TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2022

90

Victoria Mingote Bueno

Representation and Metric Learning Advances for Deep Neural Network Face and Speaker Biometric Systems

Director/es
Miguel Artiaga, Antonio

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

REPRESENTATION AND METRIC LEARNING ADVANCES FOR DEEP NEURAL NETWORK FACE AND SPEAKER BIOMETRIC SYSTEMS

Autor

Victoria Mingote Bueno

Director/es

Miguel Artiaga, Antonio

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Tecnologías de la Información y Comunicaciones en Redes Móviles

2022

Universidad de Zaragoza

Programa de Doctorado en Tecnologías de la Información y Comunicaciones en Redes Móviles

Ph.D. Thesis

Representation and Metric Learning Advances for Deep Neural Network Face and Speaker Biometric Systems

Victoria Mingote Bueno

Thesis Advisor

Dr. Antonio Miguel Artiaga

Departamento de ingeniería electrónica y comunicaciones

Escuela de Ingeniería y Arquitectura

March 2022

Universidad Zaragoza

A mi familia y amigos.

A todos los que han estado ahí.

Abstract

The increasing use of technological devices and biometric recognition systems in people daily lives has motivated a great deal of research interest in the development of effective and robust systems. However, there are still some challenges to be solved in these systems when Deep Neural Networks (DNNs) are employed. For this reason, this thesis proposes different approaches to address these issues.

First of all, we have analyzed the effect of introducing the most widespread DNN architectures to develop systems for face and text-dependent speaker verification tasks. In this analysis, we observed that state-of-the-art DNNs established for many tasks, including face verification, did not perform efficiently for text-dependent speaker verification. Therefore, we have conducted a study to find the cause of this poor performance and we have noted that under certain circumstances this problem is due to the use of a global average layer as pooling mechanism in DNN architectures. Since the order of the phonetic information is relevant in text-dependent speaker verification task, whether a global average pooling is employed, this order is neglected and the results achieved for the verification performance metrics are too high. Hence, the first approach proposed in this thesis is an alignment mechanism which is used to replace the global average pooling. This alignment mechanism allows to keep the temporal structure and to encode the utterance and speaker identity in a supervector. As alignment mechanism, different types of approaches such as Hidden Markov Model (HMM) or Gaussian Mixture Model (GMM) can be used. Moreover, during the development of this mechanism, we also noted that the lack of larger training databases is another important issue to create these systems. Therefore, we have also introduced a new architecture philosophy based on the Knowledge Distillation (KD) approach. This architecture is known as teacher-student architecture and provides robustness to the systems during the training process and against possible overfitting due to the lack of data. In this part, another alternative approach is proposed to focus on the relevant frames of the sequence and maintain the phonetic information, which consists of Multi-head Self-Attention (MSA). The architecture proposed to use the MSA layers also introduces phonetic embeddings and memory layers to improve the discrimination between speakers and utterances. Moreover, to complete the architecture with the previous techniques, another approach has been incorporated where two learnable vectors have been introduced which are called class and distillation tokens. Using these tokens during training, temporal information is kept and encoded into the tokens, so that at the end, a global utterance descriptor similar to the supervector is obtained.

Apart from the above approaches to obtain robust representations, the other main part of this thesis has focused on introducing new loss functions to train DNN architectures. Traditional loss functions have provided reasonably good results for many tasks, but there are not usually designed to optimize the goal task. For this reason, we have proposed several new loss functions as objective for training DNN architectures which are based on the final verification metrics. The first approach developed for this part is inspired by the Area Under the ROC Curve (AUC). Thus, we have presented a differentiable approximation of this metric called *aAUC* loss to successfully train a triplet neural network as back-end. However, the selection of the training data has to be carefully done to carry out this back-end, so it involves a high computational cost. Therefore, we have developed several approaches to take advantage of training with a loss function oriented to the goal task but keeping the efficiency and speed of multi-class training. To implement these approaches, the differentiable approximation of the Detection Cost Function (*aDCF*) and Cost of Log-Likelihood Ratio (CLLR) verification metrics have been employed as training objective loss. By optimizing DNN architectures to minimize these loss functions, the system learns to reduce errors in decisions and scores produced. The use of these approaches has also shown a better ability to learn more general representations than training with other traditional loss functions. Finally, we have also proposed a new straightforward back-end that employs the information learned by the matrix of the last layer of DNN architecture during training with *aDCF* loss. Using the matrix of this last layer, an enrollment model with a learnable vector is trained for each enrollment identity to perform the verification process.

