evaluate our systems in different test conditions, considering short utterances from different conditions (broadcast and conversational speech) and for the task with and without out of set (OOS) languages.

3.3.3.1. Datasets Description

As we have mentioned before, the NIST LRE 2009 consists of a set of audio recordings from Conversational Telephone Speech (CTS), used in the previous evaluations for both development and test purposes, consisting of spontaneous telephone conversations; and broadcast news data from "Voice of America" (VOA), obtained via an automatic acquisition system where telephone and non-telephone speech are mixed. All raw data containing radio broadcast speech, with the corresponding language and audio source labels were distributed to participants, comprising a total of 40 languages (23 target and 17 out of set).

All the **training** data considered in this set of experiments belongs to VOA in order to avoid unbalanced mix of CTS and VOA. Further, we have selected 8 representative languages for which up to 200 hours of audio are available, as it was already done in [Gonzalez-Dominguez *et al.*, 2014] (and in previous experiments of Section 3.2.3), in an effort to avoid the disparity on training data for every language (from $\sim$10 to $\sim$950 hours). We have used all the other target languages in NIST LRE 2009 to build a train set with $\sim$200 hours in order to train an out of set class. Thus, we differentiate 3 different sets of languages in our experiments:

- Target languages: US English (eng), Spanish (spa), Dari (dar), French (fre), Pashto (pas), Russian (rus), Urdu (urd) and Chinese Mandarin (chi).

Target languages	**Trained_OOS**	**Real_OOS**
US_English (eng)	Amharic	Arabic
Spanish (spa)	Bosnian	Azerbaijani
Dari (dar)	Cantonese	Belorussian
French (fre)	Haitian Creole	Bengali
Pashto (pas)	Croatian	Bulgarian
Russian (rus)	Indian English	Unknown English
Urdu (urd)	Farsi	Italian
Mandarin Chinese (chi)	Georgian	Japanese
	Hausa	Punjabi
	Hindi	Romanian
	Korean	Shanghai-wu
	Portuguese	Southern-min
	Turkish	Swahili
	Ukrainian	Tagalog
	Vietnamese	Thai
		Tibetan
		Uzbek

Table 3.6: *List of all the different language sets from NIST LRE 2009 considered in our experiments for LSTM systems for language recognition.*

- Trained out of set languages (trained_OOS): This set contains all the other target languages considered in NIST LRE 2009 (more details given in Table 3.6).
- Real out of set languages (real_OOS): This set contains all the out of set languages considered in NIST LRE 2009 (more details given in Table 3.6).

Our test sets are the following:

- **main_test_set:** consists of the trials from the NIST LRE 2009 3 s condition evaluation set belonging to VOA for the 8 target languages, yielding a total of 2942 test segments and 23536 trials.
- **trained_OOS_test_set:** formed by the main test set plus all the VOA trials of the 3 s condition in the NIST LRE 2009 of the trained_OOS languages (see Table 3.6) leading to 8931 (where 5989 of them are OOS) test segments and a total of 80379 trials.
- **real_OOS_test_set:** constrains the main test set plus all the VOA trials of the 3 s condition in the NIST LRE 2009 of the real_OOS languages (see Table 3.6) yielding to 6437 test segments (where 3495 of them are OOS) test segments and a total of 57933 trials.
- **cts_test_set:** consists of the trials from the NIST LRE 2009 3 s condition evaluation set

belonging to CTS. From the 8 target languages only 4 have CTS test files (chi, rus, eng and urd) yielding a total of 1320 test segments and 10560 trials.

In addition, we wanted to evaluate the performance of our system when even shorter test utterances are considered. Therefore, we have selected all the files in our main test set with more than 2.25 seconds of speech according to our VAD yielding to a subset of 2100 files. Then, we have cut these recordings to build different duration subsets ranging from 0.1 to 2.25 seconds of speech.

3.3.3.2. Evaluation Metrics

Two different metrics were used in order to assess the performance of our systems:

- Accuracy. The percentage of correctly identified trials when making hard decisions (by selecting the top scored language).
- EER$_{avg}$, the mean of the Equal Error Rate computed language by language which is an extended metric in the community.

For the sake of clarity we do not deal with the problem of setting optimal thresholds (calibration). Therefore, C_{avg}, the metric used in the NIST LRE 2009 evaluation is not used in this section.

3.3.3.3. Experiments and Results

1. Discarding initial frame scores

   In uni-directional, left-to-right LSTMs, such as the one used in our experiments, an output is based on previous and present inputs in the sequence. Therefore, the last output scores are the most reliable. Figure 3.7 shows how the average performance of 5 different architectures varies with the percentage of initial frame scores discarded (we will describe the different architectures later in this section, just relative improvement matters now). Selecting (and averaging) just the last 10% of the output frame scores leads to improved robustness of the utterance level score. Thus, from now on, the results will be shown when 90% of the initial frame scores are discarded.

2. System performance (Results on the 3 seconds VOA test set)

   In [Gonzalez-Dominguez *et al.*, 2014] it was shown that LSTM RNNs are a good approach to exploit useful temporal information for LID with proprietary implementation. Motivated by those results, in this study we explore different architectures and configurations using open-source code.

   Table 3.7 summarizes the performance of 5 LSTM RNNs systems in terms of EER$_{avg}$ and accuracy. We highlight first that 4 out of the 5 proposed architectures for the LSTM RNN system outperform the reference i-vector based system in EER$_{avg}$. This fact is particularly

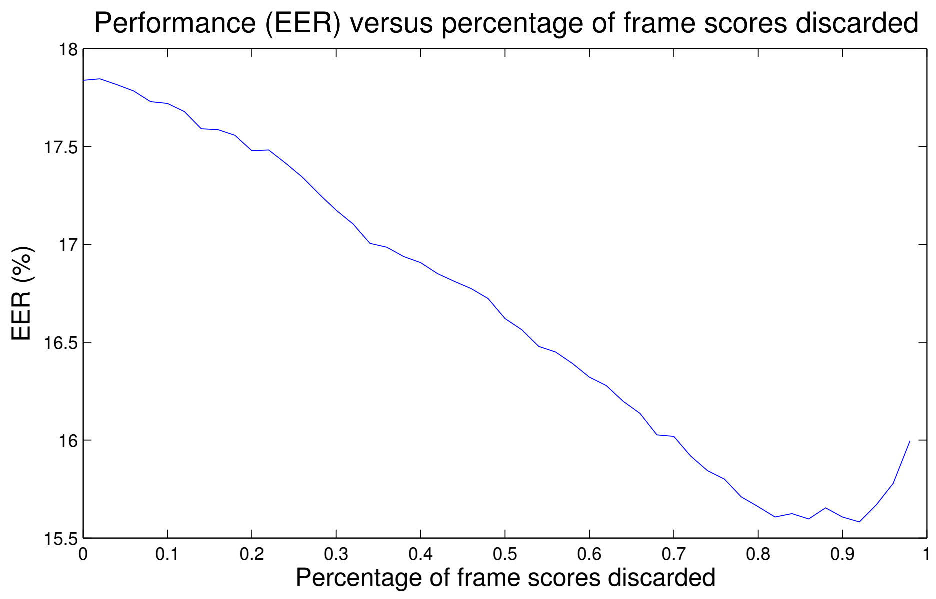


Figure 3.7: *In RNN based systems, the output score is computed based on previous and present inputs, being the last outputs the most reliable scores. Discarding the less reliable scores may lead to a better performance. In this figure we show the average performance (EER_{avg}) of 5 LSTM systems versus percentage of initial frame scores discarded.*

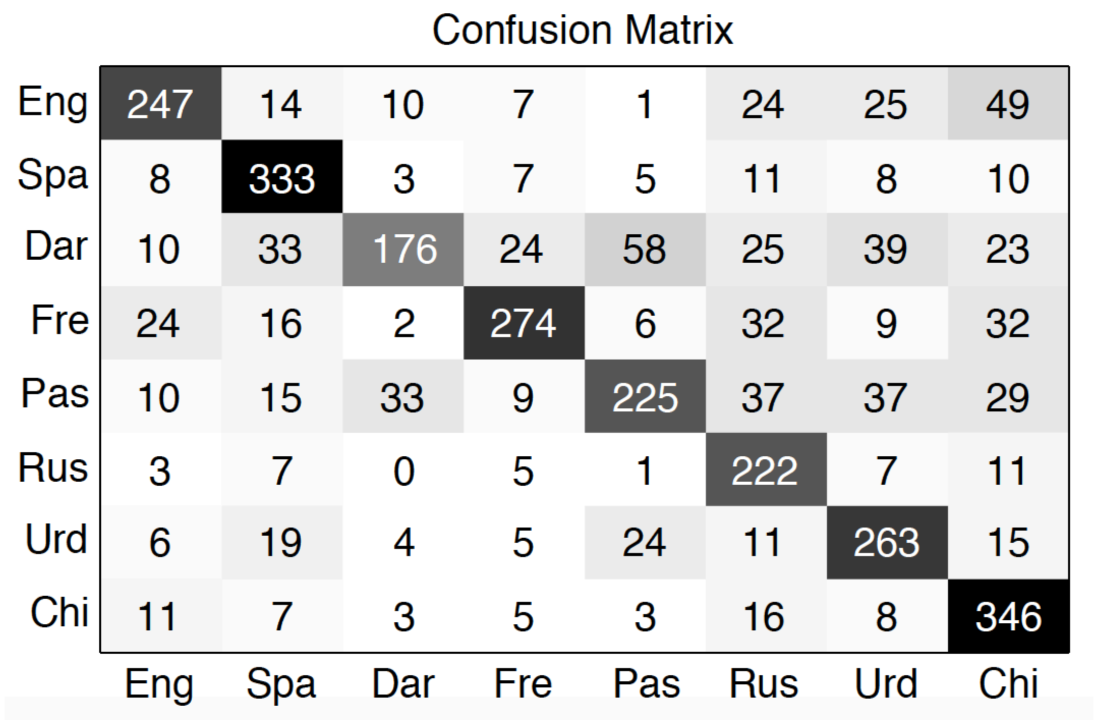
Architecture		**Complexity**			**Performance (%)**		
ID		Size	Time/Iter	#Iters	Acc.	EER_{avg}	Improvement
#1	reference i-vector	∼23M			65.02	16.94	-
#2	lstm_1_layer_512_units	∼1.2M	∼25min	∼125	57.51	17.82	-
#3	lstm_1_layer_750_units	∼2.5M	∼45min	∼300	63.39	15.61	∼7.85
#4	lstm_1_layer_1024_units	∼4.4M	∼75min	∼350	65.63	15.10	∼10.86
#5	lstm_2_layer_256_units	∼850k	∼20min	∼500	63.73	14.96	∼11.69
#6	lstm_2_layer_512_units	∼3.3M	∼60min	∼400	**70.90**	**12.51**	**∼26.15**

Table 3.7: *System performance on the 8 target languages subset of NIST LRE 2009 (3s test segments). The last column stands for the improvement in terms of EER_{avg} with respect to the reference system.*

interesting taking into account that the proposed architectures have from 5 to 21 times fewer parameters (see *Size* in Table 3.7) than the reference system.

Table 3.7 also shows the computational complexity of the different topologies in terms of time per training iteration and the approximate number of iterations necessary until convergence. Even though the number of iterations depends on many factors (initialization, learning rate, etc.) we can observe that both the time per iteration and the number of iterations increase with the total size of the model.

While EER is widely used in the field of LID, in this experiment we are facing a multi-class



	Eng	Spa	Dar	Fre	Pas	Rus	Urd	Chi
Eng	247	14	10	7	1	24	25	49
Spa	8	333	3	7	5	11	8	10
Dar	10	33	176	24	58	25	39	23
Fre	24	16	2	274	6	32	9	32
Pas	10	15	33	9	225	37	37	29
Rus	3	7	0	5	1	222	7	11
Urd	6	19	4	5	24	11	263	15
Chi	11	7	3	5	3	16	8	346

Figure 3.8: *Best system confusion matrix. Confusion matrix of the best LSTM RNN single system, lstm_2_layer_512_units (system #6), on the 8 target languages subset of NIST LRE 2009 (3 s test segments). Ground truth is represented in the Y axis while the predicted language is represented in the X axis.*

classification problem. In order to analyze both the confusion and the discrimination performance of the systems considering all the languages pairs, Figure 3.8 shows the confusion matrix of the best single system, #6 in Table 3.7. In this confusion matrix we can see that our system predicts correctly most of the test segments but there are still some frequent confusions as Dari - Pashto.

3. Out of set performance

In this set of results we show the performance of LSTM RNN based systems for language identification in presence of out of set (OOS) languages. In order to face this problem we have trained an additional OOS class with a mixture of audio coming from 15 different languages (see Table 3.6 for more details). We have selected three systems for comparison; the reference i-vector based system, the single LSTM system with best performance so far (512 units per layer and 2 layers), and the same system with a single layer (512 units per layer and 1 layer) to test this task. Results are summarized in Table 3.8.

We highlight three major results. First, when dealing with the main test set (no out of set test data) the performance of the out of set (OOS) LSTM RNN system (the system

	EER$_{avg}$(%)		
	main_test_set	trained_OOS_test_set	real_OOS_test_set
reference i-vector based system	16.94	21.60	21.87
lstm_1_layer_512_units	17.82	-	-
lstm_2_layer_512_units	**12.51**	-	-
OOS_lstm_1_layer_512_units	18.14	23.57	23.75
OOS_lstm_2_layer_512_units	14.83	**20.84**	**21.10**

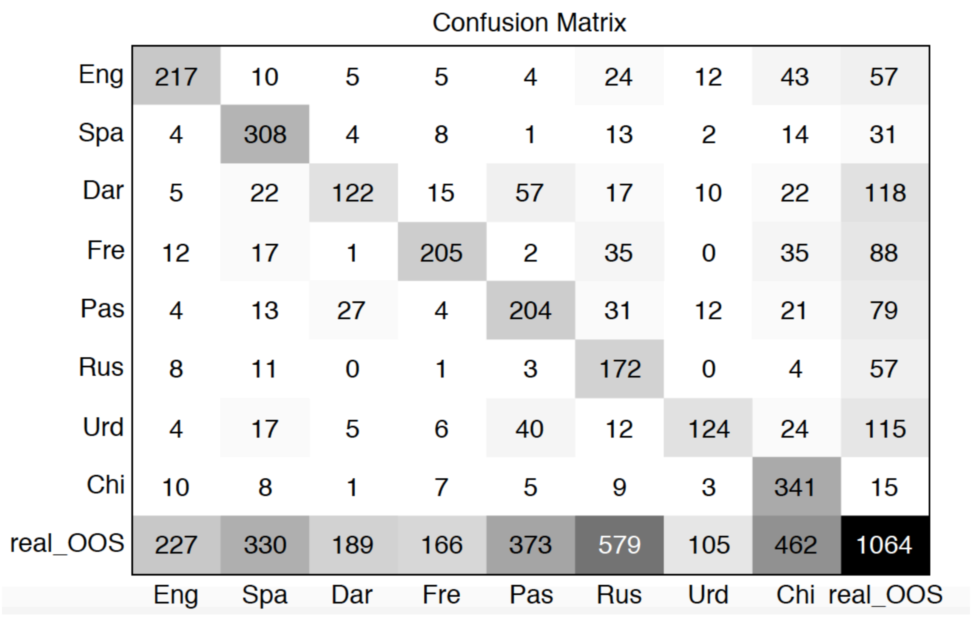
Table 3.8: *Out of set LSTM and i-vector systems performance on the NIST LRE 2009 eight languages subset. The two first systems are the same than shown in Table 3.7. The two systems following are systems capable of dealing with out of set task.*

trained with 1 class for modeling out of set inputs) degrades moderately with respect to the standard system. This result was expected because the new class we are training is harder to learn than the 8 initial classes (it is seeing utterances from 15 languages instead of just one). Thus, the error, when back propagated, dominates the network at training time disturbing the fine-tune of the classes corresponding the 8 target languages. Moreover, among the languages used to train the OOS class, there are languages highly similar with the target languages (e.g. Indian English vs U.S. English). Second, the performance when dealing with real out of set data is as good as it is when dealing with the trained out of set dataset. Having comparable performance with those datasets proves the potential ability of this kind of neural network to tackle the out of set problem (network fed with unseen classes as inputs) and generalize robustly. Finally, we can see that the LSTM system performs better than the reference system, proving its robustness, but the gap gets much closer when dealing with out of set data.

In order to have a better insight into the behavior of the LSTM RNN system when dealing with out of set test segments, we show in Figure 3.9 the confusion matrix of the best out of set system, OOS_lstm_2_layer_512_units, when fed with real out of set test utterances. The most important point here is to see how, while not disturbing too much the classification of the 8 target languages (compare with Figure 3.8), the accuracy obtained for the out of set test segments is far worse. Thus, while our approach seems a good starting point there is still room for further improvement.

4. Limited duration test utterances

   In these experiments we show the results obtained with even shorter versions of our test set in order to gain some insight about the relation between the performance of the system (in terms of accuracy) and the length of the test utterances. Thus, we explore the performance degradation when limiting the test duration. Following the same reasoning than in the previous out of set experiments, the systems used to perform this task are the systems with 512 units per layer and one or two layers.



Confusion Matrix

	Eng	Spa	Dar	Fre	Pas	Rus	Urd	Chi	real_OOS
Eng	217	10	5	5	4	24	12	43	57
Spa	4	308	4	8	1	13	2	14	31
Dar	5	22	122	15	57	17	10	22	118
Fre	12	17	1	205	2	35	0	35	88
Pas	4	13	27	4	204	31	12	21	79
Rus	8	11	0	1	3	172	0	4	57
Urd	4	17	5	6	40	12	124	24	115
Chi	10	8	1	7	5	9	3	341	15
real_OOS	227	330	189	166	373	579	105	462	1064

Figure 3.9: *Out of set confusion matrix. Confusion matrix of the best out of set LSTM RNN system, lstm_2_layer_512_units (system #6), on the real out of set NIST LRE 2009 set (3 s test segments). Ground truth is represented in the Y axis while the predicted language is represented in the X axis.*

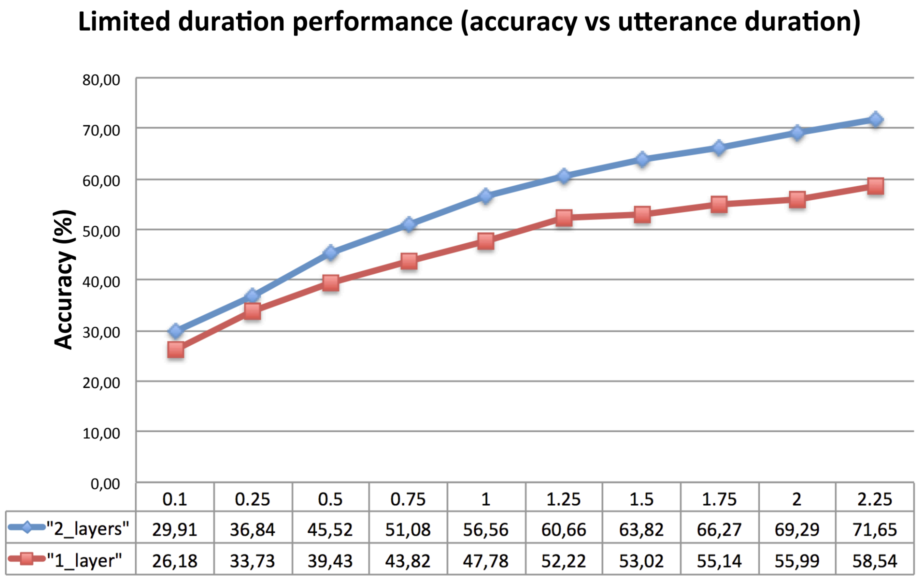
The average accuracy as a function of the test duration is depicted in Figure 3.10. To have a better understanding on those results it may be useful to realize that a typical phoneme duration is about 60-70 ms so we would like to highlight that accuracy rates about 30% are achieved with less than two phonemes in average. Second, as expected, the performance of both systems increase with the duration of the test so a more reliable system can be built when larger test utterances are present.

5. Testing with CTS data

   In our experiments of this section we chose our data to be VOA only in order to avoid effects due to a mismatch or imbalance between different databases. Here, we want to test how our system performs in presence of CTS data, even though our neural networks and i-vector systems have been trained only on VOA data. The systems selected for this experiment are, for the same reasons, the ones chosen for the out of set experiments. In Table 3.9 we can see that the LSTM system performs even better with CTS data and clearly outperforms the i-vector based system, proving its robustness and generalization capabilities.

3.3.4. Analysis on NIST LRE 2015

In this section we evaluate our LSTM-based system on the development and evaluation data of the NIST LRE 2015, which poses a challenge in terms of similar languages, unbalanced data,



	0.1	0.25	0.5	0.75	1	1.25	1.5	1.75	2	2.25
◆ "2_layers"	29,91	36,84	45,52	51,08	56,56	60,66	63,82	66,27	69,29	71,65
■ "1_layer"	26,18	33,73	39,43	43,82	47,78	52,22	53,02	55,14	55,99	58,54

Figure 3.10: *Limited duration performance. Accuracy of our LSTM RNN systems when dealing with fixed-time, super-short test utterances.*

	EER$_{avg}$(%)	
	main_test_set	cts_test_set
reference i-vector based system	16.94	25.17
lstm_1_layer_512_units	17.82	16.09
lstm_2_layer_512_units	**12.51**	10.71

Table 3.9: *Mismatched performance. All the systems shown here have been trained only with VOA audio while tested on both VOA and CTS data.*

duration variability and database mismatch.

3.3.4.1. Datasets Description

The dataset used for this second set of analysis is the one provided by NIST for the core task (limited training data) of the NIST LRE 2015 [NIST, b]. This dataset includes CTS and Broadcast Narrow Band Speech (BNBS) within the training data, and 20 different languages grouped according to 6 clusters (see Table 3.10), with no inter-cluster trials. The total amount of hours available for training and development per language ranges from about half an hour to more than 100 hours. For more information, see the evaluation plan for the NIST LRE 2015 [NIST, b].

Differently from the other dataset used in the previous analysis, the emphasis of these experiments is on discriminating among languages that are similar to each other and frequently mutually intelligible. Moreover, some clusters are composed of a set of languages with completely

Cluster	Target Languages
Arabic	Egyptian, Iraqi, Levantine, Maghrebi, Modern Standard
Chinese	Cantonese, Mandarin, Min, Wu
English	British, General American, Indian
French	West African, Haitian Creole
Slavic	Polish, Russian
Iberian	Caribbean Spanish, European Spanish, Latin American Spanish, Brazilian Portuguese

Table 3.10: *Target languages and language clusters in NIST LRE 2015.*

different amount of data available to train the system, which makes the task more challenging, since it requires dealing with unbalanced clusters.

In order to evaluate the performance of the system in this challenging task, two subsets have been selected as evaluation sets. The first subset mimics the experiments done in the evaluation time, when only the development data was available. Thus, this evaluation set is a 15% of the development data, which was not used to train the system. This subset was split in segments of 3, 10 and 30 seconds to evaluate the performance in different durations. The second subset is the real evaluation dataset, which was not limited to segments of given durations but covered a broad range of speech durations.

3.3.4.2. Evaluation Metrics

Two different metrics were used:

- C_{avg}, as described in the NIST LRE 2015 [NIST, b] evaluation plan, has been used as the main error measure to evaluate the capabilities of the system to identify languages in a one-vs-all way, just considering languages inside the same cluster. C_{avg} is a cost function that penalizes taking bad decisions, therefore it considers both discrimination and the ability of setting optimal thresholds (i.e. calibration).

- EER_{avg}, the mean of the Equal Error Rate computed language by language has been used for easier comparison being an extensively used metric in the community.

3.3.4.3. Experiments and Results

1. Development set of NIST LRE 2015

   The system proposed in our experiments using NIST LRE 2015 consists of one neural network per cluster. All the neural networks have the same architecture which corresponds to the best performing system in the previous experiments on the NIST LRE 2009 (see previous Section 3.3.3): two hidden layers of 512 units followed by an output layer. The

		C_{avg} / **EER$_{avg}$(%) per cluster**						
	System	Arabic	English	French	Iberic	Slavic	Chinese	**Average**
	LSTM	**0.1379** / **13.28**	0.1888 / **11.40**	**0.0270** / **2.92**	**0.1711** / **14.40**	**0.1501** / **15.24**	**0.1011** / **8.73**	**0.1293** / **11.00**
3 sec	i-vector	0.1559 / 15.86	**0.1391** / 11.47	0.0568 / 6.17	0.1996 / 19.90	0.1971 / 19.84	0.2206 / 19.24	0.1615 / 15.41
	Fusion	*0.1150 / 11.27*	*0.1248 / 7.86*	*0.0286 / 2.25*	*0.1328 / 10.67*	*0.1371 / 13.69*	*0.0975 / 7.83*	*0.1060 / 8.93*
	LSTM	0.0976 / 9.65	0.1932 / 9.63	0.0304 / 2.12	0.1673 / **11.43**	0.1059 / 10.85	**0.0906** / **7.44**	0.1142 / 8.51
10 sec	i-vector	**0.0750** / **7.63**	**0.0747** / **4.28**	**0.0198** / **1.06**	**0.1449** / 12.26	**0.1004** / **10.18**	0.1133 / 8.32	**0.0880** / **7.29**
	Fusion	*0.0609 / 5.81*	*0.0461 / 4.44*	*0.0127 / 1.06*	*0.1003 / 7.92*	*0.0717 / 7.12*	*0.0648 / 3.83*	*0.0594 / 5.03*
	LSTM	0.0859 / 8.93	0.1876 / 6.79	0.0104 / 1.88	0.1473 / 10.33	0.0868 / 8.68	0.0995 / 7.16	0.1029 / 7.30
30 sec	i-vector	**0.0308** / **3.23**	**0.0199** / **0.76**	**0** / **0**	**0.1278** / **8.60**	**0.0423** / **4.59**	**0.0493** / **3.77**	**0.0450** / **3.49**
	Fusion	*0.0306 / 2.84*	*0.0387 / 0.28*	*0 / 0*	*0.0984 / 5.38*	*0.0331 / 2.99*	*0.0460 / 1.92*	*0.0411 / 2.24*

Table 3.11: *Results on the NIST LRE 2015 development dataset (testing on an unseen 15% of the development dataset) in terms of EER_{avg} and C_{avg}.*

hidden layers are uni-directional LSTM layers while the output layer is a softmax with as many units as languages in the cluster.

In this section we want to analyze the results of the best performing system in a controlled environment when dealing with a more challenging scenario. In these experiments we have 6 clusters with 2 to 6 similar languages in each and the data available to train is neither balanced nor controlled at all. For the following experiments we have not fine tuned our system because we want to test the performance of the best system in the previously controlled environment when facing a more difficult task, dealing with closer languages, two different sources of data, completely unbalanced datasets, etc.

Table 3.11 summarizes the results on an unseen 15% of the development set. Two major messages can be extracted from these results. First of all, note that the LSTM RNN based system performs better than the i-vector system when the test utterances are short enough (most of the 3 s utterances and some of the 10s) while the i-vector approach is solidly better when dealing with long utterances. For example, in the 3 seconds subset, as we show in Figure 3.11, the recurrent neural network approach has an improvement higher than 20% in terms of C_{avg} with respect to the reference system. Secondly, we can observe that the fusion of the two systems behaves considerably better and is more robust than any of the single systems, outperforming the deep learning approach even in short durations and the i-vector system when facing long ones.

2. Test set of NIST LRE 2015

The second scenario we wanted to test our system in corresponds to the fixed-training task of the NIST Language Recognition Evaluation 2015. In this experiment we have the same setup as we had in the previous one but a big mismatch is observed between the development data set and the test set. In these conditions, we wanted to analyze the impact of this mismatch in the proposed system and compare it with the degradation suffered by the classical i-vector approach.

Our results as depicted in Figure 3.12. We can observe the performance of the two different

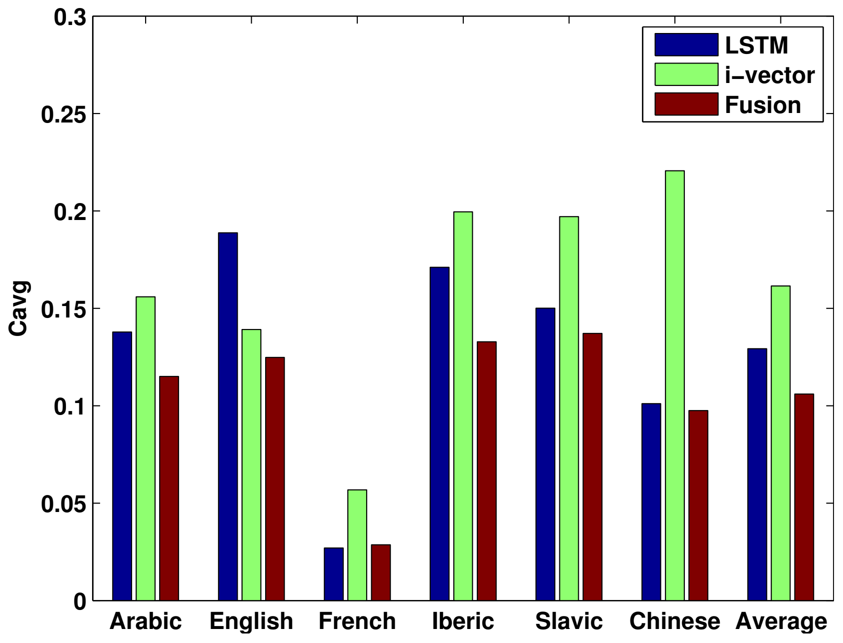


Figure 3.11: *Performance of the proposed system compared to the reference i-vector system on our 3 seconds subset of the development set of NIST LRE 2015*

systems analyzed in function of some duration bins ranging from those shorter to 3 seconds to those longer than 30 seconds. The last four bars in Figure 3.12 are the result of the two analyzed systems in the evaluation test set. First of all, we can see that in this scenario the i-vector system outperforms the LSTM RNN based system, showing that our deep learning approach suffers from a bigger degradation in presence of such a mismatch. Secondly, we can see that even though the performance of the i-vector system has a similar result than the LSTM for short utterances, it becomes more accurate when the test utterances become longer while the LSTM seem not to gain so much accuracy in presence of long utterances due to the big mismatch between training and test data.

Finally, we can also see in Figure 3.12 the result of two different fusions of the analyzed systems. The first fusion is the submitted linear logistic regression fusion learned over training data. The result of this fusion is not better than the best system itself in any of the durations, probably because training a fusion with data that does not represent the test data led to a wrong fusion which is not able to extract complementary information. In order to prove this hypothesis we have also shown a post-eval fusion (not valid for the evaluation task) with 2-fold cross-validation (Fusion CV in Figure 3.12), splitting the test-set in two subsets and training the fusion on one dataset and applied to the other one and vice-versa, leading to an optimistic fusion. As the results show, this fusion performs better than the single systems in every duration proving that the information learned by the two individual systems is complementary.

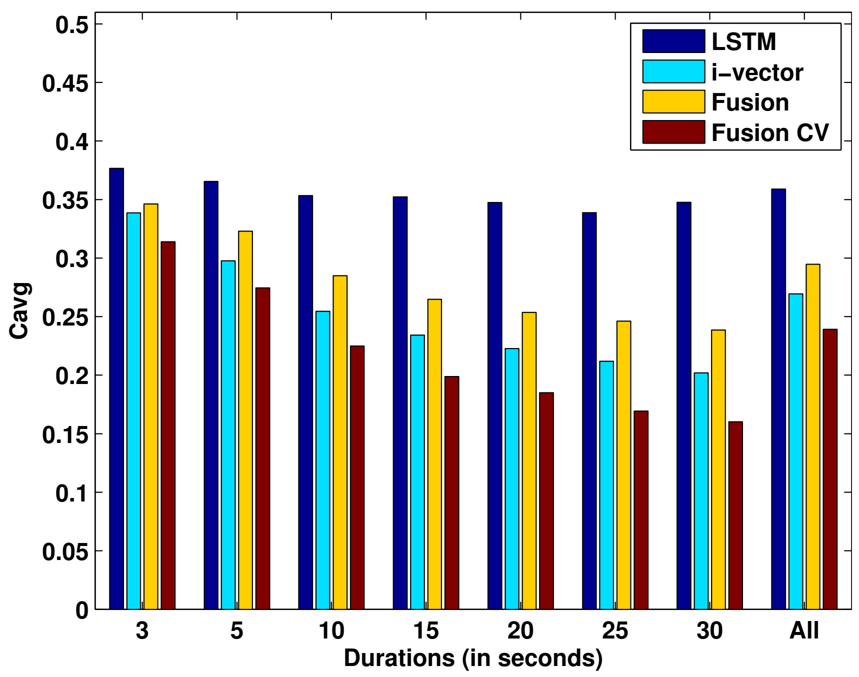


Figure 3.12: *Results of our proposed LSTM RNN system and the reference i-vector system on NIST LRE 2015 fixed-training task divided by duration bins, followed by the overall results.*

3.4. Chapter Summary

In this chapter, we have presented two different end-to-end DNN architectures for the task of language recognition: convolutional deep neural networks (CDNNs) and long short-term memory recurrent neural networks (LSTM RNN).

Firstly, we analyzed the performance of CDNN-based systems, which are lighter (in terms of number of parameters) than traditional approaches based on i-vectors and other deep learning architectures. We tested them in short test segments (less than 3 seconds of speech) on two different tasks extracted from the NIST LRE 2009 dataset: pair-wise and multiple language recognition. The proposed models for the task of language-pairs managed to outperform the reference systems in two out of the five pairs considered. For the case of multiple languages, our CDNN-based systems, obtain performances comparable to the i-vector baseline system while using at least 100 times fewer parameters. Furthermore, by combining our CDNN-based systems and the reference systems, we obtain improvements, which means that both approaches extract complementary information.

Secondly, we explored an efficient LSTM RNN approach for language identification that, being lighter as well, successfully exploits temporal information of the speech signal sequences. We kept the focus on the short test segments task. In particular, we presented a set of experiments on LSTM-based systems in different conditions from the NIST LRE 2009 and 2015 frameworks. Results show that the proposed system using significantly fewer parameters (∼1-5 M vs ∼23 M) clearly outperforms the reference system on a controlled environment with balanced datasets, limited channel variability and no big mismatch between development and test, specially when dealing with short utterances. Furthermore, we evaluated the accuracy of the system when using very short utterances (from 0.1 to 2.25 seconds of speech). For our proposed LSTM RNN scheme, we found that accuracy over 70% may be achieved with about 2 seconds while more than 0.5 seconds are needed in order to obtain an accuracy rate over 50%. In this same controlled setup, we have also tackled the problem of non seen languages (out of set) with an LSTM RNN approach with little degradation when dealing with out of set languages with respect to the system tested with the languages used to train the out of set class. In addition, we have tested our systems with a mismatch in the databases (training with VOA and testing with CTS) obtaining results comparable with the matched experiment, the proposed system outperforms the reference i-vector system on the same conditions, proving its robustness. Results using NIST LRE 2009 (8 languages selected) demonstrate that LSTM RNN based approaches using open source implementations outperform the standard i-vector based approach when dealing with short test durations with an improvement of $\sim 26\%$ in terms of Equal Error Rate with respect to the reference i-vector based system. On the other hand, when facing a more challenging scenario with highly-unbalanced datasets and closer languages the performance of the neural network system gets closer to the classical i-vector approach, being still better on short utterances while the i-vector is more accurate on longer ones. Moreover, the information learned by LSTM RNN systems in this scenario is complementary to the information learned by the i-vector system, being the fusion of both systems solidly better in all the durations. On the actual test set of the NIST LRE 2015, which has all the variability previously mentioned but it also adds a mismatch between the training data and the test data, our LSTM systems are more sensitive than the i-vector based systems to this severe mismatch and cannot surpass its performance. Our findings show that this LSTM end-to-end approach for language recognition with $\sim$85% less parameters than a classical i-vector system can achieve robust and comparable results in several challenging scenarios. Nevertheless, this kind of systems strongly depend on the similarity between the development and the test dataset and seem to need further research on variability compensation.

Chapter 4

Frame-by-frame DNN-based Representation: Bottleneck Features

Automatic language and speaker recognition sytems have been including deep neural networks (DNNs) in their pipeline in the last few years after their success in the related field of automatic speech recognition (ASR). These DNNs started replacing different parts of the traditional i-vector approach, achieving impressive performance and becoming part of the state-of-the-art in these fields.

In this chapter, DNNs inclusion in language and speaker recognition is reviewed, focusing on one of the most successful approaches: DNNs as bottleneck feature extractors. In this case, DNNs were used to obtain a frame-by-frame representation of the input signal which resulted in a new sequence of feature vectors representing the signal. This way, bottleneck features replaced or complemented the traditional acoustic features such as MFCCs or PLP and improved noticeably the performance of existing language and speaker recognition systems.

Furthermore, we present our study on bottleneck feature based language and speaker recognition over well-known datasets, the NIST LRE 2015 and the NIST SRE 2010, where we analyze the effect of different configurations of the bottleneck-based DNN in the final performance.

4.1. Introduction to Bottleneck Features for Language and Speaker Recognition

Language and speaker recognition systems based on bottleneck (BN) features have become state-of-the-art approaches in these fields. Generally in these systems, a deep neural network (DNN) with a bottleneck layer (BN) is trained for speech recognition. This DNN is fed with input vectors representing frames of audio segments, and the time-dependent output of the bottleneck layer is used as a new frame-by-frame representation of the audio signal. With those feature vectors, the classical UBM/i-vector scheme is used to obtain an utterance level representation which will be used for the target language or speaker recognition task. A representation of a

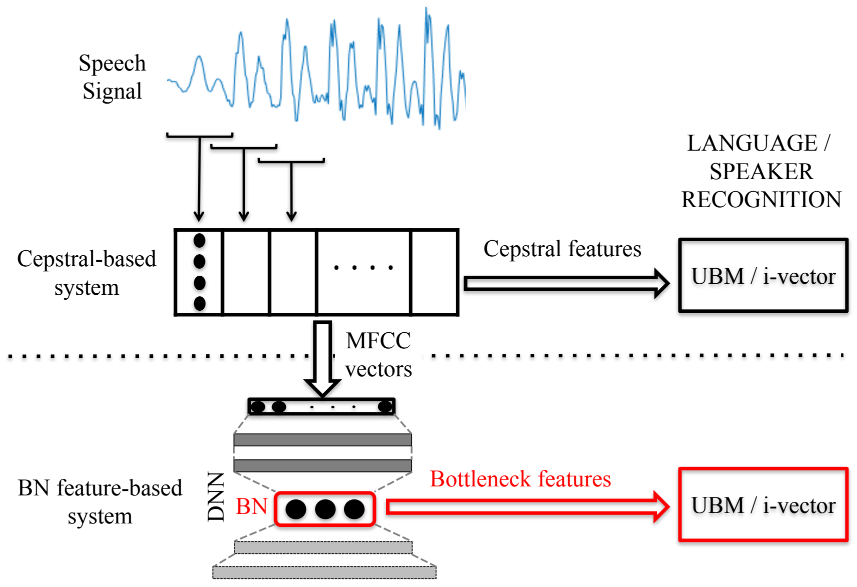


Figure 4.1: *Representation of language/speaker recognition system structure. This is a graphical representation of a generic language/speaker recognition system, both based on cepstral features and bottleneck features.*

possible structure of this kind of systems where cepstral features are replaced by the resulting BN features is depicted in Figure 4.1.

Broadly speaking, bottleneck features can be seen then as a new frame-wise representation of an audio signal, learned directly by a DNN. The underlying motivation is to obtain more abstract feature vectors, which help to model the feature space and contain useful information, allowing the network to learn it by itself and reducing the dependency of hand-crafted features.

In general, a DNN with a bottleneck layer is a feed-forward neural network composed of several hidden layers where one of them, the bottleneck layer, is relatively small with respect to the rest. Moreover, it often applies a linear transformation in contrast to non-linear transformations applied in the rest of the hidden layers. This bottleneck layer aims to compress the information learned by the previous layers, projecting it into an useful representation for the task for which the DNN is trained (for instance, automatic speech recognition, ASR) [Grezl *et al.*, 2007]. Even though the bottleneck layer forces the network to compress information making the classification harder, it has advantages as the transformation from input to classification is less straightforward making the system less prone to overfitting and more robust [Matějka *et al.*, 2014b]. Moreover, it achieves dimensionality reduction from a full hidden layer in order to be used as feature vectors for other tasks.

Typically, in language and speaker recognition systems based on bottleneck features, the DNN is trained for ASR. For the case of language recognition, this is motivated by the phonemes and phoneme sequences being different depending on the language. In fact, the first successful approaches to automatic language recognition were systems based on phonetic information

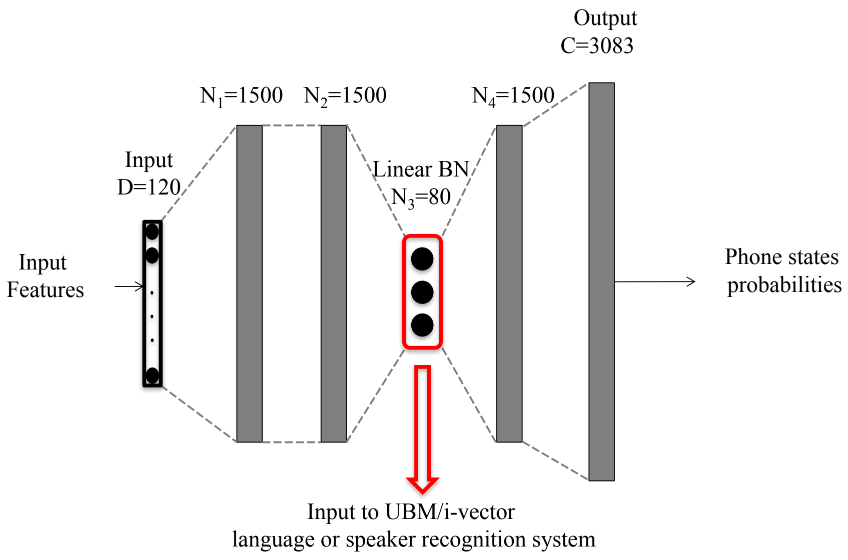


Figure 4.2: *Example of DNN architecture with bottleneck layer. This is a graphical representation of the topology of a DNN with a BN layer, whose outputs (activation values) are used as input feature vectors for the language or speaker recognition system.*

(PRLM, PPRLM) [Zissman and Singer, 1994]. Therefore, the phonetic information has proved to be useful for the task of language recognition itself [Garcia-Romero and McCree, 2015; Matějka *et al.*, 2014b]. For the case of speaker recognition, the DNN is also able to capture phonetic information which helps characterizing different speakers. However, the discriminative power of BN features is sometimes not enough to outperform traditional cepstral features but a good complement to them.

An example of the structure of a DNN used as bottleneck feature extractor for language and speaker recognition in the following sections is shown in Figure 4.2.

4.2. Analysis of Bottleneck Features for Language Recognition

In this section, we analyze in a systematic way how the topology of the DNN influences the performance of a bottleneck feature-based language recognition system over the NIST LRE 2015 dataset. In particular, we present results with different topologies for the DNN used to extract the bottleneck features, comparing them and against a reference i-vector system based on classical cepstral representation of the input signal. For instance, we explore how the position of the bottleneck layer, either closer to the input layer, or moving towards the output layer, obtaining higher level information each time, influences the final system performance.

4.2.1. Database Description: Switchboard and NIST LRE 2015

4.2.1.1. Training Datasets

The language recognition system used in this section has two clearly separated parts to be trained: the DNN used as feature extractor, and the i-vector pipeline (which includes GMM-UBM and total variability subspace training).

Thereby, two different databases are considered in order to train each part: Switchboard database and NIST LRE 2015 training data.

- Switchboard

  In this section, we use the Switchboard (Part 1 release) database [Godfrey and Holliman, 1993] to train the DNN to be used later as a bottleneck extractor. This set contains approximately 320 hours of speech (telephone conversations in English) from around 4800 speakers. A 10% of this dataset is reserved to validate the DNN performance.

  This dataset is labeled for speech recognition purposes, at word level, and will be used to train an ASR system (developed in Kaldi [Povey *et al.*, 2011b], a toolkit for speech recognition widely used in the field) to obtain triphone state level label alignments. These alignments are then used to train the DNN with a bottleneck layer.

- NIST LRE 2015 training data

  To train the language recognition system (UBM and T matrix), we use the NIST Language Recognition Evaluation 2015 training data [NIST, b].

  As in other evaluations, the dataset contains both conversational telephone speech (CTS) and broadcast narrowband speech (BNBS) data. It involves segments of speech from twenty different languages, grouped into six clusters according to similarities and relation between languages (see Table 4.1). This way, the focus of this evaluation is distinguishing among closely related languages, i.e., within each cluster.

  In particular, we use the data provided for the core task (with limited data) of the NIST LRE 2015, which contains a set of segments from the twenty target languages. The segments were audited by the Linguistic Data Consortium (LDC), so they contain at least 30 seconds of speech belonging to the labeled language. It should be taken into account that the amount of data per language, even within a cluster, varies notably. For example, the English cluster includes about 30 minutes of British English but, however, more than 100 hours of General American (see Table 4.1).

  To train the i-vector language recognition system we choose randomly 85% of this training data, using the remaining 15% for evaluation purposes as explained below.

4.2.1.2. Test Datasets

In this section, we evaluate our language recognition systems in two different datasets from the NIST LRE 2015.

Cluster	**Target Languages**	**Hours of data**
Arabic	Egyptian	95.4
	Iraqi	37.2
	Levantine	41.1
	Maghrebi	38.6
	Modern Standard	3.7
Chinese	Cantonese	3.4
	Mandarin	71.8
	Min	8.1
	Wu	7.7
English	British	0.5
	General American	100.0
	Indian	8.1
French	West African	7.7
	Haitian Creole	2.7
Slavic	Polish	30.0
	Russian	18.0
Iberian	Caribbean Spanish	26.9
	European Spanish	8.1
	Latin American Spanish	6.9
	Brazilian Portuguese	0.8

Table 4.1: *Cluster of target languages and approximate amount of data per language in the NIST LRE 2015 training dataset.*

- Matched test dataset

  Firstly, we consider a matched setup, in which we evaluate the systems with the 15% of the training NIST LRE 2015 dataset which has not been used for training the language recognition system. We segmented the speech recordings into fragments of 3, 10 and 30 seconds of speech. The number of fragments used for each subset is 84446, 25592 and 8757, respectively. This corresponds to approximately 70 hours of actual speech for this matched test dataset.

- Mismatched test dataset

  Then, in order to test how the reference system and the developed bottleneck features based system perform in a mismatched dataset (from a totally different collection of audio recordings with respect to data used to train the systems), we present this comparison over the evaluation data of NIST LRE 2015. In this case, test segments are not constrained to have a specific duration as in previous evaluations and in our other test set (3, 10 or 30 seconds). Instead, they covered a broad range of durations, from 3 to 260 seconds of

	DNN	**UBM/i-vector**
Train	Switchboard (90%)	NIST LRE 2015 (85% from training data)
Test	Switchboard (10%)	1) Matched dataset (15% from training data, NIST LRE 2015)
		2) Mismatched dataset (evaluation data, NIST LRE 2015)

Table 4.2: *Datasets used for training and testing our bottleneck feature based language recognition systems.*

speech. The evaluation dataset includes 164334 segments from all the target languages used in training, both from CTS and BNBS. More details can be found in the NIST LRE 2015 evaluation plan [NIST, b].

This structure of subsets for training and testing the systems is summarized in Table 4.2.

4.2.2. Evaluation Metrics

In order to evaluate the developed systems, we used different evaluation metrics.

Firstly, we show results of the DNNs used as bottleneck feature extractors in terms of phoneme state frame accuracy, i.e. the percentage of frames classified correctly by the DNN according to the given phoneme state labels.

Secondly, performance for the final language recognition task is presented as average Equal Error Rate (EER): we first compute EER as a one-versus-all approach, and then average them to have a final average EER (EER_{avg}). Since we are evaluating in the context of the NIST LRE 2015, which involves clusters of similar languages, we treated each cluster separately as it was done in the evaluation, not taking into account scores of a given test segment outside the cluster it belongs to. This way, we compute the average EER for each cluster, and use the final average of those partial results as the final language recognition system performance.