Resumen

El aumento del uso de dispositivos tecnológicos y sistemas de reconocimiento biométrico en la vida cotidiana de las personas ha motivado un gran interés en la investigación y el desarrollo de sistemas eficaces y robustos. Sin embargo, todavía existen algunos retos que resolver en estos sistemas cuando se emplean redes neuronales profundas. Por esta razón, esta tesis propone diferentes enfoques para abordar estas cuestiones.

En primer lugar, hemos analizado el efecto de introducir las arquitecturas de redes neuronales profundas más extendidas para desarrollar sistemas para tareas de verificación de caras y locutores dependientes del texto. En este análisis, hemos observado que las redes neuronales profundas del estado del arte establecidas para muchas tareas, incluyendo la verificación de caras, no funcionan de forma eficiente para la verificación de locutores dependientes del texto. Por lo tanto, hemos realizado un estudio para encontrar la causa de este pobre rendimiento y hemos notado que este problema se debe al uso de la capa de promediado global como mecanismo de agrupación en las arquitecturas de redes neuronales profundas. Dado que el orden de la información fonética es relevante en la tarea de verificación del locutor dependiente del texto, si se emplea una agrupación de promediado global, este orden se descuida y los resultados obtenidos para las métricas de rendimiento son demasiado altos. Por lo tanto, el primer enfoque propuesto en esta tesis es un mecanismo de alineamiento que se utiliza para reemplazar el uso del promediado global como mecanismo de agrupación. Este mecanismo de alineamiento permite mantener la estructura temporal y codificar la frase y la identidad del locutor en un supervector. Como mecanismo de alineamiento, se pueden utilizar diferentes tipos de planteamientos como los modelos ocultos de Markov o los modelos de mezcla de Gaussianas. Además, durante el desarrollo de este mecanismo, también observamos que la falta de bases de datos de entrenamiento más grandes es otro problema importante para crear estos sistemas. Por lo tanto, también hemos introducido una nueva filosofía de arquitectura basada en el enfoque de destilación de conocimiento. Esta arquitectura es conocida como arquitectura profesor-estudiante y proporciona robustez a los sistemas durante el proceso de entrenamiento y contra un posible sobreajuste debido a la falta de datos. En esta parte, se propone otro enfoque alternativo para centrarse en los instantes relevantes de la secuencia y mantener la información fonética, dicho enfoque consiste en la auto-atención multi-cabezal. La arquitectura propuesta para utilizar las capas de auto-atención multi-cabezal también introduce incrustaciones fonéticas y capas de memoria para mejorar la discriminación entre locutores y expresiones. Además, para completar la arquitectura con las técnicas anteriores, se ha incorporado otro enfoque en el que se han

introducido dos vectores aprendibles que se denominan tokens de clase y de destilación. Utilizando estos tokens durante el entrenamiento, se mantiene la información temporal y se codifica en los tokens, de manera que al final se obtiene un descriptor global de los enunciados similar al supervector.

Además de los enfoques anteriores para obtener representaciones robustas, la otra parte principal de esta tesis se ha centrado en la introducción de nuevas funciones de pérdida para entrenar arquitecturas de redes neuronales profundas. Las funciones de pérdida tradicionales han proporcionado resultados razonablemente buenos para muchas tareas, pero no suelen estar diseñadas para optimizar la tarea objetivo. Por esta razón, hemos propuesto varias funciones de pérdida nuevas como objetivo para entrenar arquitecturas de redes neuronales profundas que se basan en las métricas finales de verificación. El primer enfoque desarrollado para esta parte se inspira en el Área Bajo la Curva ROC. Así que hemos presentado una aproximación diferenciable de esta métrica denominada *aAUC loss* para entrenar con éxito una red neuronal de tripletes como back-end. Sin embargo, la selección de los datos de entrenamiento tiene que ser cuidadosamente realizada para llevar a cabo este back-end, por lo que esto supone un alto coste computacional. Por lo tanto, hemos desarrollado varios enfoques para aprovechar el entrenamiento con una función de pérdida orientada a la tarea objetivo pero manteniendo la eficiencia y velocidad del entrenamiento multiclase. Para implementar estos enfoques, se han empleado como objetivo de entrenamiento la aproximación diferenciable de las siguientes métricas de verificación, la función de coste de detección (*aDCF*) y el coste de la relación de log-verosimilitud (CLLR). Al optimizar las arquitecturas de redes neuronales profundas para minimizar estas funciones de pérdida, el sistema aprende a reducir los errores en las decisiones y las puntuaciones producidas. El uso de estos enfoques también ha demostrado una mejor capacidad para aprender representaciones más generales que el entrenamiento con otras funciones de pérdida tradicionales. Por último, también hemos propuesto un nuevo back-end sencillo que emplea la información aprendida por la matriz de la última capa de la arquitectura de redes neuronales profundas durante el entrenamiento con la *aDCF loss*. Utilizando la matriz de esta última capa, se entrena un modelo de inscripción con un vector aprendible para cada identidad de inscripción para realizar el proceso de verificación.