4.2.3. Cepstral based i-vector Reference System Description

The reference language recognition system considered for this set of experiments follows the classical i-vector based approach, in which each i-vector will be a low-dimension representation of a given utterance.

In order to compute the mentioned i-vectors, each audio recording is represented by a feature vector for each frame (or segment of 20 ms of speech in our system). We use the Mel-frequency Cepstral Coefficients (MFCC) as parameters (or input feature vectors) for our reference system. These parameters represent the acoustic information contained in the audio recordings. In particular, in these experiments, we augmented the MFCCs with temporal information given by the Shifted Delta Cepstral coefficients (SDC) [Torres-Carrasquillo *et al.*, 2002], in order to consider the information given by the context of a certain frame.

To compute these input vectors, we use a Hamming window of 20 ms, with 50% overlap, and a filter-bank of 25 Mel-scaled filters. We compute then the MFCC-SDC for each frame [Torres-

Duration	**30s**	**10s**	**3s**
EER$_{avg}$ (in %)	7.35	14.08	21.56

Table 4.3: *Cepstral based i-vector reference system (i-vector based on MFCC-SDC features) performance in NIST LRE 2015, in terms of average EER (EER$_{avg}$) of all language clusters.*

Carrasquillo *et al.*, 2002], created by stacking delta cepstral coefficients computed across multiple speech frames. In particular, we use 7 MFCCs, with one frame advance and delay for the delta computation, and we stack 7 blocks with a time shift of 3 frames between them (7-1-3-7 configuration), which results in 56-dimensional feature vectors representing each frame of the utterance.

In this feature space, an Universal Background Model (UBM) composed of 1024 Gaussian components is trained, and Baum-Welch statistics are computed over this UBM for each utterance. Then, a Total Variability (TV) subspace of 400 dimension is derived from them by using PCA (Principal Component Analysis) followed by 10 EM iterations. All this process is carried out using Kaldi [Povey *et al.*, 2011b]. Finally, cosine similarity is used to classify the resulting i-vectors. Results of this system can be seen in Table 4.3.

The performance of our reference system improves with the amount of speech contained in the test segments. This way, results range from 7.35% of average EER (EER$_{avg}$) in the case of test segments containing 30 seconds of speech to 21.56% for shorter segments of 3 seconds. This is due to a better estimation of the i-vector obtained when the amount of speech is enough to have reliable estimations for this approach. It should be noticed that the target task is classification among similar languages, in the context of the NIST LRE 2015, which makes the task especially difficult when test segments are short (few seconds of speech).

4.2.4. Bottleneck Feature based Language Recognition System Description

To develop the language recognition system used in this section, frame level bottleneck features are extracted from a DNN trained for ASR. This bottleneck feature vectors replace the MFCC-SDC feature vectors used in the reference system.

The DNN architecture used in this section is a feedforward network with an input layer, three to five hidden layers, and the output layer, which follows the general architecture depicted in Figure 4.2.

To feed the network, 20-dimensional MFCC input vectors are used and preprocessed stacking 31 frames together (15 frames of context in both temporal directions). Then, the temporal trajectory of each MFCC coefficient is smoothed by applying a Hamming window followed by a DCT of which only coefficients 0 to 5 are kept, as done in [Karafiát *et al.*, 2014]. The resulting 120-dimensional feature vector is used as the input to the DNN.

This input layer is followed by three to five hidden layers, which perform non-linear transformations (except the bottleneck layer that aims to compress information and uses a linear activation) of the input in order to obtain a better phoneme state classification performance. These hidden layers are composed of 1500 units which apply the *sigmoid* function as activation or non-linear transformation. The activation value computed in each unit is the resulting value of the application of the non-linear function to the input multiplied by the weights and bias.

The bottleneck layer applies a linear transformation, and we vary its size and position depending on the experiment.

Finally, the architecture is completed with a softmax layer, which outputs the probability of each input to correspond to a given phoneme state. In our case, the output layer tries to discriminate among 3083 triphone states.

The training algorithm used is the stochastic gradient descent with batches of size 512 samples (i.e. computing the gradients with each block of 512 input segments), and the cost function we aim to optimize is cross-entropy. The training process is stopped when the error function converges according to a validation set, a disjoint set from the training data (the remaining 10% of Switchboard dataset not used for training).

4.2.5. Experiments and Results

In order to develop a bottleneck feature based language recognition system, many parameters have to be tuned. For this set of experiments, we first obtain the phonetic alignments of a given set of Switchboard with an ASR system via Kaldi with a fixed configuration. The configuration parameters of the UBM/i-vector system used as language recognition backend are also fixed through the experiments in this section. Then, we explore different configurations of the DNN used as bottleneck feature extractor. Thus, in this section, we describe these experiments varying the DNN architecture and present results, both in terms of DNN performance (frame accuracy of phoneme states classification) and final language recognition performance (average EER, EER_{avg}), as it was described in Section 4.2.2.

We evaluate the language recognition system in the test-development dataset described in Section 4.2.1.2, where we explore the influence of variations in the topology of the DNN, and, finally, we show the results in the evaluation dataset of the NIST LRE 2015.

4.2.5.1. Results on the Matched Test Dataset

1. Number of Hidden Layers

   This first set of experiments aims to evaluate the performance of systems in which the number of hidden layers in the DNN varies from 3 to 5, with the bottleneck layer occupying central positions (layers 2, 3 and 3, respectively). Despite resulting in a better performance in terms of phoneme frame classification with the 5 layers configuration (see Table 4.4), it is the architecture with 4 hidden layers the one that reaches the lowest EER_{avg} in terms of language recognition performance.

   It is very interesting to see that the system which gives the best performance in terms of phoneme frame accuracy does not lead to a better bottleneck feature extractor for LID,

Number of	DNN	EER$_{avg}$ (in %)		
Hidden Layers	**Frame Accuracy**	**30s**	**10s**	**3s**
3	47.82	5.52	9.04	14.34
4	49.55	**4.33**	**7.81**	**13.76**
5	**50.46**	5.22	8.57	14.15

Table 4.4: *DNN (phoneme classification, frame accuracy) and language recognition performance (EER$_{avg}$ of all language clusters) when varying the number of layers of the DNN for bottleneck feature extraction.*

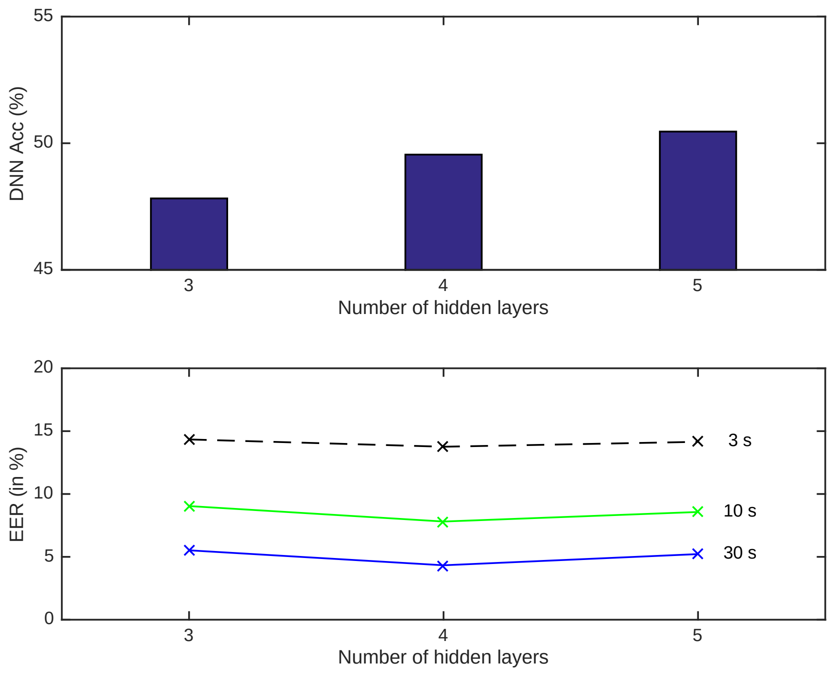


Figure 4.3: *Phoneme frame accuracy of DNN (upper part of the figure) and language recognition systems performance (lower part) for different test durations (3, 10 and 30s) in the matched test dataset of NIST LRE 2015 with different number of hidden layers of the DNN.*

which can be seen also in Figure 4.3.

This might be partially because the ideal speech recognizer would suppress as much as possible information not important for phoneme discrimination, which might include relevant information to distinguish between languages. Also, there is a possible overfitting to the database used to train the DNN, which is different from the one where the LID task is evaluated.

As we have mentioned before, phonetic information is useful for the task of language recognition, since phonemes and phoneme sequences vary depending on the language. However, although both tasks ASR (for which the DNN is trained) and LID might share some information, it is not exactly the same since more factors are involved, and this is

Position of BN Layer	**DNN Frame Accuracy**	**EER$_{avg}$ (in %) 30s**	**10s**	**3s**
First	49.17	9.37	12.24	16.59
Second	49.46	6.27	9.55	14.58
Third	**49.55**	**4.33**	**7.81**	**13.76**
Fourth	48.05	4.64	8.00	14.17

Table 4.5: *DNN (phoneme classification, frame accuracy) and language recognition performance (EER$_{avg}$ of all language clusters), comparing different position for the bottleneck layer in the DNN.*

related to the difference in trends of phoneme frame accuracy and LID performance.

Also, a DNN with a bottleneck layer can be seen as two different networks, which are focused on different tasks: the first network would be the complete network trained for ASR; and the second network would be the part from the input to the BN layer, which can be used as feature extractor for other tasks (LID in our case). Thus, for the complete network, the BN layer imposes a constraint for the information to pass through the network. Such restriction is damaging the performance of the network in terms of frame accuracy for ASR: the same networks trained with three and four full hidden layers (1500 hidden units as the rest) reached a frame accuracy of 49.29% and 50.08% respectively, performance that drops when including the BN layer to 47.82% and 49.55% respectively, as we can see in Table 4.4. However, that information compression is good if the aim is to use that restricted information for other task as LID. Therefore, for the LID system used in this section, we want the DNN to focus mainly on the second part, to obtain a compact representation of the signal useful for LID and not optimize the DNN only for ASR. Therefore, the discriminative task (ASR) is easier for the DNN when the classifier is more complex (5 layers DNN), which is making easier the classification task, and, thus, improves the frame accuracy. However, that network is not being forced to focus on obtaining a compact representation of the signal (and hopefully good), which is used for LID afterwards. Even though there might be other factors influencing this, these explanations support the results where not the best frame accuracy of the DNN related to the task of ASR leads to the best DNN as feature extractor for LID.

2. Bottleneck Layer Position

   Keeping fixed the architecture of the DNN to four hidden layers, we explore how the language recognition system performs depending on the position that the bottleneck layer occupies in the network. Results can be seen in Table 4.5.

   These experiments were carried out in order to explore how different layers of the DNN, which correspond to different levels of extracted information, perform in terms of language recognition.

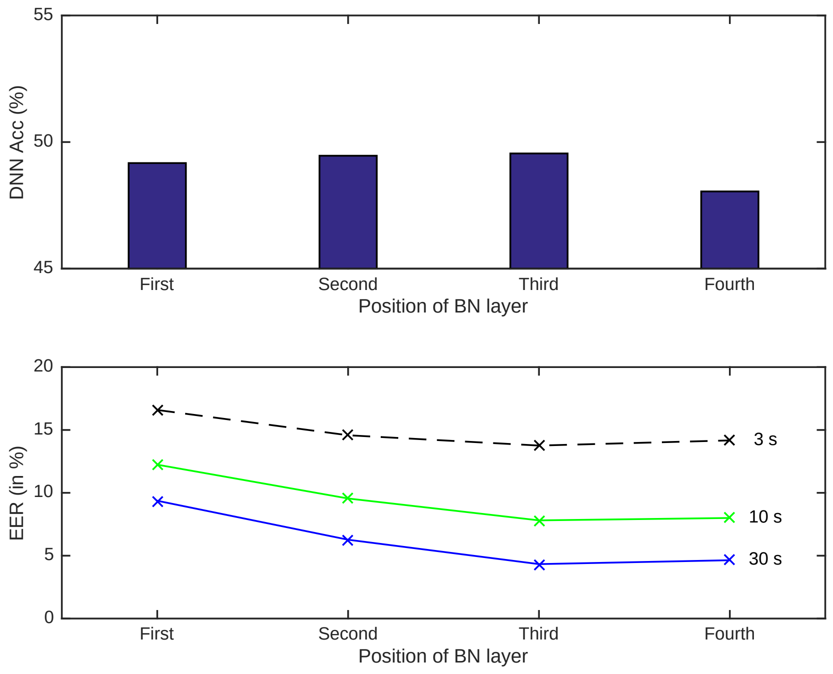


Figure 4.4: *Phoneme frame accuracy of DNN (upper part of the figure) and language recognition systems performance (lower part) for different test durations (3, 10 and 30s) in the matched test dataset of NIST LRE 2015 when the bottleneck layer moves from first to fourth layer in a four hidden layer topology.*

Feedforward DNNs are trained to automatically learn a transformation that maps an input to an output. Each of the hidden layers is then learning features which help with that discriminative task, and as it has been seen in different research works [Lecun *et al.*, 2015], deeper layers learn more abstract features, with more information which help to final classification. In this case, the position of the BN layer is related to the degree of proximity of the information to the phonetic states that the network learns to code.

The closer the BN layer to the input layer, the noisier the resulting representation would be, which might explain the drop in performance for the first and second layers with respect to the results of the last two layers.

The best performance in terms of EER_{avg} for language recognition is obtained when the bottleneck layer is located in the third layer, but that result is very close to the one obtained with the BN at the fourth layer, as it can be seen at the bottom side of Figure 4.4. Bottleneck features from these two layers seem to contain useful information, although still third layer, which is further from the output layer, gives the best results.

Performance of the DNN for phoneme state classification also drops when the BN layer moves from layer third to fourth, from 49.55% to 48.05%. In this topology, the bottleneck layer in position fourth is connected directly to the output layer, resulting in a weight matrix that connects a small layer with just 80 hidden units with the output layer, of size 3083. These weights might be difficult to learn, which may explain this drop in performance

Size of	**DNN**	**EER$_{avg}$ (in %)**		
BN Layer	**Frame Accuracy**	**30s**	**10s**	**3s**
20	46.60	6.42	11.73	18.58
40	48.81	4.57	8.67	15.07
60	49.20	4.63	8.27	14.33
80	49.55	**4.33**	**7.81**	13.76
100	49.60	4.51	7.82	13.39
120	**49.70**	4.94	7.99	**13.08**

Table 4.6: *DNN (phoneme classification, frame accuracy) and language recognition performance in the matched test dataset of NIST LRE 2015 (EER$_{avg}$ of all language clusters) for the experiments with different bottleneck layer size.*

of the DNN.

3. Bottleneck Layer Size

This set of experiments focuses on the size of the bottleneck layer, i.e. the number of hidden units in the bottleneck layer. We train six different neural networks with sizes of the bottleneck layer ranging from 20 to 120 with a step of 20 units. Results are shown in Table 4.6.

In this case, we observe a big gap in performance between the 20 and the 40-dimensional bottleneck layers, both in terms of DNN classification accuracy and language recognition performance (see the elbow at size of 40 units for the bottleneck layer in Figure 4.5).

The rest of the sizes seems to perform in a similar way, with relative improvements not bigger than 9% relative for 30 seconds long utterances, and around 13% relative for test segments of 3 seconds of duration.

As we can see from the results, frame accuracy increases with the size of the bottleneck layer. In fact, the constraint for the information to flow through the network imposed by the bottleneck layer is not beneficial for ASR itself, as we mentioned in the experiments varying the number of layers with the improvement of frame accuracy when training the DNN with full hidden layers. However, that restriction of information included with the BN layer is useful for the use of the DNN as feature extractor for the purposes of LID and other tasks. At the same time, large BN feature vectors increase the complexity of the UBM/i-vector system, which has to deal with high dimensional spaces. As it was shown with the introduction of total variability in the field, which allowed to reduce the feature space dimensionality from the supervector to the i-vector, modeling on low dimensional spaces makes the task less difficult.

Nonetheless, it should be noted that the larger the bottleneck, the more memory resources and computation time is needed to train the language recognition system, due to the

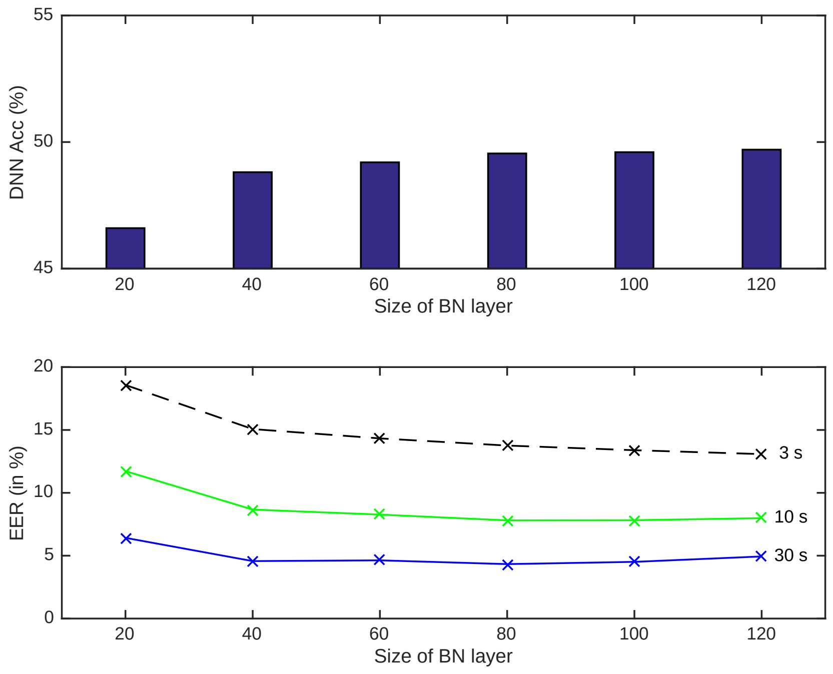


Figure 4.5: *Phoneme frame accuracy of DNN (upper part of the figure) and language recognition systems performance (lower part) for different test durations (3, 10 and 30s) in the matched test dataset of NIST LRE 2015 when the bottleneck layer size (number of hidden units) varies.*

increase in the dimensionality of the feature space.

For these reasons, we select as the best configuration the one with a 80-dimensional bottleneck layer for this setup. It can be observed that the best performance in terms of language recognition for the case of 3 seconds segments is obtained when bottleneck feature vectors have a dimension of 120, showing that for very short speech segments the system might benefit from using larger bottleneck layers.

4.2.5.2. Results on the Mismatched Test Dataset

In this section, we show the results over the mismatched test dataset considered in this experimental part, which corresponds to the evaluation dataset provided by NIST for the LRE 2015.

Different challenging aspects were involved in this evaluation: it focused on similar languages divided in clusters, and, also, on short test segments evaluated as a single task. The distribution of durations of test segments is shown in Figure 4.6.

We evaluate both the reference cepstral based i-vector system and the best configuration according to development results, separated by cluster and on average for all of them (see Table 4.7 and Figure 4.7, respectively). The bottleneck based approach shows a relative improvement of $\sim$8.5% in terms of average EER_{avg} with respect to the cepstral based i-vector reference system based on MFCC-SDC features.

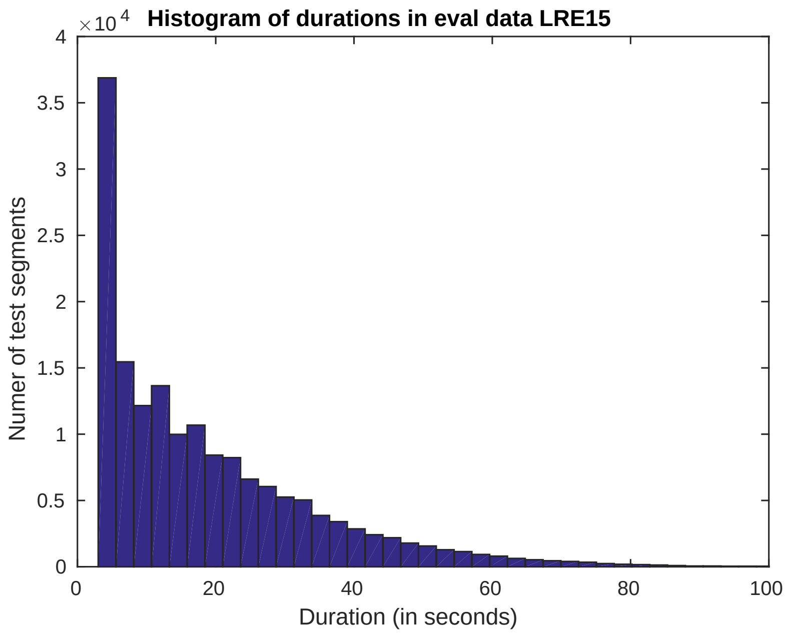


Figure 4.6: *Test duration segments histogram of the mismatched test dataset (the evaluation data of the NIST LRE 2015).*

System	EER$_{avg}$ (in %)
Cepstral-based (reference)	31.51
Bottleneck feature-based	**28.83**

Table 4.7: *Language recognition performance (EER$_{avg}$ of all clusters) for the evaluation data of NIST LRE 2015.*

Results over this test data (see Figure 4.7) show a degradation in comparison with relative improvements obtained in the development set. However, bottleneck feature-based approach outperforms the reference system based on MFCC-SDC features.

It should be noted as well that many factors are different between the development part used as test and this evaluation dataset, such as duration of test segments, or the mismatch existing between the train and test data in this case. Then, degradation in this test set might be mainly caused by the existing mismatch between training and evaluation data.

4.3. Analysis of Bottleneck Features for Speaker Recognition

Bottleneck features have proved to be very effective in i-vector based speaker recognition. However, the bottleneck feature extraction is usually fully optimized for speech rather than speaker recognition task. Thus, in this section, we explore whether DNNs suboptimal for speech recognition can provide better bottleneck features for speaker recognition. We experiment with

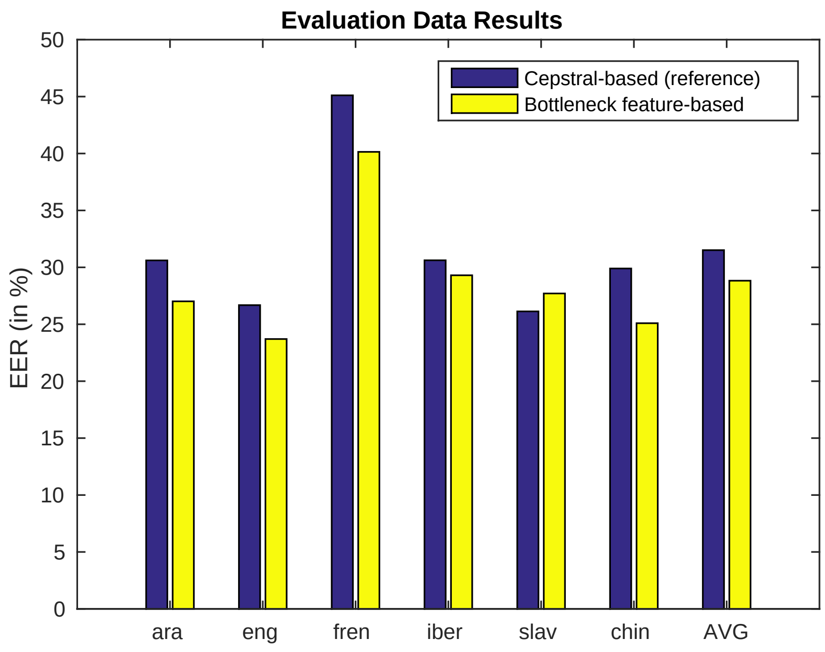


Figure 4.7: *Evaluation data results. This figure shows the performance per cluster and on average for the cepstral based i-vector reference system and the bottleneck feature based language recognition system, for the best configuration found on the development and over the actual evaluation data of the NIST LRE 2015. This configuration was 80-dimensional bottleneck features from third hidden layer in a four hidden layer DNN architecture.*

different features optimized for speech or speaker recognition as input to the DNN. We also experiment with under-trained DNN, where the training was interrupted before the full convergence of the speech recognition objective, in order to check whether the bottleneck features extracted have more information useful for the target task of speaker recognition. Moreover, we analyze the effect of normalizing the features at the input and/or at the output of bottleneck features extraction to see how it affects the final speaker recognition system performance. We evaluated the systems in the NIST SRE 2010, condition 5, female task. Results show that the best configuration of the DNN in terms of phone accuracy does not necessary imply better performance of the final speaker recognition system. Finally, we compare the performance of bottleneck features and the standard MFCC features in i-vector/PLDA speaker recognition system. The best bottleneck features yield up to 37% of relative improvement in terms of EER.