Contents

List of Figures

List of Tables

List of Acronyms

AER	Assignment Error Rate
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASM	Active Shape Models
ASR	Automatic Speech Recognition
AUC	Area Under the ROC Curve
BDK	Bayesian Dark Knowledge
CE	Cross-Entropy
CLLR	Cost of Log-Likelihood Ratio
CNN	Convolutional Neural Network
DCF	Detection Cost Function
DCT	Discrete Cosine Transform
DER	Diarization Error Rate
DET	Detection Error Trade-off
DNN	Deep Neural Network
DTW	Dynamic Time Warping
EBGM	Elastic Bunch Graph Matching
EER	Equal Error Rate
EM	Expectation Maximization
FA	Factor Analysis
FA	False Alarm
FAR	False Alarm Rate
FB	Filter Bank
FF	Feed-Forward
FFT	Fast Fourier Transform
FM	False Match
FN	False Negative
FNM	False Non-Match
FP	False Positive
FPR	False Positive Rate
FR	False Rejection
FRR	False Rejection Rate
GMM	Gaussian Mixture Model

GRU	Gated Recurrent Units
H	Entropy
HMM	Hidden Markov Model
ICA	Independent Component Analysis
JFA	Joint Factor Analysis
KD	Knowledge Distillation
KLT	Karhunen-Loeve Transform
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
LFW	Labeled Faces-in-the Wild
LLR	Log-Likelihood Ratio
LPC	Linear Predictive Coding
LR	Logistic Regression
LRE	Language Recognition Evaluation
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori
MFCC	Mel-Frequency Cepstral Coefficients
MFM	Max-Feature Map
MOBIO	Mobile Biometrics
MSA	Multi-head Self-Attention
NAP	Nuissance Attribute Projection
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
PCA	Principal Component Analysis
PDF	Probability Density Function
PIN	Personal Identification Number
PLDA	Probabilistic Linear Discriminant Analysis
PLP	Perceptual Linear Predictive
RBF	Radial Basis Function
RE	Random Erasing
ReLU	Rectified Linear Units
ResNet	Residual Neural Network
ROC	Receiver Operating Characteristic
RTVE	Radio Televisión Española
SAD	Speech Activity Detection
SCPD	Speech Change Point Detection
SDC	Shift Delta Cepstrum
SdSV	Short-duration Speaker Verification
SIFT	Scale Invariant Feature Transform
SGD	Stochastic Gradient Descent
SGLD	Stochastic Gradient Langevin Dynamics
SRE	Speaker Recognition Evaluation
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine

TDNN	Time Delay Neural Network
TFA	Tied Factor Analysis
TMFA	Tied Mixture of Factor Analysis
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TVM	Total Variability Modelling
UBM	Universal Background Model
VAD	Voice Activity Detector
ViT	Vision Transformer
VQ	Vector Quantizer
WGB	Weight Gaussian Back-end

Part I

Introduction and Theoretical Background

1 Introduction

1.1 Motivation

During the last decades, the evolution of systems is leading to a more natural and intuitive human interaction with technology. Technology has become a crucial part of day-to-day life since mobile phones, cameras, laptops, and other technological devices are widely used by many people in their daily lives to create content and keep connected to each other. Besides, nowadays, the vast majority of services are accessed through smart machines based on systems with Artificial Intelligence (AI) algorithms. These services include government services, banking, e-commerce, hotel reservations and many other fields related to defense, education, work, business and travel. Therefore, this large expansion in the use of these devices and services has led to the need of new systems that improve access security. As a result, the use of systems based on biometric features to recognize individuals and grant them access is becoming increasingly common. Motivated by the exponential growth of systems based on AI algorithms and their applications, multimedia technologies have also profoundly changed the way to interact, communicate, learn, entertain, and process the audiovisual information available.

Likewise, the rise of these technologies has also promoted the generation of a large amount of new multimedia and broadcast data like news, talk shows, debates or series through platforms of video on demand (Youtube, Netflix, HBO, etc.). To increase the value of all available multimedia content and enhance the user experience, this content has to be analyzed and catalogued, which is very expensive to do manually. For this reason, the development of new efficient tools is needed to process the available audiovisual content and extract relevant information from these archives. This fact has led to a wide extension of powerful systems based on machine learning algorithms such as Deep Neural Networks (DNNs). These algorithms have shown great performance in many current audiovisual applications with a large amount of data. Thus, the successful development of these systems has motivated the application of the same approaches in more challenging tasks such as the automatic description of audiovisual archives from different domains or in cases where the available data is limited. In recent years, the processing of audiovisual information has also extended to the digitalization of older repositories, which are composed of historical documents to be preserved. Digitalization with new tools has made possible to extract important information, which helps to improve the accessibility of these audiovisual archives.

Motivated by the above insights, this thesis has focused on two main challenges. One of them is the introduction of new tools to analysis and catalogue the available multimedia data. The second is the development of biometric systems which allow us to create models with better capacities to extract relevant information from the data and personalize the system. This personalization allows enhancing the human interaction with biometric recognition systems. In addition, the various approaches implemented also aim to improve the security and robustness of these systems against attacks. Two examples of the applications that have motivated the development of these systems during this thesis are shown in Figure 1.1 and 1.2. Figure 1.1 depicts an example of the application of these systems for cataloging multimedia content where, from an audiovisual file, it is determined whether two characters are appearing at each moment and also, if they are speaking. While in Figure 1.2, another type of application is represented in which speech utterances of different users are initially stored in the system and when a person wants to access to the system, the system compares the new utterance with the stored utterances and decided whether this person is registered and can access or not. To carry out the development of these systems, different biometric techniques have been employed. In the following section, biometrics and the different types of systems are introduced.

1.2 Biometrics

Biometrics is the branch of science that deals with measuring and analyzing biological data. In the technological field, biometrics is defined as a recognition technology that detects, assigns, verifies and authenticates true identity of the individual based on certain human characteristics which are unique to each individual. For this reason, the use of these characteristics is becoming widespread for the development of recognition systems.

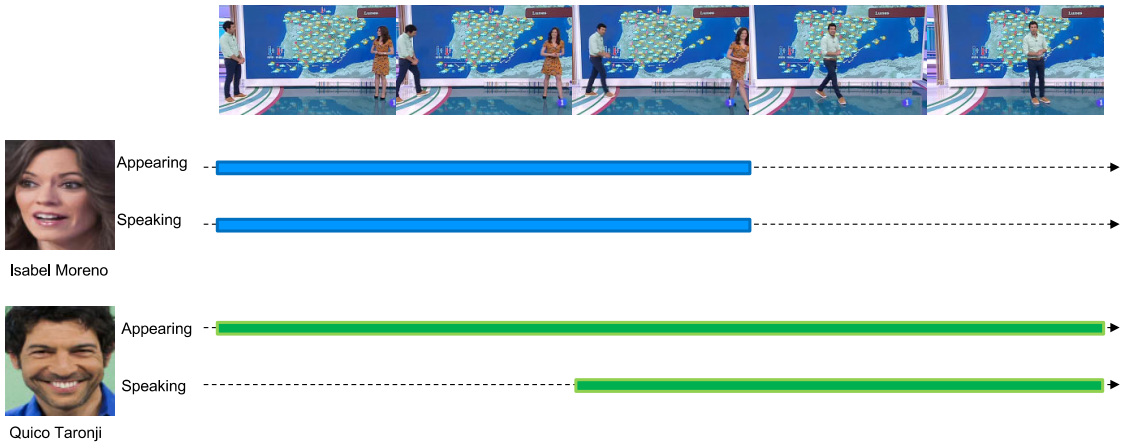


Figure 1.1: Example of analysis and cataloguing multimedia content.