4.3.1. Feature Extraction and Normalization

4.3.1.1. Input Features

In these experiments, we used two different sets of input features to feed the DNN: one is optimized for ASR (we will refer to them as “ASR features”) meanwhile the other is optimized for speaker recognition (referred to as “MFCC features”).

Thus, the experiments tagged as “ASR feat.” are those that use the first set of input features optimized for ASR [Grézl *et al.*, 2009b]. These feature vectors are composed of 24 Mel-filter bank log outputs concatenated with 13 fundamental frequency (F0) features, resulting in a 37-dimensional vector as described in detail in [Karafiát *et al.*, 2014]. Furthermore, utterance mean subtraction is applied on the whole feature vector, which is what we used as default for the ASR task [Karafiát *et al.*, 2014].

For the rest of the experiments, tagged as “MFCC”, we trained the DNN with the traditional MFCC parameterization used successfully in speaker recognition, either adding the derivatives or not (Δ and $\Delta\Delta$). We used 24 Mel-filter banks to compute these MFCC vectors of 20 coefficients, including c0.

4.3.1.2. Short-term Mean and Variance Normalization

The aim of the feature normalization techniques is to compensate the mismatch existent between feature vectors due to environmental effects.

In our experiments, we consider the normalization strategy known as “short-term mean and variance normalization” (ST-MVN), which was shown to be a simple and fast method to successfully normalize speech segments for the speaker recognition task [Alam *et al.*, 2011]. This ST-MVN consists of normalizing the mean and variance in a symmetric sliding window as follows:

$$\bar{F}_{i,j} = \frac{F_{i,j} - \mu_{i,j}}{\sigma_{i,j}} \tag{4.1}$$

where F corresponds to the feature matrix; i and j are the indexes of the frame and the coefficient of the feature vector, respectively; and $\mu_{i,j}$ and $\sigma_{i,j}$ are the mean and standard deviation within the corresponding window. Typically, the window is 3 seconds long (i.e. 150 frames to the left and 150 frames to the right).

This normalization, when applied to cepstral features such as MFCC, is what we also call “floating window cepstral mean and variance normalization” or “short-term cepstral mean and variance normalization” (ST-CMVN).

4.3.2. Datasets and Performance Metrics

We use two different datasets in order to train the two parts of the system: the DNN and the i-vector/PLDA system.

We train the DNN using the Fisher English Part 1 and Part 2 datasets. The dataset is composed of approximately 1700 hours of speech. We use 90% of the data for training and the remaining 10% for validation (speakers in these two sets are disjoint). In order to evaluate the performance of the DNN for the task of phoneme classification, we use the frame-by-frame tied-state classification accuracy, which will be referred to as “phone accuracy” for simplicity.

The i-vector/PLDA speaker recognition system is developed using the female portion of the PRISM [Ferrer *et al.*, 2011] training dataset, discarding any noise or reverberation data. This set comprises Fisher 1 and 2, Switchboard phase 2 and 3 and Switchboard cellphone phases 1 and 2, along with a set of Mixer data sets. A total number of 9670 speakers is used to train the PLDA models.

Finally, the speaker recognition systems are evaluated on female test data from the NIST SRE 2010, condition 5 (telephone condition, normal vocal effort conversational telephone speech in enrollment and test) [NIST, 2010], which includes a total of 236781 trials (3704 targets and 233077 non-targets). The recognition performance is evaluated in terms of the equal error rate (EER, in %) and the minimum detection cost functions (DCF^{min}) as defined in the NIST Speaker Recognition Evaluations 2008 (DCF$_{08}^{min}$) and 2010 (DCF$_{10}^{min}$) [NIST, 2008, 2010].

4.3.3. I-vector PLDA Baseline System

The speaker recognition system used as the reference in this study follows the scheme based on i-vectors and PLDA modeling [Dehak *et al.*, 2011a; Kenny, 2010], which has been a state-of-the-art approach for the speaker recognition task.

As features for this baseline system, we use a 60-dimensional input vector for each frame, which corresponds to the MFCCs+Δ+$\Delta\Delta$ parameterization. To compute these input vectors, we use the same configuration as described in Section 4.3.1.1. Finally, they are normalized according to the ST-MVN described in Section 4.3.1.2, using a sliding window of 3 seconds.

With those features, we train a GMM-UBM, collect the sufficient statistics and train the i-vector extractor (total variability matrix), using the data described in Section 4.3.2. The UBM consists of 512 Gaussian components, and the obtained i-vectors are 400-dimensional vectors. Dimensionality of i-vectors is reduced to 250 using LDA. Such i-vectors are then transformed by global mean and variance normalization, followed by length-normalization [Dehak *et al.*, 2011a; Garcia-Romero and Espy-Wilson, 2011].

Finally, the comparison of i-vectors is done via PLDA [Kenny, 2010], a generative model that models i-vector distributions allowing for direct evaluation of the desired log-likelihood ratio verification score.

The results of this baseline system can be seen in Table 4.8.

It should be noticed that this system is a scaled down system to allow for fast turnaround of the experiments, but conclusions hold for a large-scale system: UBM of 2048 Gaussian components and 600-dimensional i-vectors (see Table 4.10).

	EER (%)	**DCF$_{08}^{min}$**	**DCF$_{10}^{min}$**
Baseline	2.68	0.133	0.517

Table 4.8: *Performance of speaker recognition system based on MFCCs, UBM of 512 Gaussian components, 400-dimensional i-vectors, evaluated on the NIST SRE 2010, condition 5, female task.*

4.3.4. DNN Architecture for Bottleneck Extraction

The DNN used in the experimental part of this section follows again the generic structure shown in Figure 4.2.

For two sets of experiments, we use the two feature sets (ASR and speaker recognition optimized features) as described in Section 4.3.1.1. In both cases, the feature vectors are preprocessed as follows: 31 frames are stacked together (central frame $\pm$ 15 frames of context); then, a Hamming window followed by DCT consisting of 0th to 5th bases are applied on the temporal trajectory of each MFCC (or ASR feature) coefficient [Karafiát *et al.*, 2014]. The resulting feature vector is used as the input to the DNN.

The DNN consists of four hidden layers with 1500, 1500, 80 and 1500 hidden units, respectively. The 80-dimensional layer is the linear bottleneck layer, while the other three apply the sigmoid function as the activation function. The size of 80 for the bottleneck layer was chosen due to experiments performed in [Matějka *et al.*, 2016], for which 80 provided the best performance.

The DNN has an output layer, which applies a softmax function and consists of 2423 units corresponding to triphone tied-states. These states were obtained from the original triphone state tying obtained during GMM-HMM training. For the experiment shown in the first row of Table 4.9, an extended output target ("EOT") set considering 9824 triphone states was used.

The cost function that is optimized is the cross-entropy, and the DNN is trained using stochastic gradient descent.

4.3.5. I-vector PLDA System from Bottleneck Features

The speaker recognition system used for the experiments based on bottleneck features follows the same scheme as the one described in Section 4.3.3. The only difference is that MFCC features are replaced with the bottleneck features described in Section 4.3.4. Otherwise the same i-vector/PLDA speaker recognition system is trained on top of the bottleneck features.

4.3.6. Experiments and Results

We carried out a set of experiments in order to analyze the influence of different aspects when dealing with the speaker recognition systems based on the bottleneck features as summarized in Table 4.9.

The first aspect analyzed is the DNN input features, which are either optimized for ASR or speaker recognition ("ASR feat." vs. "MFCC"). Then, feature normalization is also analyzed (see column 2 of Table 4.9): in the experiments with features optimized for ASR, we applied per utterance mean normalization on top of the input vectors ("Utt. CMN"); while in the experiments using MFCCs, we used the floating window or short-term CMVN ("ST-CMVN"). Finally, for all the experiments, we show results either using "raw bottlenecks" or "normalized bottlenecks", i.e. applying or not applying short-term mean and variance normalization on top of the bottleneck features (right or left hand sides of the table, respectively) [Ferrer *et al.*, 2016].

	Input	**Phone**	**Raw bottlenecks**			**Normalized bottlenecks (MVN)**		
Features	**Norm.**	**Acc.(%)**	**EER(%)**	**DCF$_{08}^{min}$**	**DCF$_{10}^{min}$**	**EER(%)**	**DCF$_{08}^{min}$**	**DCF$_{10}^{min}$**
ASR feat. (EOT)	Utt. CMN	*	2.31	0.105	0.374	2.10	0.093	0.359
ASR feat.	Utt. CMN	**49.8**	2.51	0.100	0.360	2.32	0.103	0.348
MFCC$_{\Delta+\Delta\Delta}$	ST-CMVN	49.6	1.99	0.085	0.328	2.02	0.089	0.337
MFCC$_{20dim}$	ST-CMVN	45.57	**1.67**	**0.079**	**0.312**	**1.91**	**0.088**	**0.325**
MFCC$_{\Delta+\Delta\Delta}$ (UT)	ST-CMVN	47.3	1.72	0.081	0.332	2.06	0.090	0.319
MFCC$_{20dim}$ (UT)	ST-CMVN	42.8	1.68	**0.075**	0.334	**1.78**	**0.083**	**0.317**

Table 4.9: *Performance of speaker recognition systems based on bottleneck features on the NIST SRE 2010, condition 5, female task, with an UBM of 512 Gaussian components and 400-dimensional i-vectors. *For this case, the classification accuracy was 49.4%, but in the more difficult task of classifying 9824 triphone states compared to 2423 states used for other experiments.*

In the following sections, we comment on both the performance of the DNN as phone classifier and the final speaker recognition systems. The results in terms of speaker recognition performance can be compared to the performance of the baseline system based on MFCCs, which is shown in Table 4.8.

4.3.6.1. Frame Phone Accuracy of the DNN

In third column of Table 4.9, we can see the phone accuracy obtained for the validation set when training the DNN for the ASR task.

In terms of phone accuracy, we observed a degradation in performance when the derivatives are not included in the input feature vectors (MFCC$_{20dim}$ experiment). However, it should be mentioned that even without the derivatives, the context is taking into account since frames are stacked in the preprocessing of the input to the DNN (see Section 4.3.4).

Moreover, we see that the ASR features (with per utterance mean normalization) yield better performance in terms of phone accuracy than the MFCCs since they are expected to be optimized for ASR. As we will comment on later, this does not lead to a better performance in the speaker recognition task.

To see whether degradation in phone accuracy was due to the change between ASR features and MFCCs, or to the normalization (utterance CMN applied to ASR features or ST-CMVN applied to MFCCs), we carried out a experiment using ASR features normalized with to ST-CMVN, and in that case, the phone accuracy decreased to 47.56% on the validation set.

Finally, it should be noticed that experiments denoted by "UT" (under-trained) are those in which the training of the network was interrupted even when improvements on validation still existed (i.e. training was stopped few epochs before the convergence). We did this in order to verify the hypothesis of poor correlation between phone accuracy of the DNN and discriminative power of the resulting bottleneck features for the task of speaker recognition.

4.3.6.2. Speaker Recognition Results

1. ASR Optimized Features

   In the experiments based on ASR features as the input to DNN, applying short-term MVN (typically used for features in speaker recognition) on top of the resulting bottleneck features yields a slight improvement in performance (∼10% relative).

   However, even though the phone accuracy reaches the highest values with these ASR features, bottleneck features obtained from these DNNs do not seem to be as discriminative as the ones obtained with DNNs trained using MFCCs optimized for the speaker recognition task.

   This is also supported by experiment in first row of Table 4.9. In this experiment, the DNN was trained to classify 9824 triphone states (four times more than in the rest of the experiments), and the phone accuracy was 49.4%. However, the resulting bottleneck features provided similar results that the experiment with the same ASR features, but less triphone states as the DNN outputs.

   Even so, these experiments based on bottleneck features outperform the baseline system (see Table 4.8).

2. Speaker Recognition Optimized Features

   The bottleneck features provided by DNNs trained using MFCC parameterization seem more discriminative for the speaker recognition task.

   Using these MFCC features as input to the network, different experiments have been carried out. Opposite to what was observed with the ASR features, when MFCCs with ST-CMVN are used as the input to the DNN, normalizing the resulting bottleneck features did not help or even resulted in slight degradation in performance.

   Moreover, in the experiments marked as $\text{MFCC}_{\Delta+\Delta\Delta}$ in the table, we used a 60-dimensional vector of 20 MFCCs with derivatives (Δ and $\Delta\Delta$), while just the 20 MFCCs were used in the experiments denoted by MFCC_{20dim} (all short-term cepstral mean and variance normalized). Comparing these two rows of Table 4.9, we can see that adding the delta coefficients seems not to increase or even decrease the performance. It should be noticed that even without the derivatives, the context is taken into account due to the staking of frames done at the preprocessing of the input. These 20-dimensional feature vectors got worse phone accuracy but resulted in the best speaker recognition performance, so redundancy introduced by the derivatives helped only in terms of phone discrimination but not in speaker recognition. We see again that better ASR performance (in terms of phone accuracy) does not necessarily correspond to better speaker recognition performance. The hypothesis might be that a DNN optimized for the best discrimination among phoneme states would lead to loosing relevant information for speaker recognition.

	EER (%)	**DCF$_{08}^{min}$**	**DCF$_{10}^{min}$**
BaselineFull	1.99	0.104	0.383
MFCC$_{\Delta+\Delta\Delta}$	1.62	0.065	0.220
MFCC$_{20dim}$	**1.46**	**0.057**	**0.209**
BN+MFCC$_{\Delta+\Delta\Delta}$	**0.96**	**0.042**	**0.146**
BN+MFCC$_{20dim}$	1.26	0.051	0.216

Table 4.10: *Comparison of performance on the NIST SRE 2010, condition 5, female task for large-scale system: UBM of 2048 Gaussian components, 600-dimensional i-vectors.*

Using the best configuration, we see relative improvements up to ∼37% in terms of EER with respect to the baseline system.

3. "Under-trained" DNN Experiments

   In order to verify the hypothesis mentioned before, the last two rows of Table 4.9 show results of DNNs whose training has been stopped before reaching the optimal performance for ASR task (stopped few epochs before the convergence). For those DNNs, results in speaker recognition task give similar or even better performances even though the results did not reach the best values in term of the phone accuracy. Therefore, we see that suboptimal training of DNNs for ASR can result in better feature extractors (DNN with bottleneck layer) for speaker recognition.

4. Full Speaker Recognition System Results and Concatenation of bottleneck features and MFCC

   Finally, a comparison in performance between large-scale speaker recognition systems (UBM with 2048 Gaussian components, and 600-dimensional i-vectors) can be seen in Table 4.10 for the best experiments described above (bottleneck features from MFCC-based DNNs). We see a relative improvement up to ∼27% in terms of EER when using bottleneck features from a DNN trained with ST-CMVN MFCCs without derivatives (same DNN as in the experiment shown in the fourth row of Table 4.9, but with large-scale system).

   In the last two rows of Table 4.10, we also show results using bottleneck features (BN) concatenated with MFCCs (approach that was used in [Richardson *et al.*, 2015b]), which provided the best performance (up to ∼52% of relative improvement in terms of EER). The bottleneck features used for this concatenation were the ones that provided the best performance in speaker recognition (from a DNN trained with ST-CMVN 20-dimensional MFCCs, row 4 in Table 4.9).

4.3.6.3. Further Discussion

According to results of this analysis, we see that suboptimal DNNs for ASR can provide better bottleneck features for speaker recognition than fully optimized DNNs for the speech recognition task. In order to further analyze that idea, apart from the "under-trained" experiments, we trained a DNN including a new hidden layer (with 1500 hidden units) between the bottleneck layer and the output layer (i.e. having 5 instead of 4 hidden layers). In that experiment, the phone accuracy was higher than in the rest of the experiments, but again, we did not observe any improvement in the speaker recognition performance.

Our hypothesis is that, since bottleneck features are discriminatively trained for phoneme recognition, they should suppress the information about speaker. We believe that the main benefit of using such features is that they lead to more sensible clustering of the acoustic feature space when training GMM-UBM (i.e. GMM components roughly corresponds to phonemes). This is also supported by our experiments using bottleneck features just for alignment of frames to UBM components, while the sufficient statistics for i-vector extraction are collected using MFCCs [Matějka *et al.*, 2016]. Therefore, for a good speaker recognition performance, we need bottleneck features, which already provide good clustering, but at the same time do not suppress too much of the speaker information.

4.4. Chapter Summary

In this chapter we have analyzed how different variations in the DNN used as bottleneck feature extractor for language and speaker recognition affects the performance of consequent i-vector systems trained on top of this new frame-by-frame representation. This can be seen as a step towards the direction of understanding the abstract representation learned by DNNs from the speech signal, which is not easily interpretable.

Therefore, our first analysis for the task of language recognition constitutes a systematic study on different configurations of the DNN, where we varied the number of hidden layers, the position of the bottleneck layer in the DNN and the size of this layer. Then, we show the effect of different representations obtained by the network both in terms of phoneme state frame accuracy of the DNN and language recognition (EER_{avg}). We see that the performance of the DNN in terms of phoneme state classification do not correspond with the best performance of the resulting bottleneck features for language identification in the i-vector pipeline. This is caused by a combination of effects. Firstly, the better the DNN classifies the phoneme states, ideally, the more language information is dropped from the resulting bottleneck representation. Moreover, the database used to train and evaluate DNN performance is similar while the database used to evaluate LID performance is different to those used to train and evaluate DNN performance, which influences as well the differences in tendencies of performance of DNN and language recognition systems. Finally, this bottleneck feature based language recognition system is compared to a reference system, following the same i-vector backend but based on MFCC-SDC parameterization of the input audio signal. This bottleneck feature approach shows relative improvements of about 36% for 3 seconds test segments, 44% for 10 seconds long segments and 41% when testing in 30 seconds fragments (increasing with duration), with respect to the reference system for the case of the test-development dataset (matched conditions). When testing this approach on the evaluation data of the NIST LRE 2015 (mismatched conditions), a degradation in relative improvement of the bottleneck features replacing the traditional parameters is observed, caused mainly by the mismatch existing between training and test data in this case. However, bottleneck feature based language recognition approach still outperforms the classical approach based on acoustic features (MFCC-SDC) used as reference in this section, with a relative improvement of approximately 8% on average.

In the second analysis concerning bottleneck feature-based speaker recognition, we studied whether not fully optimized networks trained for ASR could provide better bottleneck features for speaker recognition. Then, we analyzed the influence of different aspects (input features, short-term mean and variance normalization, "under-trained" DNNs) when training DNNs to optimize the performance of speaker recognition systems based on bottleneck features. We evaluated the performance of the resulting bottleneck features in the NIST SRE 2010, condition 5, female task. From the results obtained in this study, we observe that the best features for ASR task do not necessary perform the best when training a network, which is used as feature extractor for speaker recognition. Even though the phone accuracy of the DNN can increase with these features (ASR features), the best performance in speaker recognition was obtained using the typical MFCCs as used for speaker recognition tasks. According to the results, applying ST-MVN to the MFCCs before training the DNN yields the best performance, and performing that normalization on top of the bottleneck features helps just when input features to the DNN are those optimized for ASR (ASR features with CMN per utterance). Moreover, the performed experiments do not show much correlation between the frame-by-frame phoneme states classification and the ability of the resulting bottlenecks to discriminate between speakers: the best phone accuracy does not yield the best performance in the speaker recognition task. For example, with just 20 dimensional MFCC feature vectors in which the derivatives have not been added (although context is included when preprocessing the input) we obtained the best results in speaker recognition, while the performance in phone accuracy degrades. Finally, using bottleneck features from a DNN trained on MFCCs with ST-CMVN, we obtained up to 37% of relative improvement with respect to the baseline system (i-vector based on MFCCs).

Chapter 5

Utterance Level Representation: DNN-based Embeddings

One of the last proofs of success of Deep Neural Networks in the fields of speaker and language recognition is their use as *embedding* extractors. In this context, an embedding is understood as an utterance level representation obtained from a DNN trained discriminatively for the task of speaker or language identification. It is a fixed-length representation of the input audio segment, regardless its duration, similar to traditional i-vector representation in that sense, but whose aim is to capture mostly relevant information for the target task.

In this chapter, we describe the concept of *embedding* and their success in speaker recognition in previous works. Then, we present our contribution for the task of language recognition, which is based on their adaptation from previous works in speaker recognition and which showed impressive results outperforming classical strong i-vector systems over challenging datasets as the ones provided by NIST for the last Language Recognition Evaluations (LRE) 2015 and 2017.

5.1. Introduction to DNN-Embeddings

The so-called *embedding* representation as it is used in this chapter comes from the idea of obtaining a compact fixed-length representation of an utterance of variable duration, as opposed to the frame-by-frame representation of the signal such as classical MFCCs or PLPs or the bottleneck features described in previous chapters.

Moreover, unlike i-vector, which is already a fixed-length representation of an audio segment of variable duration, these embeddings are obtained with a DNN trained for the target task capturing information relevant to the task (speaker or language recognition) instead of variability from different sources, useful or not for the end system.

In this section, these DNN-embeddings are described and previous related work is presented.

5.1.1. Concept of Embeddings

As it has been mentioned previously, bottleneck features learned by means of DNNs have become the common frame-by-frame representation of the speech signal in state-of-the-art systems for speaker [Garcia-Romero and McCree, 2015; Lozano-Diez *et al.*, 2016; Yaman *et al.*, 2012] and language recognition [Fér *et al.*, 2015; Jiang *et al.*, 2014; Lozano-Diez *et al.*, 2017; Matějka *et al.*, 2014a; Richardson *et al.*, 2015a; Song *et al.*, 2013]. However, the duration of speech recordings widely varies, and so do their frame-wise representation, fact that poses a challenge in consequent modeling.

This shortcoming has been traditionally handled by the *i-vector* in the speaker and language recognition community, which compacts the utterance representation in a fixed-length vector. Even so, these i-vectors are generated by a model based on factor analysis, and corresponding GMM-UBM and total variability subspace are trained in an unsupervised way. Their aim is to capture information about sources of variability in the training data, but this information is not necessarily relevant to the target application.

Recently, the concept of *embedding* as a compact fixed-length representation of an utterance has emerged as a competitor to i-vectors in speaker and language recognition. This utterance-level representation is extracted from a sequence summarizing neural network trained discriminatively for the objective task, and it is meant to capture mostly information relevant to that given task.

A generic architecture for these DNNs based on embeddings is depicted in Figure 5.1.

Generally speaking, the DNN can be split into two main parts separated by the pooling layer.

The first part of the DNN (up to the pooling layer) would work on a frame-by-frame basis. It would take as input a frame-wise representation of the signal, such as MFCCs or bottleneck features for the case of speaker or language recognition. Then, a number of hidden layers would follow and will ideally capture the information contained in this frame-by-frame representation of the input.

Following this frame-level part of the network, a pooling (or sequence summarizing) layer will be added in order to summarize the information outputted by the last layer of this first part of the DNN. This pooling layer would compress the output values of the previous layer, which outputs values for each frame in a given sequence, into a single vector of values per sequence. This pooling mechanism might be the computation of some statistics such as mean or standard deviation over time.

The second part of this architecture would connect the output of the previous pooling layer to the output layer of the network. This part might consist of one or more hidden layers working already on utterance level, whose outputs can be used as *embeddings* modeled by some other backend.

During training, the architecture described would make use of an output layer which outputs the posterior probabilities for the given set of classes, such as speaker or language identities.