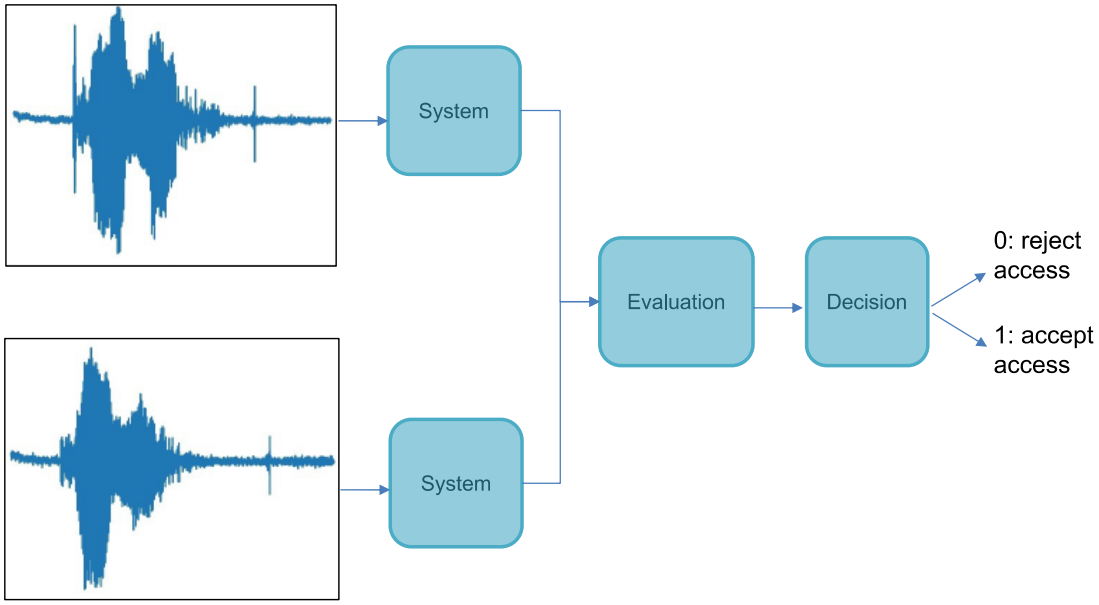


Figure 1.2: Example of a biometric recognition system for a secure access.

Traditionally, recognition systems have employed standard techniques based on possession, such as keys, cards or passports, or on knowledge, such as passwords or personal identification number (PIN). However, the introduction of biometric technology in recognition systems has improved security and privacy in accessing these systems. Biometric systems can be categorized into two modalities in function of the characteristic employed as shown in Figure 1.3. On one side, there are biometrics systems that use physiological characteristics such as Face, Fingerprint, DNA, Hand geometry, Iris and Ear. While the other group is based on the use of behavioural characteristics such as Voice, Signature and Keystroke. Both modalities are composed of unique characteristics which are intrinsically associated with the individual and cannot be transferred or copied for fraudulent used.

To develop a specific biometric recognition system, the choice of the biometric modality used depends upon the application requirements since every biometric characteristic

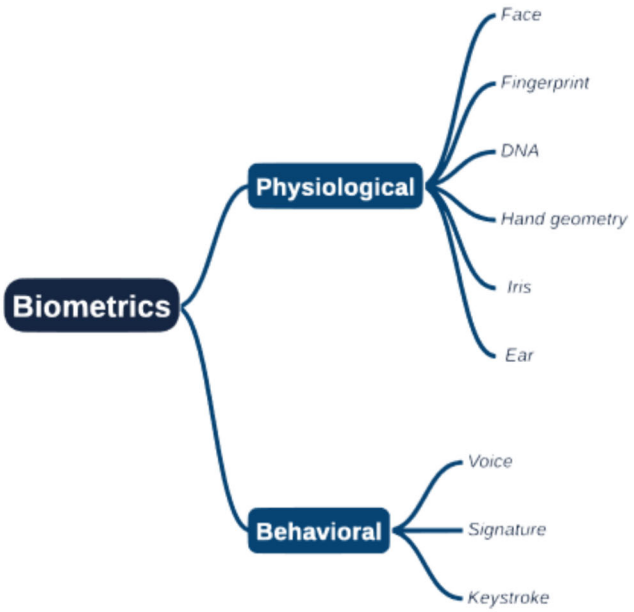


Figure 1.3: Biometric modalities.

has its advantages and disadvantages. Moreover, specific applications can only employ certain attributes. Recently, the trend is the fusion of two or more biometric characteristics to design a multimodal biometric system to overcome possible problems of using a single characteristic. There are a large number of applications where these biometric recognition systems are increasingly used such as:

- Surveillance: to improve the security and the reliability of access to certain resources.
- Personalization: to enhance the interaction with domotic applications.
- Authentication: to verify the identity of an individual.
- Forensic: to help prove the identity of the criminal and discharge the innocent focusing on inspecting digital data to discover relevant information.
- Digitalization or Cataloguing: to analyze and describe the multimedia content.

Since these systems began to revolutionize the way in which recognition was performed in previous applications, face and voice modalities have been two of the most preferred biometric data. This is motivated by the fact that both attributes are easy to obtain without the cooperation of the person and using a easy-friendly procedure. Besides, the low-cost to capture face and audio through sensors in smartphones, laptops and tables has given an advantage of these modalities compared to other biometrics.

Recognition systems based on face and voice attributes can work in several operating modes as Figure 1.4 depicts. Typically, two of these operating modes are mainly employed: the identification mode and the verification mode. The former consists of comparing a biometric sample collected from a subject with all the identities from the

database to find the subject identity. On the other hand, the verification mode is a process where two biometric samples are compared to determine whether both samples belong to the same identity or not. Although the two previous have been the most analyzed in the literature, the rest of the operating modes are also needed to improve the systems since these modes help to simplify the process made by the identification or verification process. For example, face or speaker detection allows cleaning data without face or voice information to determine the identity.

To summarise, in this thesis, we have worked in three specific fields: face verification, speaker verification and language recognition. Besides, speaker verification field involves two different modes. Depending on the constraints of the lexicon content of the utterances, speaker verification is usually divided into two categories: text-independent and text-dependent speaker verification. In the former, there are no restrictions on the uttered phrase pronounced which can produce a large variability in the duration of the utterances, while in the latter requires the same constraints in the lexicon content. In this dissertation, we have focused on text-dependent speaker verification.

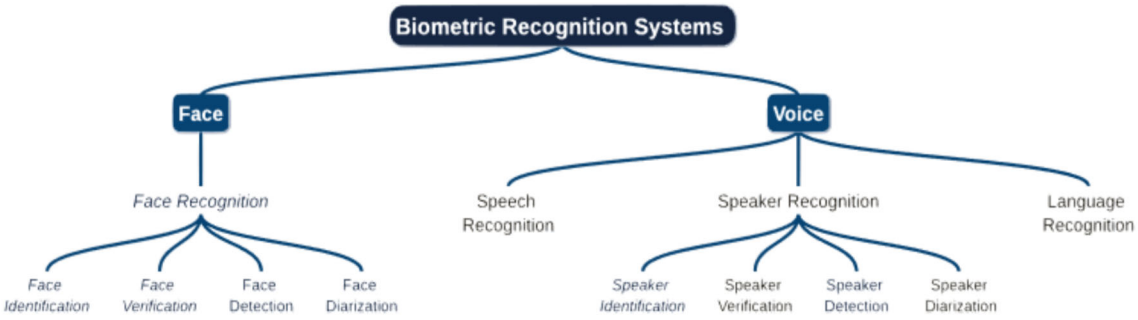


Figure 1.4: Types of face and voice recognition.

1.3 Thesis Objectives

Although face and voice recognition systems are mature technologies, there are still some challenging tasks which need further improvement and continued research. Therefore, the main goal of this thesis is the exploration of new approaches based on DNNs for biometric recognition systems using face and voice traits. In particular, focusing on those techniques that improve the generalization capacity when facing new situations and also, to train robust systems especially in the case of training data is limited. This broad objective can be divided into the following more specific ones:

- Applying existing and common approaches for face and speaker verification to establish a baseline system based on DNNs.
- The exploration of different alternatives to correctly encode phrase and speaker information for text-dependent speaker verification.

- Introducing techniques to provide more robustness to the DNNs employed in our systems.
- The development of several loss functions which are more suitable for face, speaker and language verification tasks.
- The creation of new back-ends to perform the face and speaker verification process.
- Applying the different approaches developed to the automatic indexing of multimedia content.