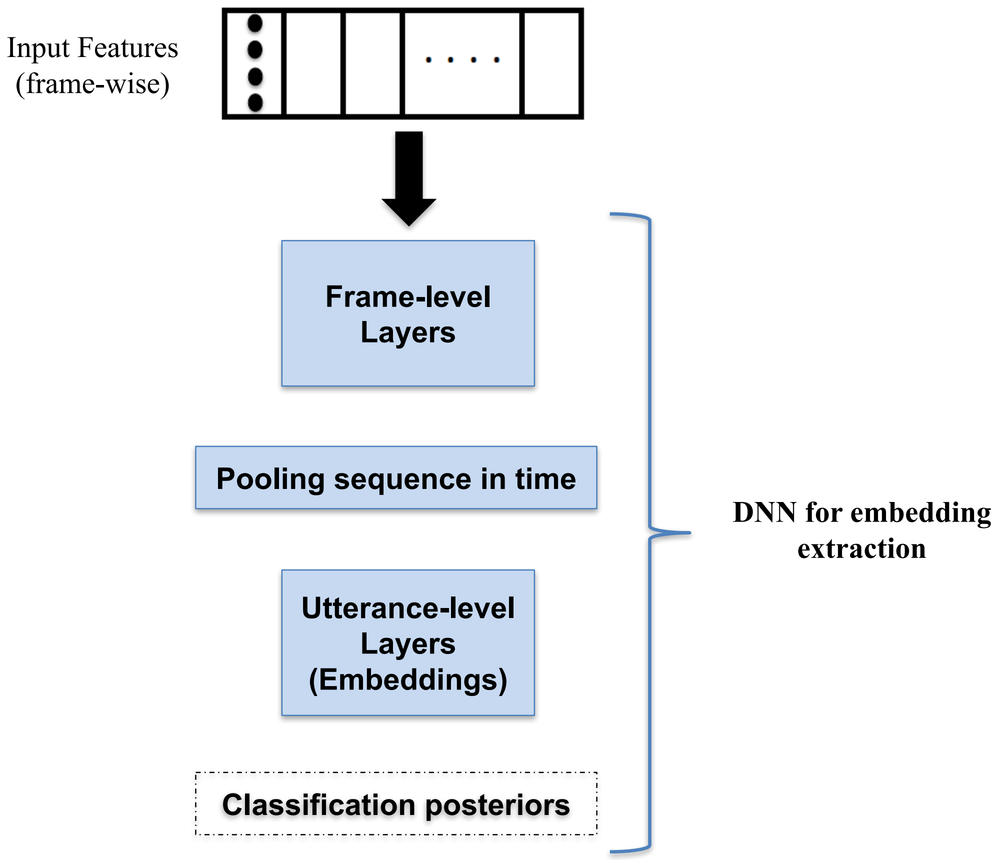

Input Features
(frame-wise)
Frame-level Layers
Pooling sequence in time
Utterance-level Layers (Embeddings)
Classification posteriors
DNN for embedding extraction

Figure 5.1: *Generic architecture of a DNN used as embedding extractor.*

However, this output layer will not be used during the embedding extraction stage, in the context of language and speaker recognition based on embeddings, since the DNN is used to obtain a representation of the signal, and not as end-to-end system which directly performs the target task. For embedding extraction, a given sequence would be forwarded through the network up to the embedding layers, and the resulting utterance representations would be modeled by a backend to perform the final task.

5.1.2. Prior Related Work

Several attempts to used DNNs to replace GMM-UBM and i-vectors in order to obtain an utterance level representation (*embedding*) have recently been published in the literature for the fields of speaker and language recognition.

For instance, this idea is explored for text-dependent speaker recognition in [Variani *et al.*, 2014] and in [Heigold *et al.*, 2016]. In these works, the information is summarized by means of the mean over the frame-wise outputs of one or more of its layers. A feedforward DNN is used in [Variani *et al.*, 2014], meanwhile a recurrent neural network (RNN) is used in [Heigold *et al.*, 2016].

Regarding language recognition, similar ideas have been also applied. For example, outputs of hidden layers are averaged over time and stacked together in [Li *et al.*, 2016], forming a representation for the utterance in a high-dimensional space. Moreover, in [Pešán *et al.*, 2016], one of the layers in the DNN averages frame-by-frame activations, and posteriors produced by the network are used for language identification as end-to-end approach. Also, [Gelly and Gauvain, 2017] presents a DNN-based system which learns language dependent vectors with angular proximity loss function.

Recently, a fairly simple architecture has been developed for text-independent speaker verification in [Snyder *et al.*, 2017]. Here, the first frame-by-frame part of the DNN is based on Time Delay Neural Networks (TDNN), in order to exploit temporal context of sequences. Then, embeddings are obtained as outputs of two hidden layers after the pooling layer, where not just mean but also standard deviation are computed over time. The whole network is then trained with multi-class cross-entropy for speaker identification over a given set of speakers. Subsequent modeling of embeddings with Probabilistic Linear Discriminant Analysis [Prince, 2007] (PLDA) achieves a performance comparable to i-vectors in the speaker verification context.

5.2. DNN-based Embeddings for Language Recognition

Motivated by the success of DNN-embeddings or similar ideas of prior research works presented in the previous section, we developed a language recognition system based on DNN-embeddings, following the approach used for speaker verification in [Snyder *et al.*, 2017].

Unlike [Pešán *et al.*, 2016], we used the extracted embeddings (instead of posteriors) to train a generative model for classification (Gaussian Linear Classifier, GLC). Our architecture is also simpler than the one presented in [Li *et al.*, 2016], with smaller embedding layers, and yields better results.

The proposed language recognition system based on embeddings and the changes introduced with respect to the previous work for speaker verification [Snyder *et al.*, 2017] are presented in the following sections. Our proposed system has been explored and tested on two challenging frameworks provided by NIST for the last two Language Recognition Evaluations on 2015 and 2017.

5.2.1. Proposed DNN Embeddings for Language Recognition

As we have mentioned, we developed a DNN-based language recognition system with embeddings, following the approach used for speaker verification in [Snyder *et al.*, 2017], in which we introduced several changes.

5.2.1.1. Input Features: Stacked Bottleneck Features

A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through a DNN and reading off the vector of values at the bottleneck layer.

In these experiments, we feed the embedding DNN with the so-called *stacked* bottleneck feature (SBN) vectors.

In our case, they are extracted from a cascade of two DNNs trained for the task of automatic speech recognition (ASR). Thus, the bottleneck feature vector output by the first network is stacked in time, defining context-dependent input features for the second one (hence the term "stacked"). The input features for this first DNN are 24 log Mel-scale filter bank outputs augmented with 2 fundamental frequency features based on [Talkin, 1995], resulting in a 26-dimensional feature vector. Then, mean subtraction is applied at the utterance level, and Hamming window followed by DCT consisting of 0th to 5th base are applied on the time trajectory of each parameter resulting in $(24 + 2) \times 6 = 156$ coefficients to feed the first network. The bottleneck outputs from the first network (80 dimensional) are sampled at times $t - 10$, $t - 5$, t, $t + 5$ and $t + 10$, where t is the index of the current frame. The resulting 400-dimensional features are then used as inputs for the second stage DNN.

DNNs used as bottleneck feature extractors have the same architecture: three hidden layers with 1500 hidden units each, a bottleneck layer and an output layer. The dimensionality of the bottleneck layer is set to 80 for the first network and either 30 or 80 (depending on the experiment) for the second one. For the experiments with monolingual bottleneck features (SBN30), we trained the networks with Fisher English, and with 9824 outputs (triphones). For experiments with multilingual bottleneck features (SBN80, ML), we used BABEL dataset with 17 languages and 15558 outputs (tied triphones per each language). This multilingual network is trained using block softmax as output layer [Fer *et al.*, 2017].

The outputs from the bottleneck layer of the second DNN (referred to as SBN) are the final features used in our experiments as input vectors for both embedding and i-vector systems.

5.2.1.2. Architecture

Our architecture follows the general DNN-embedding structure described in Section 5.1.1.

In particular, for the first part of the DNN we used two bidirectional long short-term memory (BLSTM) recurrent layers followed by a fully connected layer. We used recurrent layers based on LSTM units due to their success in related works ([Gonzalez-Dominguez *et al.*, 2014; Zazo *et al.*, 2016a]) and its ability to deal with temporal information of the signal, especially over short utterances. In [Snyder *et al.*, 2017], Time Delay Neural Network architecture (TDNN) was used instead in this part of the network.

Then, a subsequent pooling layer computes mean and standard deviation statistics over the frame-by-frame outputs of the previous layer, summarizing the information of a given input sequence.

Two more fully connected layers are added after this pooling layer, already forming the second part of the network that works in an utterance level basis and whose output values will be later used as embedding representations of the input utterance. We used a sigmoid activation function as a non-linearity for all hidden units.

Finally, the output consists of a softmax layer that provides a vector of language posterior

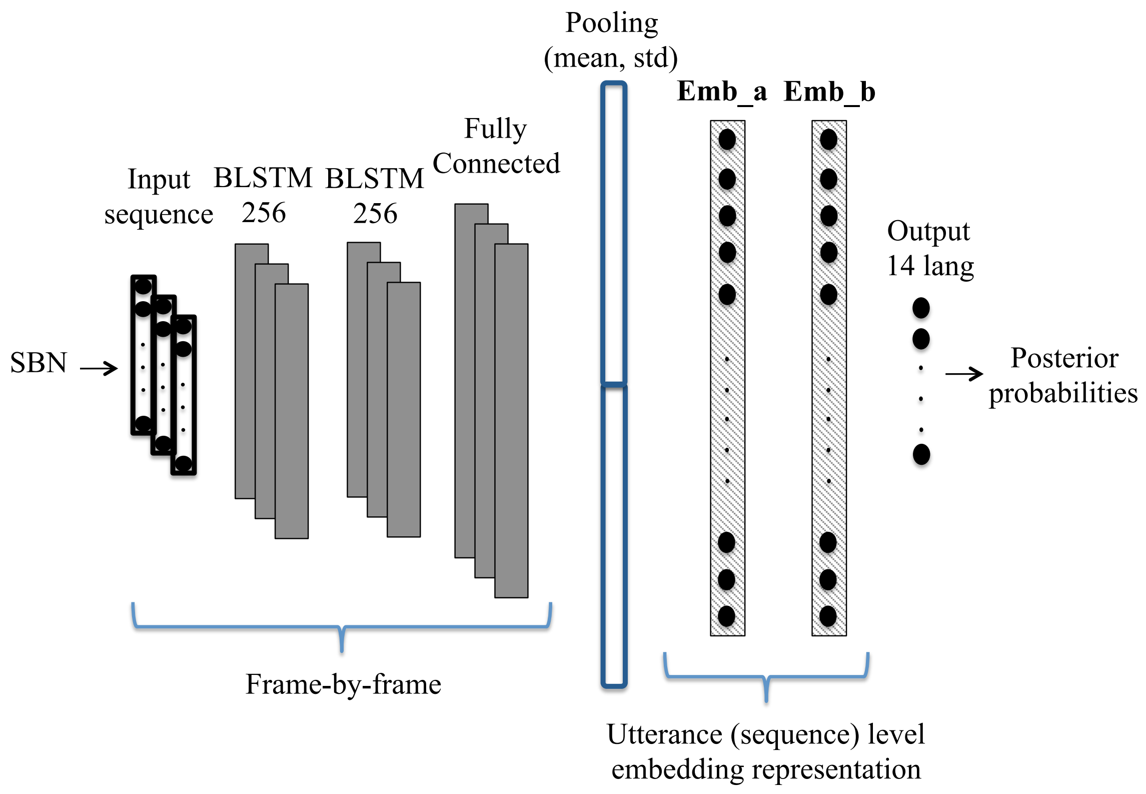


Figure 5.2: *Architecture of the proposed embedding DNN for language recognition. The size of the layers that are not specified in this figure varies according to the experiment.*

probabilities for each utterance. The size of this layer is set to the number of target languages involved in each experimental setup.

An example of this architecture is depicted in Figure 5.2.

5.2.1.3. Training

The DNN used to extract the embedding representations of utterances is first trained to classify sequences among the given set of target languages. The loss function that is optimized is multi-class cross-entropy, and this optimization is done via Adam optimizer [Kingma and Ba, 2014].

To reduce over-fitting, dropout was used in the recurrent BLSTM layers, including also dropout for the gates connections. For experiments with larger architectures, dropout was also used in the rest of the layers while training.

The stopping criteria was a maximum number of iterations (epochs) and the final model was selected according to the best accuracy on the validation set.

During training, gradients are estimated over batches. Each batch is composed of a different number of sequences depending on the experiment setup, but all sequences (except if the opposite is specified) were set to a fixed length of three seconds of speech.

Validation segments were also split into 3 second sequences.

5.2.1.4. Embedding Extraction

In order to obtain the embedding representation from the previously trained DNN, the sequence of input features for each segment are forwarded through the DNN up to the embedding layers. Thus, the output layer is not used in this stage.

It should be pointed out that even though we fed the DNN with fixed length sequences during training, this is not done for the embedding extraction step. In order to obtain the embedding-based representation for each utterance, we do not constrain the length of the given segment, but instead we forward the whole segment to obtain one embedding per utterance from each embedding layer.

Finally, the extracted embeddings are used either independently or concatenated as a fixed-length utterance representation for the language identification backend performed by means of a Gaussian Linear Classifier (GLC) in our system.

5.2.2. Reference i-vector System

In order to compare the performance of embeddings for the language recognition task with state-of-the-art i-vector based systems, we show results of an i-vector system that follows the classical i-vector pipeline [Martínez *et al.*, 2011] for LID.

In particular, a diagonal-covariance UBMs was trained using the same stacked bottleneck features that we used to train the embedding DNN, described in Section 5.2.1.1.

Then, a total variability subspace was trained on top of the training data, and i-vectors were extracted and further modeled by the backend described in the next section.

5.2.3. Language Identification Backend

5.2.3.1. Gaussian Linear Classifier

The Gaussian Linear Classifier (GLC) is a simple and commonly used backend for language identification [Cumani *et al.*, 2015; Martínez *et al.*, 2011].

This backend is used in our experiments on top of both i-vectors and embeddings, in order to obtain the vector of class-conditional log-likelihoods for each segment.

The model of each language is represented by a Gaussian distribution with mean computed over i-vectors or embeddings of each given language and a shared covariance matrix that is computed over all training data as a weighted average of within-class covariance matrices.

5.2.3.2. Calibration and Fusion

After scoring, our score vectors obtained as outputs of the GLC are pre-calibrated. Also, we present a score level fusion i-vector and embedding systems which are previously pre-calibrated.

For the purpose of pre-calibration and fusion, a simple solution was chosen to avoid overtraining. Thus, a multi-class logistic regression model is trained using the scores (loglikelihoods) from the development set.

For pre-calibration, each individual system has a trainable scale factor and an offset vector. For fusion, each system gets a single trainable scale factor, while every language gets a trainable score offset. The parameters are trained via optimizing prior-weighted multi-class cross-entropy, using an uniform (flat) prior over all target languages for both pre-calibration and fusion.

5.2.4. Analysis on NIST LRE 2015

In this section, we analyze the performance of our proposed DNN-embedding system for language recognition on the NIST LRE 2015 through different experiments. First, we analyze the performance achieved with embeddings of different size and compare them with the reference i-vector system. Then, encouraged by the results of smaller embeddings, we analyze the effect of further reducing the dimensionality with PCA. Given the fact that our DNN is trained directly for the given task, we also analyze the performance when taking DNN posteriors as scores, i.e, as an end-to-end language recognition system. Finally, we perform a score level fusion between the reference i-vector system and the embedding-based approaches and we show that the DNN was able to extract complementary information and our systems fuse well with the i-vector baseline.

5.2.4.1. Dataset Description

For this set of experiments, we report our results on the challenging NIST LRE 2015 benchmark [NIST, b]. The data that we used for training the DNN and subsequent Gaussian Linear Classifiers are composed of the dataset shipped by NIST for the *fixed* condition of the evaluation. Our bottleneck feature extractor was on the other hand trained on the Fisher English corpus instead of the allowed Switchboard.

The NIST LRE 2015 dataset consists of recordings from 20 different languages, clustered into six groups according to language similarities [NIST, b], as we show already in Table 3.10.

The training dataset provided by NIST was split into two disjoint parts: training and development datasets [Plchot *et al.*, 2016]. These datasets were created by randomly selecting 60% for the training part and 40% for the development set. The segments belonging to the development set were further split into short cuts of different durations that contain from 3 to 30 seconds of speech. After splitting the data and dividing the development segments into cuts, we ended up with 3042 segments (248 hours of speech) in training set and 42295 segments (146 hours of speech) in development set.

Moreover, a balanced training subset (up to 15 h per per language) was randomly selected and used for the DNN training (and UBM training of the reference i-vector system) in order to partially compensate big differences in the amount of available training data per language. However, no data augmentation was performed for the classes with less than 15 h of speech. The resulting dataset contains 2316 segments out of original 3042. Furthermore, we randomly

	Number of units per layer					
System	BLSTM_1	BLSTM_2	Fully Connected	Pool	Emb_a	Emb_b
DNN_1	256	256	256	512	512	300
DNN_2	256	256	256	512	256	150
DNN_3	256	256	256	512	128	75

Table 5.1: *DNN configuration for each architecture used on the NIST LRE 2015 dataset.*

selected 979 segments (up to 50 per language) from the development set for the purposes of cross-validation and model selection during DNN training.

Finally, we used the full train set (3042 segments) to train the GLC and the development set was used to obtain calibration parameters. The systems were evaluated on the full LRE 2015 evaluation dataset, which consists of 164334 test segments of different durations.

5.2.4.2. Embedding System Configuration Parameters

The architecture of the DNN for embedding extraction explored in this section follows the general structure described in Section 5.2.1.2, with differences in size per layer for each experiment as shown in Table 5.1.

For all the embedding systems, the input to the network corresponds to 30-dimensional stack bottleneck features from a cascade of DNNs trained with Fisher English corpus. The output layer is a 20-dimensional softmax layer that provides a vector of language posterior probabilities for each utterance.

A dropout rate of 30% was used in the BLSTM layers for both cells and gates. Gradients are estimated over batches of 200 samples. The maximum number of iterations (epochs) for all experiments was set to 200.

During training, both train and validation segments were split into 3 second sequences (300 frames) with no overlap and no stacking of frames. This resulted in 173109 samples (sequences) for training (approximately 144 h of speech), and 3631 samples for validation (about 3 h of speech).

5.2.4.3. I-vector System Configuration Parameters

The i-vector system used as reference in this setup consists of an UBM with 2048 Gaussian components trained on the same 30-dimensional stacked bottleneck as the ones used for the embedding systems. Both GMM-UBM and i-vector extractor were trained using the 15 h balanced dataset. The total i-vector extractor was trained in 10 iterations and the dimensionality of i-vectors was set to 600. The reference system is a strong i-vector system whose performance is state-of-the-art.

	$\mathrm{C_{avg}} \times 100$		
System	dev*	eval	eval*
Reference i-vector	4.54	**16.93**	**14.91**
DNN_1 emb_a (512)	5.92	20.26	18.68
DNN_1 emb_b (300)	**4.47**	19.03	16.25
DNN_1 emb_a_conc_b (812)	5.77	20.04	17.87
DNN_2 emb_a (256)	4.71	18.82	16.52
DNN_2 emb_b (150)	4.75	19.04	16.71
DNN_2 emb_a_conc_b (406)	4.88	19.19	16.84
DNN_3 emb_a (128)	5.51	19.02	17.01
DNN_3 emb_b (75)	4.76	19.05	16.62
DNN_3 emb_a_conc_b (203)	5.37	19.30	16.93

Table 5.2: *Comparison of i-vectors with embeddings of different size.*

5.2.4.4. Evaluation Metrics

The performance of the LID systems in this set of experiments is shown in terms of C_{avg} as defined for the NIST LRE 2015 [NIST, b].

We report also starred versions of the C_{avg} (dev* and eval*). This means that instead of being trained on a separate held-out calibration set (development set), the calibration parameters were trained on that set itself. The difference between starred and non-starred C_{avg} suggests the dataset shift between development and evaluation sets or calibration problems.

5.2.4.5. Experiments and Results

In this section, we present the results of our proposed embedding system and the corresponding baseline i-vector system.

First, we analyze the performance achieved with i-vectors and embeddings of different size in Table 5.2. Inspired by the DNN architecture used in [Snyder *et al.*, 2017] for text-independent speaker verification, we started our experiments with two embedding layers of sizes 512 and 300 respectively. We experiment with using the two embeddings independently and stacking both of them.

When looking at the results on dev set in Table 5.2, we see that embeddings achieved similar performance as the very competitive i-vector baseline. The differences in performance on the evaluation set suggest that the embeddings extracted from the discriminative model are more sensitive for the domain shift which is happening between dev and eval datasets. Similar differences between C_{avg} and C^*_{avg} for both embeddings and i-vectors suggest that both systems received comparable calibration.

	$C_{avg} \times 100$		
PCA dim	dev*	eval	eval*
No PCA (812)	5.77	20.04	17.87
500	5.02	19.26	16.55
300	4.38	18.79	15.99
100	**3.89**	**18.67**	**16.05**
25	5.67	19.98	17.91

Table 5.3: *Applying PCA to concatenated DNN_1 embeddings.*

	$C_{avg} \times 100$ (eval)		
System	None	100	25
DNN_1 emb_a_conc_b (orig 812)	20.04	18.67	19.98
DNN_2 emb_a_conc_b (orig 406)	19.19	18.11	**17.44**
DNN_3 emb_a_conc_b (orig 203)	19.30	18.70	18.13

Table 5.4: *Applying PCA to concatenated embeddings.*

By comparing the results of embeddings of different size, we see a better performance of DNN_2 system which produces embeddings of the half size w.r.t. DNN_1 (DNN_3 further halves the embeddings of DNN_2). This behavior was expected as compared to the system in [Snyder *et al.*, 2017] we deal with a closed-set problem and much lower number of classes. These results also suggest that the embeddings of larger size contain more detrimental information about channel as all networks reached the same performance on the training data. Motivated by this observation, we explore further dimensionality reduction via Principal Component Analysis (PCA) in Tables 5.3 and 5.4.

As shown in Table 5.3, reducing the dimensionality of concatenated embeddings from DNN_1 by PCA improves the results. The best performance is achieved by keeping the first 100 dimensions which results in approximately 7% relative improvement on eval.

Finally, we compare the PCA post-processing using the concatenated embeddings from all DNNs in Table 5.4. We can observe that with smaller embeddings, we are able to reduce the dimensionality further to 25 and still gain some performance improvement. We achieved the best results with embeddings from DNN_2 whose concatenated dimensionality 406 is close to the typical i-vector (400 or 600). In terms of C_{avg}, we achieved the performance of 17.44% which is already close to our i-vector baseline with 16.93%.

As our DNNs for embedding extraction are in fact trained to discriminate between languages, and softmax layer outputs corresponding posterior probabilities for each of the 20 target languages, we also analyze the use of these posteriors directly as scores. This approach corresponds to the end-to-end training and the results are summarized in Table 5.5. Although results on the

	$C_{avg} \times 100$		
System	dev*	eval	eval*
Ref. i-vector	4.54	16.93	14.91
DNN_1 posteriors	4.90	20.37	17.26
DNN_2 posteriors	3.94	19.68	16.64
DNN_3 posteriors	5.00	19.76	16.95

Table 5.5: *Performance of individual DNN systems when taking posterior probabilities as scores.*

	$C_{avg} \times 100$		
System	dev*	eval	eval*
(1) Ref. i-vector	4.54	16.93	14.91
(2) DNN_2 emb_a_conc_b + PCA (25)	3.67	17.44	15.86
(3) DNN_2 posteriors	3.94	19.68	16.64
Score fusion (1)+(2)	**2.99**	**15.69**	13.52
Score fusion (1)+(3)	3.04	16.41	**13.19**

Table 5.6: *Comparison of single systems and their fusions.*

dev set are similar to the i-vector system, this approach seems to generalize slightly worse than raw embeddings.

Comparing results in Tables 5.4 and 5.5, we can observe that PCA-reduced embeddings safely outperformed the system based on posteriors. Our results suggest that using embeddings together with a simple model (GLC) is a viable research direction, especially if we consider that the DNN does not necessarily have to be trained exactly on the target languages.

Finally, we present improvements obtained with a simple score level fusion (as described in Section 5.2.3.2) between the reference i-vector system and the best embedding-based system presented in previous sections (DNN_2 emb_a_conc_b + 25 dimensional PCA). We also show the fusion of the reference i-vector system and the posterior probability outputs from the network. Looking at Table 5.6, we can see that both fusions improve results, suggesting that these different approaches can extract complementary information from the same input features.