1.4 Structure of the Thesis

This dissertation is divided into four main parts, as shown in the conceptual map in Figure 1.5. In the first one (Chapter 1, 2 and 3), an introduction to the motivation and goals of this thesis, and also, to the state-of-the-art of the biometric recognition systems is provided. The second part (Chapter 4, 5, 6, 7 and 8) is dedicated to the different representation learning approaches developed during this dissertation. In the third part (Chapter 9, 10, 11, 12, 13 and 14), the development of specific metric learning loss functions are presented which are more suitable for the verification task than the existing ones. To finalize, Chapter 15 contains the main conclusions of the research presented in this thesis and future research lines. The organization of the different chapters and the description is as follows:

- **Chapter 1. Introduction:** In the present chapter, the motivation and goals of this thesis are introduced. Moreover, a brief background of the biometric science and recognition systems is provided.
- **Chapter 2. State-of-the-art Face and Voice Recognition:** In this chapter, we review the state-of-the-art of recognition systems that are based on the same characteristics employed to carry out this dissertation.
- **Chapter 3. Biometric Recognition Systems:** This chapter introduces the biometric recognition systems and reviews the techniques used in this thesis for each part. Furthermore, the experimental datasets used for the different systems are presented, as well as the performance metrics applied to evaluate the different approaches throughout this thesis.
- **Chapter 4. File Level Representation using Deep Neural Network Embeddings:** Once the theoretical background has been presented in the previous chapters, the baseline systems developed for face and speaker verification are introduced. Apart from baseline systems, we analyze some of the issues found with the global average pooling if it is used in the models which have promoted part of the new approaches presented in the following chapters.

- **Chapter 5. Deep Neural Network Supervectors:** Motivated by the issues presented in Chapter 4 for the text-dependent speaker verification systems, in this chapter, an alignment mechanism using Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM) information is introduced to replace the global average pooling. This kind of mechanism allows to keep the temporal structure of each phrase and obtain a neural network supervector with the speaker and phrase information since both are relevant to the text-dependent speaker verification task. The results show the relevance of keeping the phonetic order to obtain a great performance in this task. The research described in this chapter generated the following publications:

  [1] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Differentiable Supervector Extraction for Encoding Speaker and Phrase Information in Text Dependent Speaker Verification." *Proceedings of IberSPEECH 2018*, pp. 1–5. Barcelona, Spain.

  [2] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Supervector Extraction for Encoding Speaker and Phrase Information with Neural Networks for Text-Dependent Speaker Verification." *Applied Sciences*, vol. 9, no. 16, p. 3295, 2019.

  [3] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Optimization of the area under the ROC curve using neural network supervectors for text-dependent speaker verification." *Computer Speech & Language*, vol. 63,p. 101078, 2020

- **Chapter 6. Knowledge Distillation with Teacher-Student Architectures:** Using as starting point the methodological framework introduced in Chapter 5, this chapter presents an approach to improve the generalization ability and provide more robustness. This approach is based on Knowledge Distillation (KD) which consists of two neural networks, known as Teacher and Student, where the student is trained to replicate the predictions from the teacher, so it learns their variability during the training process. The results confirm that KD architectures combined with techniques that augment the variability of the input signals can improve the single model presented in Chapter 5. The work presented in this chapter has resulted in the following publications:

  [3] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Optimization of the area under the ROC curve using neural network supervectors for text-dependent speaker verification." *Computer Speech & Language*, vol. 63,p. 101078, 2020

  [4] V. Mingote, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, "Knowledge Distillation and Random Erasing Data Augmentation for Text-Dependent Speaker Verification." *Proceedings of ICASSP 2020*, pp. 6824-6828. Barcelona, Spain.

- **Chapter 7. Multi-head Self-Attention Mechanisms with Memory Layers:** The methodological framework introduced in Chapter 5 demonstrated the relevance of keeping the phonetic knowledge for text-dependent speaker verification

task. Thus, in this chapter, we combine the temporal attention of multiple parallel heads with memory layers and the phonetic embeddings extracted from a phonetic classification network, which helps to guide the attention mechanism with the role of the positional embedding. These results have led to the following publication:

[5] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Memory Layers with Multi-Head Attention Mechanism for Text Dependent Speaker Verification." *Proceedings of ICASSP 2021*, pp. 6154-6158. Toronto, Canada.

- **Chapter 8. Class and Distillation Tokens for Multi-head Self-Attention Systems:** As an evolution of the approaches shown in the previous chapters, this chapter introduces a Bayesian estimation of a learnable global representation known as class token to replace the global average pooling mechanism. Besides, a distillation token using KD approach is combined with the class token. This distillation token is trained to mimic the predictions from the teacher network. The research described in this chapter generated the following publication:

  [6] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Class Token and Knowledge Distillation for Multi-head Self-Attention Speaker Verification Systems." arXiv preprint arXiv:2111.03842, *Submitted to IEEE Transactions on Audio, Speech and Language*, 2021.