To conclude this analysis on the NIST LRE 2015, we presented a DNN architecture with embeddings for language identification where the DNN is trained to discriminate between languages and at the same time learns an utterance level representation of input segments. These fixed-length representations i.e., embeddings are further used to train a generative classifier (GLC). This is a novel approach for the problem of LID and it is in line with the research that has proven to be promising for speaker verification in [Snyder *et al.*, 2017]. Our results are comparable with a state-of-the-art i-vector system on the NIST LRE 2015 setup. Also, we would like to highlight

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 5.7: *Language and clusters of the NIST LRE 2017 dataset.*

that using embeddings instead of posteriors from the DNN provided better results, and points towards the direction of training more general DNNs i.e., trained on much larger and variable set of languages, that can provide more general embeddings usable across various LID tasks.

5.2.5. Submission for NIST LRE 2017

In this section, we briefly describe the submitted system for the NIST LRE 2017 by Brno University of Technology (Czech Republic) in collaboration with Politecnico di Torino (Italy), Universidad Autonoma de Madrid (Spain) and Phonexia (Czech Republic), and we will focus on the description of the DNN-based embedding subsystem included in the final submission, extending the study of this system after the evaluation period in Section 5.2.6.

We first present the dataset provided by NIST for this evaluation, followed by a description of the primary submission and the comparison of results between individual embedding and i-vector based subsystems.

5.2.5.1. Dataset Description

In this section, we describe the original database provided by NIST for LRE 2017 [NIST, a]. This benchmark consists of five clusters of similar languages, with a total of fourteen different languages, as shown in Table 5.7. Clusters and languages are similar to the ones used on the NIST LRE 2015, but some languages and the French cluster were discarded in this evaluation.

In particular, we focused on the primary (fixed) condition (except for the experiments with multilingual bottleneck features) and we used all data supplied by NIST (training and development) for this condition. All data was down-sampled to 8kHz.

For training the neural network used as bottleneck feature extractor, we used the annotated Fisher I and II databases provided for the evaluation, with three copies of noisy and reverberated variants of the original audio files (as it was done for all data in post-evaluation experiments presented in Section 5.2.6.1). For the experiments with multilingual bottleneck features, this

Systems	EVL (DEV)
[1] SBN30D ClusterDep 4096/800 (MGC)	18.68 (14.57)
[2] SBN30D 4096/800 NLPLDA (MGC)	19.80 (17.30)
[3] DNN embeddings (GLC)	24.07 (19.55)
[4] SBN80 4096/800 (iXtractor with FSH+SWB) (MGC)	21.35 (18.05)
Primary Fusion [1 + 2 + 3 + 4]	16.60 (12.96)

Table 5.8: *Results of the core systems and fusions for the submission to NIST LRE 2017.*

neural network was trained using the 17 languages from BABEL program dataset [1].

In order to train the embedding neural network, we used the set of training data released as such by the organizers, which consists of 16205 utterances. We added two thirds of the development data to this set, using the remaining third as a validation set. We cut long segments from the development set expanding it to 6090 segments. Therefore, 20301 segments were used for training the embedding extractor and 1994 were used for validation (model selection).

Regarding the final classification stage, the training data and the two thirds of the development set were also utilized to train the Gaussian Linear Classifier (GLC) used as backend for both embeddings and i-vectors. Calibration and fusion parameters were obtained using the whole development set (6090 segments).

Finally, the systems were evaluated on the full LRE 2017 evaluation dataset, which consists of 25451 test segments of approximately 3, 10 or 30 seconds of speech.

5.2.5.2. Primary Submission

In our primary submission to NIST LRE 2017, most of the systems are based on i-vectors and bottleneck features, but we also developed a subsystem based on DNN-embeddings. Results are summarized in Table 5.8. Results are shown in terms of the equalized actual cost defined as primary evaluation metric [NIST, a], for the actual evaluation set (EVL) and the held out test set that we used for system development during the evaluation period (DEV).

The primary submission consists on the score-level fusion (via logistic regression) of four subsystems. Three of them are based on different i-vector approaches, and one based on embeddings.

The DNN-embedding subsystem followed the architecture presented in Section 5.2.1, with the following configuration: we set the size of the BLSTM layers and the fully connected layer to 256, and the sizes of the embedding layers are set to 512 and 300 units respectively. This is the original architecture used for the analysis on NIST LRE 2015 analyzed in Section 5.2.4. The output layer is a 14-dimensional softmax layer that provides a vector of language posterior probabilities for each utterance. This embedding subsystem was improved and further analyzed after the evaluation period, and results are shown in Section 5.2.6.

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

The subsystem referred to as SBN30D ClusterDep 4096/800 in Table 5.8 is a cluster dependent approach, where 5 (as there are 5 language clusters defined in this setup) i-vector systems are fused. Individual UBMs (with 4096 Gaussian components) are trained only on data belonging to the specific cluster, on top of 30-dimensional SBN features (augmented with delta coefficients). However, the total variability matrix (800-dimensional subspace) is trained on a common training dataset for all individual cluster-dependents subsystems. This approach is the best individual system, although it is already a fusion of 5 subsystems.

In the second row of Table 5.8, we show results for an i-vector system from 30-dimensional SBN features (augmented with delta coefficients) where Non-Linear PLDA [Cumani and Laface, 2017] is applied to estimate a transformation of i-vectors (instead of using it to compute language recognition scores directly [Plchot *et al.*, 2018]), and a backend is trained on top of the transformed i-vectors for scoring.

Finally, in the fourth row of Table 5.8 we can see the results for the last i-vector system involved in the primary submission, where the i-vector extractor is trained on Fisher and Switchboard using 80-dimensional SBN features (multilingual). The UBM is composed of 4096 Gaussian components and 800-dimensional i-vectors are derived from the total variability model.

Regarding the backends, as in previous experiments with embeddings, the Gaussian Linear Classifier (GLC) was used as backend for scoring. However, for the rest of the systems based on i-vectors, used a Multi-Gaussian Classifier (MGC), which assumes that i-vectors of each language are generated by a combination of Gaussian distributions (in our case assumed to be three Gaussian components due to three different sources of data as provided for the evaluation: VAST, MSL14 and previous LRE). With this backend, at test time a language score is computed from a GMM whose components are the Gaussian distributions associated to the target language, assuming uniform weights over the data sources.

5.2.5.3. Post-evaluation Results for Primary Submission

In this section, we present some post-evaluation results obtained for the primary submitted individual systems and the primary fusion system that would result after fusing the improved systems

In Table 5.9, post-evaluation results on the development and evaluation datasets are shown. Comparing these results with Table 5.8, we see that we were able to improve all individual systems. From the results, we can immediately notice that the MGC classifier which gives us a good performance on the development set, did not outperform the GLC backend for the evaluation data.

Last two rows of Table 5.9 contain the fusion used as primary system for submission and the fusion of just i-vector based subsystems. We can see that each of the improved post-evaluation systems achieves better performance on evaluation data than the corresponding system used for the original submission (see Table 5.8 for comparison). On the contrary, the performance on the development set is, in general, worse. Surprisingly, the overall fusion is slightly worse also on the evaluation set. This is caused by a different behavior of the embedding system, which has

Systems	Difference	Post-evaluation EVL (DEV)
[1] SBN30D ClusterDep 4096/800	MGC $\rightarrow$ GLC	16.88 (17.94)
[2] SBN30D 4096/800 NLPLDA	MGC $\rightarrow$ GLC	19.30 (20.30)
[3] DNN embeddings	More data + data augmentation, **large** DNN	19.61 (16.14)
[4] SBN80 4096/800 (iXtractor with FSH+SWB)	MGC $\rightarrow$ GLC	18.79 (19.41)
Primary Fusion $[1+2+3+4]$	-	16.78 (15.14)
Fusion of i-vector systems only $[1+2+4]$	-	15.71(16.52)

Table 5.9: *Results of the core systems and fusions in post-evaluation of NIST LRE 2017. In the second column we include the differences with respect to the submitted system for the evaluation.*

improved in both datasets and received a large weight in the fusion compared to the best i-vector system. By removing the embedding system from the fusion we indeed obtain better results on the evaluation set. This behavior is most-likely caused by overtraining of our embedding system and points out that, compared to i-vectors, using these DNN-based architectures comes with a risk and requires more careful planning when designing the training datasets.

Further analysis of the DNN-embedding subsystem is performed in the next section. Some extended analysis of i-vector based systems was performed after the evaluation, and it is presented in [Plchot *et al.*, 2018]. However, we will not include this analysis since this Dissertation focuses on DNN-based approaches.

5.2.6. Post-evaluation Analysis of Embeddings on NIST LRE 2017

As part of the post-evaluation analysis after the participation in the last NIST LRE 2017, and after the promising results shown by embedding-based systems on the LRE 2015 framework, we continued with this research line exploring and analyzing DNN embeddings for language recognition on the most recent NIST LRE 2017.

In particular, in this section we describe the dataset used in the experimental part of this section and we analyze further improvements made after the evaluation period for the embedding subsystem submitted to this last NIST LRE 2017 [Plchot *et al.*, 2018]. Thus, we study the influence of increasing the size of our original system architecture, as well as the performance when using different input features and a wide range of augmented training data. Performance for different test segment duration is also analyzed.

We compared the system based on embeddings with a corresponding i-vector system, both trained on the same features, and we were able to gradually improve the performance of the embedding system via increasing the model size and the amount of training data until we clearly outperformed the system based on i-vectors. We achieved further improvement by a score-level fusion of both systems, i-vector and embedding, suggesting again that embeddings are able to extract complementary information to i-vectors.

5.2.6.1. Dataset Description: Data Augmentation

In this section, we describe how the original database provided by NIST for LRE 2017, has been augmented by adding noise and reverberation for our post-evaluation analysis of the embeddings subsystem.

In order to analyze how the data augmentation affects the performance of embedding systems, we used different techniques to obtain corrupted copies of the DNN training data (including the two thirds of development data used as training). The augmented data were then used together with the original audio files to train the DNN used for embedding extraction. In particular, we added noise and reverberation and we also changed the tempo (speed) of the recordings.

For the noisy dataset, we prepared a set of noises that consists of three sources of different types of noise:

- 272 samples taken from the Freesound library [1] (real fan, HVAC, street, city, shop, crowd, library, office and workshop).
- 7 samples of artificially generated noises: various spectral modifications of white noise + 50 and 100 Hz hum.
- 25 samples of babbling noises by merging speech from 100 random speakers from Fisher database using speech activity detector.

These noises were then added to original training data at SNR levels sampled from two ranges: 0-8 dB and 8-20 dB.

Regarding reverberation, we used a set that consists of real Room Impulse Responses (RIR) from several databases: AIR [AIR], C4DM [C4D; Stewart and Sandler, 2010], MARDY [MAR], OPENAIR [Ope], RVB 2014 [RVB], RWCP [RWC]. Together, they form a set of RIR that simulates different types of rooms: small rooms, big rooms, lecture room, restrooms, halls, stairs, etc. All room models have more than one impulse response per room, i.e. different RIR was used for source of the signal and source of the noise to simulate different locations of their sources.

We also changed the audio playback speed (tempo) to 0.9 and 1.1 of original speed. We used SoX tool which changes speed of audio files but keeps their original pitch.

Finally, these perturbations are combined so that reverberation is applied to audio files already corrupted with noise, or noise is added on top of audio files where the tempo has been modified (referred to as *noised tempos* in the experimental part).

5.2.6.2. Embedding System Configuration Parameters

In this set of experiments, we explored two architectures for the embedding-based DNN, both following the structure presented in Section 5.2.1.2. The specific configuration for each of

[1] `http://www.freesound.org`

	Number of units per layer					
System	BLSTM_1	BLSTM_2	Fully Connected	Pool	Emb_a	Emb_b
Small	256	256	256	512	512	300
Large	256	256	1500	3000	512	512

Table 5.10: *DNN configuration for the* small *and* large *architectures used on the NIST LRE 2017 dataset.*

the architectures referred to as *small* and *large* is presented in Table 5.10.

Most of the experiments are performed on top of 30-dimensional SBN feature from a monolingual (English) network (SBN30). We also trained some systems after adding their delta coefficients (SBN30D, deltas), i.e. using 60-dimensional input feature vectors. Moreover, we present some experiments where the DNN is trained using multilingual bottleneck features (SBN80, ML) of size 80, as mentioned in Section 5.2.1.1. Thus, the input layer varies its size (30, 60 or 80) depending on the experiment.

The output layer in all the embedding systems consists of a softmax layer with 14 units corresponding to the target languages involved in this task.

To reduce overfitting, dropout rate of 30% was used in all layers, including also dropout for gates on the recurrent BLSTM layers. The maximum number of iterations (epochs) for all experiments was set to 400, and the final model was selected according to the best accuracy on the validation set.

During training, gradients are estimated over batches of 210 samples. These batches are created randomly selecting 9 seconds of speech (3 fragments of 3 seconds of speech) from 70 different input audio files. The length of input sequences is set to 3 seconds of speech (300 frames). One epoch is completed when all the files from the input list are seen by the network. The training list is different depending on the experiment, and therefore, different number of hours are used for training depending on the available data for that experiment (amount of augmented data), ranging from approximately 50.7 to 558 h per epoch (from the experiments with just original audio files to the experiment with 11 copies). There is also a comparison between training with a balanced set where the amount of data per language is limited to 15 h and the full training list.

Validation segments are the same in all the experiments, and were split into 3 second sequences, resulting in 23782 samples for validation (about 19 h of speech).

5.2.6.3. I-vector System Configuration Parameters

We consider as reference system a competitive i-vector system with an UBM of 4096 Gaussian components, trained on the same 30 dimensional stacked bottleneck features after adding delta coefficients (SBN30D, deltas).

For the purposes of GMM and i-vector extractor training, we used only the original LRE

	$C_{primary} \times 100$	
Input features	2048/600	4096/800
SBN30	23.58	-
SBN30D (+deltas)	19.80	18.53

Table 5.11: *Results of i-vector reference systems comparing different input features (SBN30 with and without deltas), and size of UBM and i-vector dimensionality. GLC backend is used on top of the i-vectors.*

2017 training dataset (with no augmentation and without the two thirds of development data). The total i-vector extractor was trained in 10 iterations and the dimensionality of i-vectors was set to 800.

We also show results of a classical i-vector system with an UBM of 2048 Gaussian components and 600-dimensional i-vectors, to observe the effect of the increased size of the model in terms of LID performance. Furthermore, for this smaller system we also present the variant without including the delta coefficients to have an exact match of the input features with many of our embedding systems.

5.2.6.4. Evaluation Metrics

In order to compare the results of the experiments in this section, we use the primary metric for the NIST LRE 2017 $C_{primary}$ as defined in the evaluation plan [NIST, a], computed with the available NIST tool.

Moreover, we show some results using the C_{avg} metric used in the previous NIST LRE 2015 evaluation [NIST, b].

5.2.6.5. Experiments and Results

To establish baselines for our embedding systems, we list the results of the i-vector systems based on the same 30 dimensional SBN features in Table 5.11. As we have mentioned before, we present two variants with smaller and larger UBM and i-vector size. The performance of a larger system with SBN30D features is clearly the best and would be also among the best single systems we developed for NIST LRE 2017 [Plchot *et al.*, 2016]. For the smaller system we also present the variant without including the delta coefficients to have an exact match of the input features with many of our embedding systems.

Next, we present results of embedding systems with larger and smaller model and compare the three different input features and two training sets described in previous sections. Results are summarized in Table 5.12.

The two architectures used for these experiments, referred to as *small* and *large*, follow the structure shown in Figure 5.2. They consist of an input layer (with 30, 60 or 80 input units depending on the features), followed by two BLSTM layers (256 cells each) and a fully connected layer, which size is set to 256 for the *small* network and 1500 units for the *large* one. Then, after

	$C_{primary} \times 100$	
	Small (812)	Large (1024)
Max. 15h per language, SBN30	24.07	22.20
Full list, SBN30	22.18	19.86
Full list, SBN30D (+deltas)	21.69	18.95
Full list, SBN80 (ML)	23.41	20.18
Full list, SBN30 + 2 noises	20.60	19.03
Full list, SBN80 (ML) + 2 noises	20.62	18.32

Table 5.12: *Results comparing different input features (SBN), training data lists and architectures of the DNN. Both columns show results on stacked embeddings (a+b) with GLC backend.*

the pooling layer, the two hidden layers whose outputs are used as embeddings, have 512 and 300 units for the *small* network, and 512 units each for the *large* architecture. Output layers are both 14 dimensional layers, corresponding to the target languages of this setup.

As we can see, the *large* configuration outperforms the *small* one in all the cases. The stacked embeddings are 1024-dimensional for the big network, and 812-dimensional for the other, which might influence these gains. Moreover, increasing the size of the layer before pooling allows the network to capture more information from the frame based part, and these larger statistics that are further propagated through the DNN should contain a better summary for the utterance level side of the network.

In Table 5.12, we compare results on two different training lists. The first row shows results using a training list that constrains the maximum number of hours per language to 15 h, which was our submission for the NIST LRE 2017. This was done in order to partially compensate the unbalanced dataset available. However, using the full training list has proven to work better in our setup.

Furthermore, we explore three different input features. All of them are stacked bottleneck (SBN) features extracted as described in Section 5.2.1.1. We experimented with 30-dimensional SBN (SBN30), trained with just English data, and their 60-dimensional variant that contains delta coefficients (SBN30D). According to these results, including this temporal information at the input seems beneficial for the task, even though the architecture based on BLSTM is also taking into account the context of each frame given in the input sequence. These delta coefficients also helped the i-vector reference system. We also used 80-dimensional SBN from a multilingual network, which outperforms systems using SBN30 when noisy data is included in the training list. This suggests that increasing the input feature size (and therefore the whole model size) makes the network more prone to overfitting or simply too large to train on a smaller dataset and that can be partially compensated by augmenting the training data with their noisy versions. The last two rows of the table show that adding noisy data outperforms results with just original data for both architectures and different input features. Next set of experiments

Training data	$C_{primary}$	$\mathrm{C_{avg}}$
Original	19.86	3.97
+ 1 reverb (clean)	20.33	4.06
+ 1 reverb (noise 8-20dB)	19.01	3.90
+ 1 noise (8-20dB)	19.40	3.77
+ 2 noises	19.03	3.85
+ 2 tempos	20.06	3.97
+ 2 reverbs	19.40	3.99
+ 2 noised tempos	18.84	3.73
+ 2 tempos + 1 noise (8-20dB)	19.57	4.00
+ 2 noises + 2 tempos	19.37	3.81
+ 4 noised tempos	18.95	3.93
+ all (4) noises (8-20dB)	18.64	3.79
+ all (7) noises	18.69	3.82
+ all (10) augmentations	17.96	3.67

Table 5.13: *Results with different data augmentations for DNN training with GLC backend. Results are shown as Equalized $C_{primary} \times 100$ from NIST LRE 2017 and $\mathrm{C_{avg}} \times 100$ from LRE 2015.*

further explores and extends this training data augmentation.

In general, Deep neural networks are known to be prone to overfitting, which in our setup was supported by the existing gap in performance between the held-out subset separated from the development set (whose remaining two thirds were included in the training set for the embedding DNN) and the actual evaluation dataset.

Increasing the amount of training data partially solves that problem. In our case, we extended the training dataset by performing data augmentation through addition of noise, reverberation and tempo variations of original audio files as described in Section 5.2.5.1.

In Table 5.13, we compare the performance of embedding systems trained with up to 11 copies of the original training data with different augmentations. These results are also presented graphically in Figure 5.3 showing the $C_{primary} \times 100$ metric and grouped by number of copies. Although we will discuss results using the primary metric of the NIST LRE 2017, Table 5.13 also includes results on $\mathrm{C_{avg}}$ as defined in LRE 2015 for the sake of comparison with results split per test segment durations.

General trends show that increasing the number of copies of the data yields improvements in performance. In particular, adding any noisy version of the data (combined or not with other corruptions) for training the embedding extractor makes the system more robust against data mismatch and brings gradual performance gains. The only two cases in which data augmentation does not improve the system trained only on original data are the ones in which just reverberation

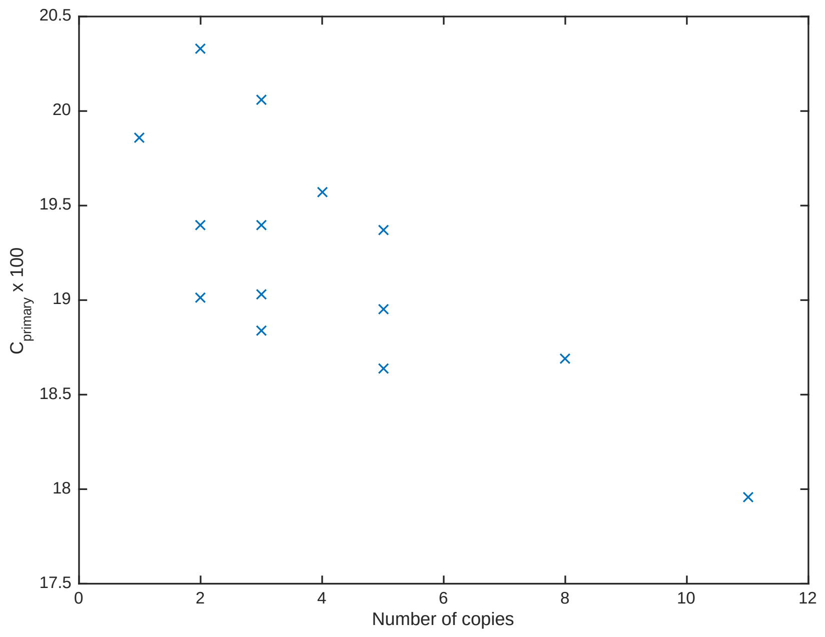


Figure 5.3: *Influence of the number of copies of the original data used to train the DNN in the performance ($C_{primary} \times 100$) of the resulting embeddings for LID.*

or tempo variations without any noise addition are performed over the original audio files. In these cases, the performance is even slightly degraded.

We have also performed a similar analysis for the i-vector system, where we did not see any benefits from data augmentation [Plchot *et al.*, 2018].

In order to evaluate how the embedding systems are influenced by the use of 3 second sequences for training, we show some examples of performance split by test segment durations (3, 10, 30 seconds and all pooled together) in Figure 5.4.

In this figure, we show the results in terms of $\mathrm{C_{avg}} \times 100$ and we experiment with the system from the fourth row in Table 5.13 (with one noisy copy of the data). Blue bars correspond to training with fixed-length 3 second sequences and yellow bars correspond to the system where each minibatch was selected to contain sequences of random duration between 2 and 3 seconds. Despite the restriction in input sequences length, the blue system still performs better even for the shortest duration condition (3s) and keeps a small performance gain also for longer durations. So far, we were not able to exploit variable training segment durations to achieve better performance or robustness of our DNN embedding architecture.

In Figure 5.5, we also present the comparison of performance between the best embedding system (the last row of Table 5.13) and the best i-vector system (SBN30D, 4096/800). The embedding system outperforms the i-vector system for all durations, increasing the gap in performance for shorter durations.

Finally, even though both i-vector and embedding extractors are trained on top of the same bottleneck features, these utterance representations are extracted with very different models,

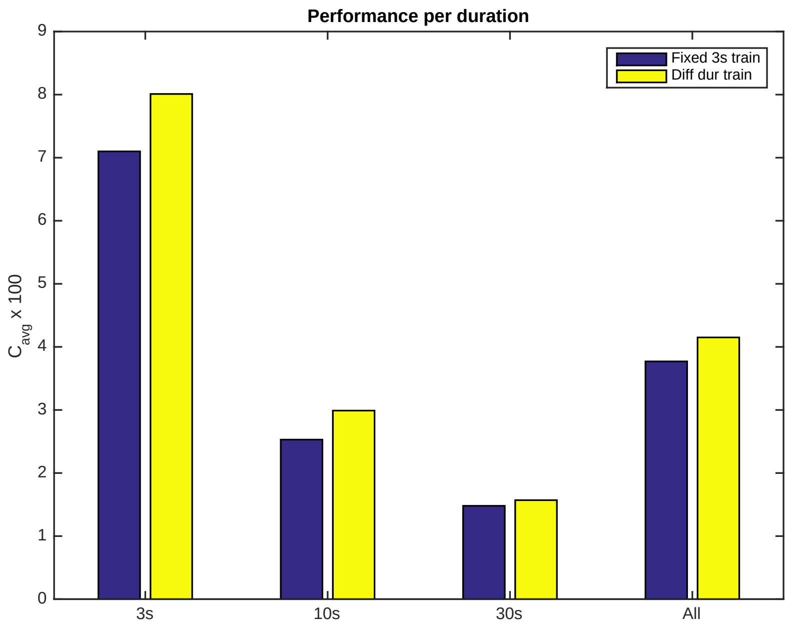


Figure 5.4: *Comparison of performance (in terms of* $\mathrm{C_{avg}} \times 100$*) split by test segment duration between two embedding systems with fixed or variable sequence length in the DNN training.*

System	$C_{primary} \times 100$
i-vector (4096/800)	18.53
Embeddings (best, all augm)	17.96
Fusion (score level)	16.66

Table 5.14: *Results of i-vector system, best embedding system and score level fusion.*

which are likely to leverage different information contained in the initial frame representation of the signal.

This hypothesis is supported by the 10% relative improvement w.r.t. the best reference i-vector system (see Table 5.14), obtained with a simple score level fusion of the best i-vector and embedding system. This result suggests that i-vectors and embeddings are complementary representations of the same utterances, which can be exploited to create a better and more robust LID system.

5.3. Chapter Summary

In this chapter, we explored the recently emerging used of DNNs trained to extract utterance level embeddings, which have become a fair competitor to traditional i-vectors for speaker recognition [Snyder *et al.*, 2017]. Thus, we adapted to the task of language recognition [Lozano-Diez *et al.*, 2018] and explored and analyzed this approach, showing their ability to outperform

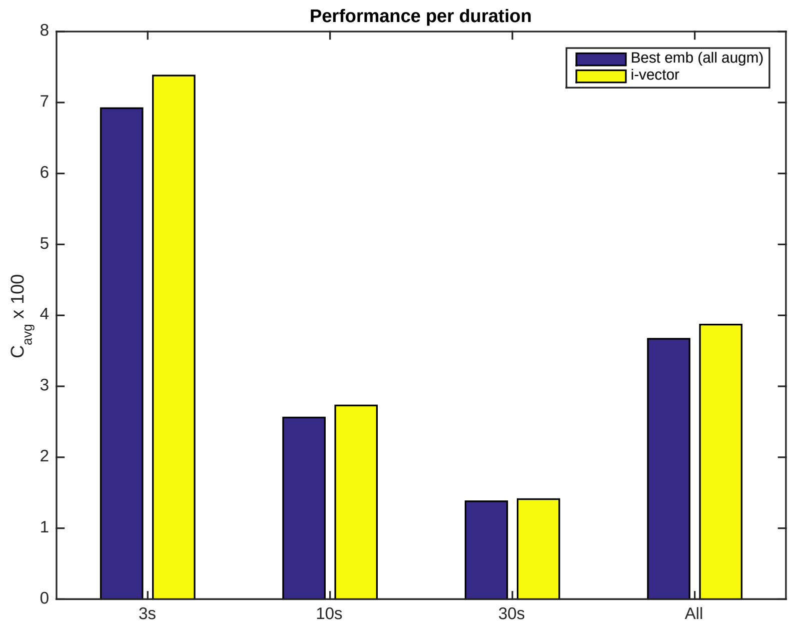


Figure 5.5: *Comparison of performance* ($C_{avg} \times 100$) *per duration between the best embedding system (all augmentations) and the best reference i-vector results.*

strong state-of-the-art i-vector systems.

In the first part of the chapter, we presented a DNN architecture with embeddings for language identification. The DNN is trained to discriminate between languages and at the same time learns an utterance level representation of input segments. These fixed-length representations i.e., embeddings are further used to train a generative classifier (GLC). This is a novel approach for the problem of LID and is in line with the research that has proven to be promising for speaker verification [Snyder *et al.*, 2017]. Our results are comparable with a state-of-the-art i-vector system on the NIST LRE 2015 setup.

In the second part of this chapter, we further explored the embedding DNN structure for LID on the most recent NIST LRE 2017. We analyzed the effect of data augmentation and the influence of different configurations of bottleneck features for DNN training in final performance of the LID system. We have also compared how are the embedding systems performing with segments of different durations and compared the results with i-vectors.

Our results on the NIST LRE 2017 setup suggest that the proposed DNN systems are data-hungry, and that adding noise to the original training data in combination with other perturbations such as reverberation or speed changes significantly improves the performance.

For this second set of experiments, the best configuration of the embedding system outperforms already the strong i-vector system, and, in both LRE 2015 and 2017, a simple score level fusion of embedding and i-vector based approaches is able to further improve the overall LID performance.

We consider DNN embeddings to be a promising line for research in language recognition and related fields, since they provide compact utterance representations and achieve state-of-the-art results. Also, we would like to highlight that using embeddings instead of posteriors from the DNN provided better results, and this fact points towards the direction of training more general DNNs i.e., trained on much larger and variable set of languages, that can provide more general embeddings usable across various LID tasks.

Chapter 6

Conclusions and Future Work

In this Thesis, language and speaker recognition systems based on different deep neural network (DNN) approaches have been explored. A review of traditional systems used to address the language and speaker recognition tasks, including state-of-the-art approaches based on deep neural networks, has been presented. After this introduction, a study of different neural network based approaches has followed, where the experiments conducted are presented. In particular, end-to-end approaches for language identification using convolutional deep neural networks (CDNNs) and long short-term memory (LSTM) recurrent neural networks are analyzed first, followed by the study of DNNs as frame-by-frame bottleneck feature extractors both for language and speaker recognition. Finally, in the last experimental chapter of this Thesis, one of the most recent and successful applications of DNNs for these two tasks has been explored: DNNs as fixed-length utterance-level representation extractors (*embeddings*). Each chapter includes contributions and conclusions drawn from this Thesis, which are summarized in this chapter, pointing out as well some future research lines planned to be followed after this Dissertation.

6.1. Conclusions

This Dissertation starts in Chapter 1 with the introduction of deep neural networks and their impact in speech processing in the last few decades. This chapter also includes the main motivation, aims and contributions of this Thesis.

Chapter 2 briefly describes the tasks of language and speaker recognition and reviews classical approaches to these problems up to state-of-the-art models based on i-vectors. Furthermore, in this chapter, deep neural networks are introduced and their applications to speech, speaker and language recognition are summarized as well, which corresponds to previous (or simultaneous) research works to those conducted in this Thesis.

Then, the three experimental chapters follow. First, Chapter 3 gathers our proposed approaches for language recognition based on convolutional DNNs and LSTMs as end-to-end classifiers, which provides an alternative to i-vector systems and with considerably less parameters. These DNN-based approaches for language recognition seem to be more sensitive to mismatched training and test datasets, but still achieve comparable or better performance than i-vector reference systems, especially for short test segments (even shorter than 3 seconds).

The second experimental chapter is Chapter 4, where an analysis of deep neural networks used as bottleneck feature extractors for the tasks of speaker and language recognition is performed. This chapter provides a frame-by-frame representation of the speech signal widely used in the field in order to replace acoustic features such as MFCCs or PLPs in classical i-vector based systems. Thus, we studied different configurations and designs of the DNNs that influence the final systems performance. These bottleneck features are obtained from a DNN trained for the task of automatic speech recognition (ASR), and therefore, they contain phonetic information which might include more or less useful information related to the final task of language or speaker recognition. Thus, the information compressed by the bottleneck layer will depend on many factors such as its position in the network (closer or further from the output layer that performs phoneme classification) or its size, and further optimization of the system configuration such as previous normalization of input features or the fully optimization of the network for the ASR task.

Besides, bottleneck feature vectors are obtained frame-wise for each speech segment, as other acoustic features (MFCCs, for instance). Then, a fixed-length representation is given by the i-vector, which contains task-dependent information together with channel variability. In Chapter 5, a novel approach is proposed in which a DNN trained for the target task of language identification provides also an utterance-level representation of the speech segment known as *embedding*. These embeddings are then fixed-length vectors that represent utterances in a similar way than i-vectors do, but they are trained for the target task so they are supposed to comprise information useful for it. This proposed approach for language recognition is in line with recently published results in speaker recognition with similar architectures, and provides state-of-the-art results, outperforming strong i-vector systems in challenging tasks, which seems to be a promising research line for further exploration.

To summarize, the main conclusions drawn from this Thesis can be summarized as follows:

- Deep neural networks are a powerful machine learning tool that can noticeably improve performance of traditional speech processing systems. In particular, DNNs have become part of state-of-the-art language and speaker recognition systems, and have led to a new paradigm with several research lines to explore.

- The use of deep neural networks for language and speaker recognition is a hot topic in these research communities, which emerged after their success in speech recognition, one of the biggest research fields whose progress in the last few years have influenced the spread of voice as the way for humans to interact with daily used devices. Advances in speech recognition are usually reflected in those of language and speaker recognition automatic systems, and at the same time, these two systems help speech recognition and related applications perform this task with a specific target language or speaker, which might be needed in some applications, apart from widening the scope of these applications.

- The evolution of speaker and language recognition has experienced a breakthrough since the introduction of deep neural networks into their pipelines. They can be used as a replacement of a whole speaker or language recognition system, or as part of them. From their use as feature extractors to substitute or complement traditional acoustic features, to their application to extract utterance-level vectors to represent the speech signal, they have become a fair competitor to the well-established i-vector approach. This way, step-by-step systems are pointing towards the direction of reducing the speech processing expertise and letting the networks to learn information from data, as it has been one of the main objectives of these machine learning tools, and it is becoming more realistic with the increasing amount of data and computational resources nowadays.

Finally, we would like to highlight the main contributions and results of this Thesis in the following lines:

- The proposed end-to-end approaches for language recognition based on convolutional DNNs and LSTMs, which provided an alternative to i-vectors with less parameters.
- The systematic study of bottleneck feature DNN-based language recognition systems, and the analysis of this approach for speaker recognition, whose optimal DNN configuration might differ from the most beneficial for the task of ASR for which the DNN is trained.
- The novel approach based on embeddings for language recognition, in line with previous works in speaker recognition, which provides a fixed-length representation of utterances directly learned by the DNN for the target task, and has shown to be a fair competitor for well-known i-vectors.

6.2. Future Work

Several research lines can be followed from the work presented in this Thesis, among which we would like to highlight the following:

- One of the first lines we would like to explore further is the approach of DNN-embeddings. For the case of language recognition we consider interesting extending our proposed system with a more general point of view where the network could be trained for a larger set of languages, allowing the systems to extract more general embeddings that could be used across different target sets of languages, including open-set tasks. Moreover, embeddings is an emerging topic for both language and speaker recognition which needs further analysis after their promising role as substitutes for the classical i-vectors, maybe towards the *meta-embedding* concept presented in [Brummer *et al.*, 2018].
- Deep neural networks, among other goals, aim to learn a representation of the input signal, which in this work is already represented by some acoustic features (MFCCs generally), transformed to bottleneck features for some systems afterwards, and this is the starting

point from where the DNNs learn. However, some information contained in the original signal that might be useful for a given task may have been already suppressed in this *pre-processing* stage. Thus, further research from raw signal might be a future line, as this approach has proven successful for related tasks such as voice activity detection [Zazo *et al.*, 2016b].

- The information contained in the speech signal is complex and full of signs from a wide range of variability sources. Generally, DNNs seem to be quite affected by usual existing mismatch between training and test environments, a problem normally present in real speech-based applications. Domain adaptation techniques could be further analyzed as a post-processing of the i-vectors and embeddings obtained in this work.

- Besides, some unwanted information still remains in the utterance-level vectors that represent our speech segments, even when the DNN is trained for the target task as it happens with embeddings. In this case, context might help focusing on useful information while learning a representation for a given task, and attention models [Vaswani *et al.*, 2017] can be considered an useful tool for this purpose.

- A wide range of DNN architectures and configurations are being applied to different research fields nowadays, which might be suitable for adaptation to speech processing related tasks, and conversely, approaches used in this work could be studied in different areas of signal processing, providing at the same time a better understanding of what these machine learning algorithms learn in each hidden layer, easier to observe with some other inputs such images.

Apéndice A

Resumen Extendido de la Tesis

Reconocimiento de Idioma y Locutor basado en Representación Bottleneck y Embedding a partir de Redes Neuronales Profundas

A.1. Resumen

El reconocimiento automático de voz ha experimentado un asombroso progreso en los últimos años, en gran medida gracias a la introducción de las redes neuronales profundas (*deep neural networks*, DNNs) en las aplicaciones de este área. Esta evolución en los sistemas de reconocimiento de voz se ha extendido a otras áreas íntimamente relacionadas como son el reconocimiento de idioma y locutor, donde dichas redes neuronales han conseguido mejorar de forma notable el rendimiento de los sistemas.

En esta Tesis Doctoral, se han explorado diferentes aproximaciones para las tareas de reconocimiento de idioma y de locutor, centrándose en los sistemas donde las redes neuronales profundas reemplazan parte (o todo) del esquema tradicional de dichos sistemas.

En particular, en el primer bloque experimental, se han analizado sistemas de reconocimiento de idioma basados en redes neuronales profundas como sistema completo (*end-to-end*), en los cuales la red neuronal se utiliza directamente como clasificador, sin ningún paso posterior, sino utilizando directamente las salidas de la red, que se corresponden con las probabilidades a posteriori de cada clase (idioma). Así, esta parte del trabajo de investigación se ha centrado en concreto en dos arquitecturas, redes neuronales convolucionales (*convolutional neural networks*) y redes recurrentes LSTM (*long short-term memory recurrent neural networks*), las cuales requieren menos recursos computacionales debido a su número de parámetros libres reducido en comparación con otro tipo de redes neuronales profundas. Por lo tanto, estos sistemas constituyen una alternativa a los tradicionales *i-vectors*, y con ellos se consiguen resultados comparables a los obtenidos con *i-vectors*, especialmente cuando el sistema tiene que lidiar con locuciones cortas. En concreto, hemos realizado experimentos comparando un sistema basado en redes convolucionales con dos sistemas de referencia, uno basado en el clásico modelo de *Factor Analysis* GMM (*Gaussian Mixture Model*) y otro basado en *i-vectors*, y hemos evaluado dichos sistemas en dos tareas diferentes de la evaluación de reconocimiento de idioma LRE (*Language Recognition Evaluation*) organizada por NIST (*National Institute of Standards and Technology*) en 2009: una primera tarea basada en reconocimiento de pares de idiomas y una segunda tarea considerando la clasificación de múltiples idiomas. Los resultados muestran un rendimiento de los sistemas basados en redes convolucionales comparable al obtenido con los sistemas de referencia, y ciertas mejoras se observan con la fusión de ambas aproximaciones. Además, presentamos experimentos llevados a cabo con redes LSTM, las cuales han demostrado su capacidad para modelar secuencias temporales. Así, evaluamos nuestros sistemas de reconocimiento de idioma basados en redes LSTM en diferentes subconjuntos de los entornos de las evaluaciones de reconocimiento de idioma de 2009 y 2015, en los cuales los sistemas basados en redes LSTM son capaces de mejorar el rendimiento del sistema de referencia *i-vector*, proporcionando un modelo con menos parámetros aunque tiende más a sobreajustarse y no es capaz de generalizar de la misma forma que el *i-vector* en entornos muy diferentes de entrenamiento y evaluación.

En el segundo bloque experimental de esta Tesis, se ha explorado uno de los tipos de aplicaciones de redes neuronales profundas más utilizados en el ámbito del procesado de la señal de voz: su uso como extractores de características. En este tipo de sistemas, las redes neuronales son utilizadas para obtener una representación trama a trama de la señal de voz, conocida como *bottleneck feature*. Dicha representación es aprendida directamente por la red y es utilizada más adelante en sustitución a las clásicas características acústicas que se utilizan como entrada en los sistemas de reconocimiento de idioma y locutor basados en *i-vector*. Este tipo de sistemas basados en características *bottleneck* ha revolucionado estos dos ámbitos debido a la notable mejora en rendimiento que presentan en comparación con los sistemas clásicos *i-vector* basados en características acústicas, que han sido desbancados del estado del arte tras muchos años. Nuestro análisis se centra en cómo diferentes configuraciones de la red neuronal usada como extractor de características, que es entrenada para la tarea de reconocimiento de voz (clasificación de fonemas), hacen que el rendimiento de los vectores de características resultantes para las tareas de reconocimiento de idioma y locutor se vea influenciado por dichas variaciones. Para el caso de los sistemas de reconocimiento de idioma, mostramos una comparación de características *bottleneck* extraídas de redes en las que se ha variado su profundidad (número de capas ocultas), la posición de la capa *bottleneck* donde se comprime la información, así como el número de unidades ocultas (tamaño) de esta capa, lo que influencia la representación obtenida por la red. Con los experimentos realizados en sistemas de reconocimiento de locutor basado en *bottleneck*, analizamos la influencia del tipo de características utilizadas para entrenar la red, así como su pre-procesamiento, y, en general, la optimización de la red para la tarea de extracción de características, la cual no tiene por qué ser la configuración óptima para el reconocimiento de voz (tarea para la que se entrena inicialmente).

Para finalizar, en el tercer bloque experimental de esta Tesis Doctoral se propone una aproximación novedosa al problema de reconocimiento de idioma, en la que la función de las redes neuronales es ahora extraer una representación a nivel de locución, de longitud fija, conocida como *embedding*, capaz de reemplazar a los clásicos *i-vectors* y proporcionando una solución al problema del modelado de secuencias de longitud variable que afecta a las características *bottleneck*. Recientemente, este tipo de esquema basado en *embeddings* ha mostrado resultados más que prometedores en el campo del reconocimiento de locutor, y nuestra propuesta es capaz de mejorar el rendimiento de un competitivo sistema *i-vector* de reconocimiento de idioma tomado como referencia, y evaluado en las últimas bases de datos sobre las que han tenido lugar las evaluaciones de reconocimiento de idioma (LREs) organizadas por NIST en 2015 y 2017. De esta manera, analizamos los sistemas de reconocimiento de idioma basados en *embeddings*, explorando diferentes arquitecturas para las redes neuronales profundas y técnicas de aumento de datos para la mejora de los sistemas. En general, estos *embeddings* constituyen un justo competidor a los ampliamente usados *i-vectors* y permiten reemplazar el modelo completo *i-vector* por una red neuronal. Además, la red es capaz de extraer información complementaria a la contenida en los *i-vectors*, incluso cuando el mismo tipo de características para representar la señal de voz son utilizadas. Por todo esto, consideramos esta contribución una línea interesante para investigar en otros campos relacionados.

A.2. Conclusiones

Esta Tesis comienza en el Capítulo 1 con la introducción de las redes neuronales profundas (DNNs, por sus siglas en inglés) así como su impacto en el campo del procesamiento de la señal de voz en las últimas décadas. Este capítulo incluye además los objetivos y la motivación de esta Tesis Doctoral, así como sus contribuciones.

En el Capítulo 2 se describen brevemente las tareas de reconocimiento de idioma y locutor y se repasan las aproximaciones clásicas a las mismas hasta llegar a los sistemas en el estado del arte basados en *i-vector*. Además, en este mismo capítulo, se introducen las redes neuronales profundas y se resumen sus aplicaciones en reconocimiento de voz, idioma y locutor, las cuales constituyen los trabajos de investigación previos (o simultáneos) a los llevados a cabo en esta Tesis Doctoral.

Seguidamente, los tres bloques experimentales son presentados en los capítulos siguientes. Primero, el Capítulo 3 recoge los sistemas propuestos para reconocimiento de idioma basados en redes convolucionales y LSTMs como clasificadores, que proporcionan una alternativa para los sistemas basados en *i-vector* pero con un número de parámetros mucho más reducido. Este tipo de esquemas para reconocimiento de idioma basados en DNNs parecen ser más sensibles al desajuste entre los datos de entrenamiento y evaluación, pero aún así consiguen rendimientos comparables a los obtenidos con los sistemas *i-vector* de referencia, especialmente cuando se trata de segmentos de test cortos (incluso menos de 3 segundos).

El segundo bloque experimental se condensa en el Capítulo 4, en el que se realiza un análisis sobre el uso de redes neuronales profundas como extractores de características *bottleneck* para las tareas de reconocimiento de idioma y de locutor. Este tipo de sistemas proporcionan una representación trama a trama de la señal de voz, ampliamente utilizadas en el ámbido de investigación para sustituir a las características acústicas como MFCCs o PLPs de los sistemas *i-vector* tradicionales. Así, se presenta un estudio sistemático de esta aproximación, analizando diferentes diseños y configuraciones de las DNNs, que influencian el rendimiento del sistema final. Estas características *bottleneck* se obtienen a través de una red neuronal entrenada para la tarea de reconocimiento de voz (*automatic speech recognition*, ASR), y como tal, contienen información fonética que puede incluir información más o menos útil para las tareas finales de reconocimiento del idioma o del locutor. Por lo tanto, la información comprimida por la capa *bottleneck* de la DNN dependerá de diversos factores como puede ser su posición dentro de la red (más o menos cercana a la capa de salida en la que se realiza la clasificación de los fonemas) o su tamaño, así como la optimización de otros parámetros de configuraión del sistema como la normalización previa de las características de entrada a la red o si ésta es optimizada completamente durante el entrenamiento para la tarea de reconocimiento de fonemas (ASR).

Por otra parte, las características *bottleneck* utilizadas se obtienen para cada trama de un segmento de voz, al igual que se hace con las características acústicas (como los MFCCs). Por lo tanto, una representación de tamaño fijo para las locuciones (de duraciones diferentes) se consigue con el conocido *i-vector*, que contiene tanto información de la tarea como de la variabilidad de canal, todo en el mismo vector. En el Capítulo 5 se propone una novedosa aproximación en la que una red neuronal es entrenada para la tarea que se desea, en nuestro caso para clasificación de idiomas, y es esta misma red la que proporciona una representación a nivel de locución para el segmento de voz conocida como *embedding*. Estos *embeddings* son, por tanto, vectores de dimensión fija que representan locuciones de forma similar a como lo hacen los *i-vectors*, pero en este caso provienen de una DNN entrenada para la tarea final y por ello, deberían contener información útil para la misma. Este sistema propuesto para reconocimiento de idioma es consecuente con resultados publicados recientemente con arquitecturas similares para reconocimiento de locutor y, además, consigue resultados en el estado del arte, mejorando sistemas *i-vector* muy competitivos sobre tareas desafiantes. Por todo esto, se considera una línea muy prometedora para su continuación como línea de investigación futura.

En resumen, las principales conclusiones que se extraen de esta Tesis Doctoral se pueden resumir en las siguientes:

- Las redes neuronales profundas constituyen una herramienta muy poderosa para el aprendizaje automático que es capaz de mejorar de forma notable el rendimiento de los sistemas tradicionales para el procesado de la señal de voz. En particular, estas DNNs se han convertido en parte del estado del arte en los sistemas de reconocimiento de idioma y locutor, y han dado lugar a un nuevo paradigma en numerosas líneas de investigación a explorar.

- El uso de redes neuronales profundas para reconocimiento de idioma y de locutor es un tema candente en estas comunidades investigadoras, y ha resurgido con fuerza tras su éxito en reconocimiento de voz, uno de los ámbitos de investigación más grandes cuyo progreso en los últimos años ha influido en la expansión de la voz como el método utilizado por los humanos para interactuar con los dispositivos utilizados en nuestro día a día. Los progresos en reconocimiento de voz se han reflejado normalmente en los conseguidos en los sistemas de reconocimiento de idioma y de locutor, y, al mismo tiempo, estos dos ámbitos han ayudado al propio reconocimiento de voz y sus aplicaciones haciendo posible su uso para tareas específicas en las que la información del idioma o del locutor es necesaria o puede ampliar su ámbito de aplicación.

- La evolución del reconocimiento automático de idioma y locutor ha experimentado un antes y un después con la introducción de las redes neuronales profundas en los mismos. Estas DNNs pueden ser utilizadas como sustitutas del sistema de reconocimiento de idioma o de locutor, o de una parte de él. Desde su uso como extractores de características para sustituir o complementar a las características acústicas tradicionales, hasta su aplicación para la extracción de vectores a nivel de locución para representar la señal de voz, se han convertido en un justo competidor para los ampliamente aceptados modelos basados en *i-vector*. De esta forma, paso a paso los sistemas se van moviendo en la dirección de reducir la información del experto en procesamiento de la señal de voz y permitir a las redes neuronales aprender directamente de los datos, que ha sido uno de los objetivos principales de estas herramientas de aprendizaje automático desde el comienzo, y que se está convirtiendo en algo realista con el incremento del tamaño de las bases de datos y los recursos de computación en la actualidad.

Finalmente, cabe destacar las principales contribuciones y resultados de esta Tesis Doctoral, que se resumen a continuación:

- Los sistemas *end-to-end* propuestos en los que las redes neuronales son utilizadas como clasificadores para la tarea final de reconocimiento de idioma basados en redes convolucionales y LSTM, que proporcionan una alternativa a los *i-vectors* con menos parámetros.

- El estudio sistemático de los sistemas de reconocimiento de idioma basados en características *bottleneck*, así como el análisis de este mismo esquema para reconocimiento de locutor, cuya configuración óptima de la red neuronal para la tarea final puede variar con respecto al diseño óptimo más beneficioso para la tarea de reconocimiento de voz (ASR) para la que se entrena inicialmente.

- La novedosa propuesta basada en representación *embedding* para reconocimiento de idioma, consecuente con los trabajos previos recientes en reconocimiento de locutor, la cual proporciona una representación de dimensión fija de las locuciones aprendida directamente por la DNN, que es entrenada para la tarea objetivo, y que ha demostrado ser un claro competidor para los ampliamente aceptados *i-vectors*.

A.3. Líneas de Trabajo Futuro

Numerosas líneas de investigación podrían ser exploradas a partir del trabajo presentado en esta Tesis Doctoral, entre las que nos gustaría destacar las siguientes:

- Una de las primeras líneas de investigación que nos gustaría explorar es los sistemas basados en *embeddings*. Para el caso de reconocimiento de idioma, consideramos interesante la ampliación de nuestra propuesta desde un punto de vista más general donde la red neuronal podría ser entrenada para clasificar un conjunto más amplio de idiomas, permitiendo así al sistema extraer una representación *embedding* más genérica que pudiera ser utilizada para diferentes tareas, incluyendo idiomas no vistos durante el entrenamiendo del sistema. Además, los *embeddings* han emergido con fuerza en ambos ámbitos, reconocimiento de idioma y locutor, por lo que necesita un análisis más profundo tras sus resultados tan prometedores como sustitutos para los clásicos *i-vectors*, quizá acercándose al concepto de *meta-embedding* presentado en [Brummer *et al.*, 2018].

- Las redes neuronales profundas tienen entre sus objetivos el ser capaces de aprender una representación de la señal de entrada, que en este trabajo se representa mediante características acústicas (normalmente, MFCCs), o bien transformadas a características *bottleneck*, y que constituyen el punto de partida del que aprenden las DNNs. Sin embargo, alguna información contenida en la señal original que podría ser útil para la tarea objetivo puede estar siendo suprimida en estas fases de parametrización (o pre-procesado). Por lo tanto, más investigación a partir de la señal original se considera una línea de investigación futura, ya que dicha aproximación ha demostrado su éxito en tareas relacionadas como la detección de actividad de voz [Zazo *et al.*, 2016b].

- La información contenida en la señal de voz es muy compleja y está repleta de muestras del amplio rango de fuentes de variabilidad que contiene. En general, las DNNs parecen verse afectadas por el desajuste observado normalmente entre los entornos de los datos de entrenamiento y los de evaluación, un problema presente en las aplicaciones reales basadas en voz. Las técnicas de adaptación al dominio (*domain adaptation*) podrían ser analizadas como una etapa de post-procesado para los *i-vector* y *embeddings* obtenidos en este trabajo.

- Además, en las representaciones de las locuciones obtenidas por nuestros sistemas, todavía persiste información no deseada, incluso cuando las redes neuronales son entrenadas para la tarea objetivo, como es el caso de los *embeddings*. Para este caso, el contexto puede ayudar a centrar la atención de la red en la información útil a la vez que aprende la representación requerida, por los que los modelos de atención (*attention models*) podrían considerarse una herramienta a explorar para este propósito.

- Un amplio rango de arquitecturas y configuraciones de las redes neuronales profundas se aplican a distintos ámbitos de investigación en la actualidad, y éstas podrían ser buenas candidatas para su adaptación a los sistemas relacionados con el procesamiento de voz. Y, vice-versa, las aproximaciones utilizadas en este trabajo podrían ser estudiadas en áreas diferentes del procesamiento de señal, proporcionando a su vez un mejor entendimiento de la forma de aprendizaje que se produce en las capas ocultas de este tipo de algoritmos de aprendizaje automático, que podría resultar más fácil de visualizar en otros contextos que cuentan con imágenes como entrada a los sistemas.

References

Aachen impulse response database. http://www.iks.rwth-aachen.de/en/research/tools-downloads/databases/aachen-impulse-response-database/. 93

C4dm (center for digital music) RIR database. http://isophonics.net/content/room-impulse-response-data-set. 93

Multichannel acoustic reverberation database at york. http://www.commsp.ee.ic.ac.uk/ sap/resources/mardy-multichannel-acoustic-reverberation-database-at-york-database/. 93

Openair impulse response database. http://www.openairlib.net/auralizationdb. 93

Reverb challenge. reverb2014.dereverberation.com. 93

Rwcp sound scene database. http://www.openslr.org/13/. 93

O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 22(10):1533–1545, Oct. 2014. ISSN 2329-9290. URL `http://dx.doi.org/10.1109/TASLP.2014.2339736`. 17, 23

M. Alam, P. Ouellet, P. Kenny, and D. O'Shaughnessy. Comparative evaluation of feature normalization techniques for speaker verification. In *Advances in Nonlinear Speech Processing*, volume 7015 of *Lecture Notes in Computer Science*, pages 246–253. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25019-4. URL `http://dx.doi.org/10.1007/978-3-642-25020-0_32`. 68

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

Y. Bao, H. Jiang, L. Dai, and C. Liu. Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6980–6984, May 2013. 3, 18

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009. 24

Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE, 2013. 37, 38

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation. 26, 31

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738. 17

H. A. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, Norwell, MA, USA, 1993. ISBN 0792393961. 17

N. Brümmer. Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scores - tutorial and user manual -. *Software available at http://sites. google. com/site/nikobrummer/focalmulticlass*, 2007. 33, 39

N. Brummer, A. Silnova, L. Burget, and T. Stafylakis. Gaussian meta-embeddings for efficient scoring of a heavy-tailed plda model. *ArXiv e-prints*, Feb. 2018. 105, 112

S. Cumani and P. Laface. Joint estimation of plda and nonlinear transformations of speaker vectors. *IEEEACM Transactions on Audio, Speech, and Language Processing*, 25(10):1890–1900, Oct 2017. ISSN 2329-9290. 91

S. Cumani, O. Plchot, and R. Fér. Exploiting i-vector posterior covariances for short-duration language recognition. In *Proceedings of Interspeech 2015*. International Speech Communication Association, 2015. ISBN 978-1-5108-1790-6. URL `http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10967`. 14, 83

N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech & Language Processing*, 19(4):788–798, 2011a. URL `http://dx.doi.org/10.1109/TASL.2010.2064307`. 13, 14, 69

N. Dehak, P. Kenny, R. Dehak, O. Glembek, P. Dumouchel, L. Burget, V. Hubeika, and F. Castaldo. Support vector machines and joint factor analysis for speaker verification. In *ICASSP*, pages 4237–4240, 2009. 26

N. Dehak, P. A. Torres-Carrasquillo, D. A. Reynolds, and R. Dehak. Language Recognition via i-vectors and Dimensionality Reduction. In *INTERSPEECH*, pages 857–860. ISCA, 2011b. 32, 39

L. Deng and X. Li. Machine learning paradigms for speech recognition: An overview. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1060–1089, May 2013. ISSN 1558-7916. 18

L. Deng, M. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech 2010*. International Speech Communication Association, September 2010. URL `https://www.microsoft.com/en-us/research/publication/binary-coding-of-speech-spectrograms-using-a-deep-auto-encoder/`. 3, 18

R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černocký. Multilingual bottleneck features for language recognition. In *Proceedings of Interspeech 2015*, volume 2015, pages 389–393, 2015. ISBN 978-1-5108-1790-6. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=10966`. 18, 78

R. Fer, P. Matejka, F. Grezl, O. Plchot, K. Vesely, and J. H. Cernocky. Multilingually trained bottleneck features in spoken language recognition. *Computer Speech & Language*, 46(Supplement C):252 – 267, 2017. ISSN 0885-2308. 81

L. Ferrer, H. Bratt, L. Burget, J. Černocký, O. Glembek, M. Graciarena, A. Lawson, Y. Lei, P. Matějka, O. Plchot, and N. Scheffer. Promoting robustness for speaker modeling in the community: the prism evaluation set. In *Proceedings of SRE11 Analysis Workshop in 2011*, pages 1–7, 2011. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=10859`. 68

L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer. Spoken language recognition based on senone posteriors. In *Proceedings of Interspeech 2014*, pages 2150–2154, 2014. URL `http://www.isca-speech.org/archive/interspeech_2014/i14_2150.html`. 18, 19

L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer. Study of senone-based deep neural network approaches for spoken language recognition. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(1):105–116, 2016. URL `http://dx.doi.org/10.1109/TASLP.2015.2496226`. 18, 70

V. Fontaine, C. Ris, J.-M. Boite, and M. S. Initialis. Nonlinear discriminant analysis for improved speech recognition. In *Proc. Eurospeech-97, Rhodes*, pages 4–2071, 1997. 18

V. Frinken, F. Zamora-Martinez, S. Espana-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke. Long-short term memory neural networks language modeling for handwriting recognition. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 701–704. IEEE, 2012. 37

D. Garcia-Romero and C. Y. Espy-Wilson. Analysis of i-vector length normalization in speaker recognition systems. In *Proceedings of Interspeech 2011*, pages 249–252. International Speech Communication Association, 2011. URL `http://www.isca-speech.org/archive/interspeech_2011/i11_0249.html`. 69

D. Garcia-Romero and A. McCree. Insights into deep neural networks for speaker recognition. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 1141–1145, 2015. URL `http://www.isca-speech.org/archive/interspeech_2015/i15_1141.html`. 20, 55, 78

D. Garcia-Romero, X. Zhang, A. McCree, and D. Povey. Improving speaker recognition performance in the domain adaptation challenge using deep neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 378–383, Dec 2014. 20

G. Gelly and J. Gauvain. Spoken language identification using lstm-based angular proximity. In *Proceedings of Interspeech 2017*, 2017. 19, 80

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000. 37

F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, Mar. 2003. ISSN 1532-4435. 37, 38

J. Godfrey and E. Holliman. Switchboard-1 release 2 ldc97s62. Web Download, https://catalog.ldc.upenn.edu/ldc97s62, 1993. 56

J. Gonzalez-Dominguez, D. Eustis, I. Lopez-Moreno, A. Senior, F. Beaufays, and P. J. Moreno. A real-time end-to-end multilingual speech recognition architecture. *IEEE Journal of Selected Topics In Signal Processing*, 9(4):749–759, June 2015. 18

J. Gonzalez-Dominguez, I. Lopez-Moreno, J. Franco-Pedroso, D. Ramos, D. T. Toledano, and J. Gonzalez-Rodriguez. Atvs-uam nist sre 2010 system. In *Proceedings of FALA 2010*, November 2010a. 26

J. Gonzalez-Dominguez, I. Lopez-Moreno, J. Franco-Pedroso, D. Ramos, D. T. Toledano, and J. Gonzalez-Rodriguez. Multilevel and session variability compensated language recognition: Atvs-uam systems at nist lre 2009. *IEEE Journal on Selected Topics in Signal Processing*, 2010b. article in press. 26

J. Gonzalez-Dominguez, I. Lopez-Moreno, H. Sak, J. Gonzalez-Rodriguez, and P. J. Moreno. Automatic language identification using long short-term memory recurrent neural networks. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2155–2159, 2014. 6, 18, 22, 33, 37, 40, 42, 81

A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385. Springer, 2012. ISBN 978-3-642-24796-5. URL `http://dx.doi.org/10.1007/978-3-642-24797-2`. 37

A. Graves, N. Jaitly, and A.-R. Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013. 37

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. 38

K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015. xviii, 37

F. Grézl, M. Karafiát, and L. Burget. Investigation into bottle-neck features for meeting speech recognition. In *Proc. Interspeech 2009*, number 9, pages 2947–2950, 2009a. ISBN 978-1-61567-692-7. 18

F. Grézl, M. Karafiát, and L. Burget. Investigation into bottle-neck features for meeting speech recognition. In *Proc. Interspeech 2009*, number 9, pages 2947–2950. International Speech Communication Association, 2009b. ISBN 978-1-61567-692-7. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=9038`. 68

F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky. Probabilistic and bottle-neck features for lvcsr of meetings. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–757–IV–760, April 2007. 3, 18, 54

A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. URL `http://arxiv.org/abs/1412.5567`. 3, 17

G. Heigold, I. Moreno, S. Bengio, and N. Shazeer. End-to-end text-dependent speaker verification. In *Proceedings of ICASSP*, March 2016. 20, 79

G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012a. 3, 17

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, Nov 2012b. ISSN 1053-5888. 17

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 2

B. Jiang, Y. Song, S. Wei, J.-H. Liu, I. V. McLoughlin, and L.-R. Dai. Deep bottleneck features for spoken language identification. *PLOS ONE*, 9(7):1–11, 07 2014. URL `https://doi.org/10.1371/journal.pone.0100795`. 18, 78

M. Karafiát, F. Grézl, K. Veselý, M. Hannemann, I. Szőke, and J. Černocký. But 2014 babel system: Analysis of adaptation in nn based systems. In *Proceedings of Interspeech 2014*, pages 3002–3006. International Speech Communication Association, 2014. ISBN 978-1-63439-435-2. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=10745`. 59, 68, 70

P. Kenny. Bayesian speaker verification with heavy-tailed priors. In *Odyssey 2010: The Speaker and Language Recognition Workshop, Brno, Czech Republic, June 28 - July 1, 2010*, page 14, 2010. URL `http://www.isca-speech.org/archive_open/odyssey_2010/od10_014.html`. 69

P. Kenny, G. Boulianne, and P. Dumouchel. Eigenvoice modeling with sparse training data. *Speech and Audio Processing, IEEE Transactions on*, 13(3):345–354, 2005. ISSN 1063-6676. 26

P. J. Kenny, V. Gupta, T. Stafylakis, P. Ouellet, and J. Alam. Deep neural networks for extracting baum-welch statistics for speaker recognition. In *Proc. ODYSSEY*, 2014. 18

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, Dec. 2014. 82

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015. ISSN 0028-0836. 63

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001. 17, 23, 24

Y. Lei, L. Ferrer, A. Lawson, M. McLaren, and N. Scheffer. Application of convolutional neural networks to language identification in noisy conditions. In *Proc. ODYSSEY*, 2014a. 18, 23

Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1695–1699, May 2014b. 18, 19, 20

R. Li, H. S. Mallidi, O. Plchot, L. Burget, and N. Dehak. Exploiting hidden-layer responses of deep neural networks for language recognition. In *Proceedings of Interspeech 2016*, 2016. ISBN 978-1-5108-3313-5. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=11272`. 19, 80

LISA. *Deep Learning Tutorial.* University of Montreal, http://deeplearning.net/tutorial/. 26, 31

I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno. Automatic Language Identification using Deep Neural Networks. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 2014. 18, 21, 22, 36

A. Lozano-Diez, O. Plchot, P. Matějka, and J. Gonzalez-Rodriguez. Dnn based embeddings for language recognition. In *Proceedings of ICASSP*, April 2018. 99

A. Lozano-Diez, A. Silnova, P. Matějka, O. Glembek, O. Plchot, J. Pešán, L. Burget, and J. Gonzalez-Rodriguez. Analysis and optimization of bottleneck features for speaker recognition. In *Proceedings of Odyssey 2016*. International Speech Communication Association, 2016. URL `http://www.fit.vutbr.cz/research/view_pub.php.cz.iso-8859-2?id=11219`. 20, 78

A. Lozano-Diez, R. Zazo, D. T. Toledano, and J. Gonzalez-Rodriguez. An analysis of the influence of deep neural network (dnn) topology in bottleneck feature based language recognition. *PLOS ONE*, 12(8):1–22, 08 2017. URL `https://doi.org/10.1371/journal.pone.0182580`. 19, 78

D. G. Martínez, O. Plchot, L. Burget, O. Glembek, and P. Matějka. Language recognition in ivectors space. In *Proceedings of Interspeech 2011*, pages 861–864, 2011. ISBN 978-1-61839-270-1. 13, 14, 15, 83

P. Matějka, L. Zhang, T. Ng, H. S. Mallidi, O. Glembek, J. Ma, and B. Zhang. Neural network bottleneck features for language identification. In *Proceedings of Odyssey 2014*. International Speech Communication Association, 2014a. 19, 78

P. Matějka, L. Zhang, T. Ng, H. S. Mallidi, O. Glembek, J. Ma, and B. Zhang. Neural network bottleneck features for language identification. In *Proceedings of Odyssey 2014*, volume 2014, pages 299–304. International Speech Communication Association, 2014b. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=10686`. 54, 55

P. Matějka, O. Glembek, O. Novotný, O. Plchot, F. Grézl, L. Burget, and J. H. Černocký. Analysis of dnn approaches to speaker identification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5100–5104, March 2016. 70, 74

M. McLaren, Y. Lei, and L. Ferrer. Advances in deep neural network approaches to speaker recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4814–4818, April 2015. 20

M. McLaren, Y. Lei, N. Scheffer, and L. Ferrer. Application of convolutional neural networks to speaker recognition in noisy conditions. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, 2014. 23

T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011. 22, 37

A. Mohamed, G. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, Jan 2012. ISSN 1558-7916. 2

G. Montavon. Deep learning for spoken language identification. In *NIPS workshop on Deep Learning for Speech Recognition and Related Applications*, 2009. 18

Y. K. Muthusamy, E. Barnard, and R. A. Cole. Reviewing automatic language identification. *IEEE Signal Processing Magazine*, 11(4):33–41, 1994. 11

NIST. NIST 2017 Language Recognition Evaluation Plan. https://www.nist.gov/sites/default/files/documents/2017/06/01/lre17_eval_plan-2017-05-31_v2.pdf, a. 89, 90, 95

NIST. The 2015 NIST Language Recognition Evaluation Plan (LRE15). http://www.nist.gov/itl/iad/mig/upload/ LRE15_EvalPlan_v23.pdf, b. 47, 48, 56, 58, 84, 86, 95

NIST. The nist year 2008 speaker recognition evaluation plan. www.itl.nist.gov/iad/mig/tests/sre/2008/sre08_evalplan_release4.pdf, 2008. 69

NIST. The 2009 nist language recognition evaluation plan, 2009. URL `http://www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf`. 27, 33

NIST. The nist year 2010 speaker recognition evaluation plan. www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r6.pdf, 2010. 69

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013. 37

J. Pešán, L. Burget, and J. Černocký. Sequence summarizing neural networks for spoken language recognition. In *Proceedings of Interspeech 2016*, pages 3285–3289, 2016. ISBN 978-1-5108-3313-5. URL `http://www.fit.vutbr.cz/research/view_pub.php?id=11273`. 19, 80

O. Plchot, P. Matějka, R. Fér, O. Glembek, O. Novotný, J. Pešán, K. Veselý, L. Ondel, M. Karafiát, F. Grézl, S. Kesiraju, L. Burget, N. Brummer, P. du Albert Swart, S. Cumani, H. S. Mallidi, and R. Li. Bat system description for nist lre 2015. In *Proceedings of Odyssey 2016*. International Speech Communication Association, 2016. URL `http://www.fit.vutbr.cz/research/view_pub.php.en.iso-8859-2?id=11221`. 84, 95

O. Plchot, P. Matějka, O. Novotný, S. Cumani, A. Lozano-Diez, J. Slavicek, M. Diez, F. Grézl, O. Glembek, K. V. Mounika, A. Silnova, L. Burget, L. Ondel, S. Kesiraju, and J. Rohdin. Analysis of but-pt submission for nist lre 2017. In *in Proc. of Odyssey 2018 (to appear)*, 2018. 91, 92, 98

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011a. ISBN 978-1-4673-0366-8. IEEE Catalog No.: CFP11SRW-USB. 32, 39, 40

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011b. IEEE Catalog No.: CFP11SRW-USB. 56, 59

S. J. D. Prince. Probabilistic linear discriminant analysis for inferences about identity. In *Proc. International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007. 20, 80

S. J. D. Prince and J. H. Elder. Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007. 15

F. Richardson, D. Reynolds, and N. Dehak. Deep neural network approaches to speaker and language recognition. *IEEE Signal Processing Letters*, 22(10):1671–1675, Oct 2015a. ISSN 1070-9908. 19, 78

F. Richardson, D. Reynolds, and N. Dehak. A unified deep neural network for speaker and language recognition. In *Proceedings of Interspeech 2015*, pages 1146–1150. International Speech Communication Association, 2015b. 73

J. Rohdin, A. Silnova, M. Diez, O. Plchot, P. Matejka, and L. Burget. End-to-end DNN Based Speaker Recognition Inspired by i-vector and PLDA. *ArXiv e-prints*, Oct. 2017. URL `https://arxiv.org/abs/1710.02369`. 20

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 2

D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur. Deep neural network embeddings for text-independent speaker verification. In *Proceedings of Interspeech 2017*, 2017. 7, 19, 20, 80, 81, 86, 87, 88, 99, 100

D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur. Deep neural network-based speaker embeddings for end-to-end speaker verification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 165–170, Dec 2016. 20

Y. Song, B. Jiang, Y. Bao, S. Wei, and L. R. Dai. I-vector representation based on bottleneck features for language identification. *Electronics Letters*, 49(24):1569–1570, November 2013. ISSN 0013-5194. 19, 78

R. Stewart and M. Sandler. Database of omnidirectional and b-format room impulse responses. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 165–168, March 2010. 93

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014. 2

D. Talkin. A robust algorithm for pitch tracking (RAPT). In W. B. Kleijn and K. Paliwal, editors, *Speech Coding and Synthesis*, New York, 1995. Elseviever. 81

P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, and J. R. Deller. Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features. In *Proc. ICSLP*, volume 1, pages 89–92, 2002. 31, 58

D. A. Van Leeuwen and N. Brummer. Channel-dependent gmm and multi-class logistic regression models for language recognition. In *Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The*, pages 1–8. IEEE, 2006. 27

E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez. Deep neural networks for small footprint text-dependent speaker verification. In *Proceedings of ICASSP*, May 2014. 20, 79

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *ArXiv e-prints*, June 2017. 106

R. Vogt and S. Sridharan. Explicit modelling of session variability for speaker verification. *Computer Speech & Language*, 22(1):17–38, 2008. 26

F. Weninger, J. Bergmann, and B. Schuller. Introducing currennt: the munich open-source cuda recurrent neural network toolkit. *Journal of Machine Learning Research*, 15, 2014. 39

S. Yaman, J. Pelecanos, and R. Sarikaya. Bottleneck features for speaker recognition. In *Proceedings of Odyssey 2012*. International Speech Communication Association, 2012. 20, 78

R. Zazo, A. Lozano-Diez, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez. Language identification in short utterances using long short-term memory (lstm) recurrent neural networks. *PLOS ONE*, 11 (1):1–17, 01 2016a. 81

R. Zazo, T. N. Sainath, G. Simko, and C. Parada. Feature learning with raw-waveform cldnns for voice activity detection. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 3668–3672, 2016b. URL `http://dx.doi.org/10.21437/Interspeech.2016-268`. 106, 112

M. A. Zissman and E. Singer. Automatic language identification of telephone speech messages using phoneme recognition and n-gram modeling. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, volume i, pages I/305–I/308 vol.1, Apr 1994. 55

Colophon

This book was typeset by the author using LaTeX2e. The main body of the text was set using a 11-points Computer Modern Roman font. All graphics and images were included formatted as Encapsuled Postscript ($^{\mathrm{TM}}$ Adobe Systems Incorporated). The final postscript output was converted to Portable Document Format (PDF) and printed.

This Thesis was printed with the financial support from EPS-UAM and the AUDIAS research group.