- **Chapter 9. Analysis of Different State-of-the-art Training Loss Functions for Verification Systems based on Deep Neural Networks:** In this chapter, we revisit the most employed loss functions for training Deep Neural Networks (DNNs). These loss functions can be categorized into two groups: identification or classification loss functions, and verification or metric learning loss functions. Throughout this chapter, we define the loss functions of each group used as references in this dissertation. Moreover, we analyze the main drawbacks of each one.

- **Chapter 10. Optimization of the Area Under the ROC Curve for Training Deep Neural Networks:** In view of the disadvantages of previous loss functions, we introduce a new loss function to train a triplet neural network as back-end for detection tasks. This loss is based on an approximation of the Area Under the Receiver Operating Characteristic Curve (aAUC), which is one of the performance metrics employed. Besides, we show that this loss function is suitable for any recognition task independently of recognizing faces, speakers or languages. The work presented in this chapter has resulted in the following publications:

  [3] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Optimization of the area under the ROC curve using neural network supervectors for text-dependent speaker verification." *Computer Speech & Language*, vol. 63,p. 101078, 2020

  [7] V. Mingote, D. Castan, M. McLaren, M. K. Nandwana, A. Ortega, and E. Lleida, A. Miguel, "Language Recognition using Triplet Neural Networks." *Proceedings of INTERSPEECH 2019*, pp. 4025-4029. Graz, Austria.

- **Chapter 11. Approximated Detection Cost Function as Training Objective Loss:** In this chapter, a novel training objective function called approximated Detection Cost Function (aDCF) is proposed to replace the classical Cross-Entropy loss using the same type of multi-class architectures. This loss function is inspired by DCF, which is another performance metric for verification task. DCF measures the decision errors produced in verification systems. These results have led to the following publications:

  [8] V. Mingote, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, "Optimization of False Acceptance/Rejection Rates and Decision Threshold for End-to-End Text-Dependent Speaker Verification Systems." *Proceedings of INTERSPEECH 2019*, pp.2903-2907. Graz, Austria.

  [9] V. Mingote, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, E. "aDCF Loss Function for Deep Metric Learning in End-to-End Text-Dependent Speaker Verification Systems." *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 772-784, 2022, doi: 10.1109/TASLP.2022.3145307.

- **Chapter 12. Training Enrollment Models by Network Optimization:** This chapter presents a novel and straightforward methodology to perform the verification process instead of a complex back-end. Using this approach, we propose to leverage the knowledge acquired by a DNN to model the identities of the training set since the last layer of the DNN can be interpreted as an embedding dictionary representing the training identities. Thus, after the initial training phase, we introduce a learnable vector for each enrollment identity, and this training process is lead by aDCF loss proposed in Chapter 11. The research described in this chapter generated the following publication:

  [10] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Training Speaker Enrollment Models by Network Optimization." *Proceedings of INTERSPEECH 2020*, pp. 3810-3814. Shangai, China.

- **Chapter 13. Multimodal Diarization Systems by Training Face Enrollment Models as Identity Representations:** In this chapter, we introduce the multimodal systems developed for the diarization of audiovisual files and the assignment of an identity to each segment. Furthermore, in the face verification system, we have included the novel approach presented in Chapter 12 to train an enrollment model for the characterization of each identity. The work presented in this chapter has resulted in the following publications:

  [11] V. Mingote, I. Viñals, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, "ViVoLAB Multimodal Diarization System for RTVE 2020 Challenge." *Proceedings of IberSPEECH 2020*, pp. 76-80. Valladolid, Spain.

  [12] V. Mingote, I. Viñals, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, "Multimodal Diarization Systems by Training Enrollment Models as Identity Representations." *Applied Sciences*, vol. 12, no. 3, p. 1141, 2022.

- **Chapter 14. Log-Likelihood Ratio Cost as Training Objective Loss:** Motivated by the great performance obtained in Chapter 11 training with aDCF loss, this chapter presents an alternative approach to optimize the parameters of a neural network using a loss function based on Log-Likelihood Ratio Cost (CLLR). This function is an application-independent metric that measures the cost of soft detection decisions over all the operating points. Thus, prior or relevance cost parameters assumptions are not employed to obtain it. These results have led to the following publication:

  [13] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Log-Likelihood-Ratio Cost Function as Objective Loss for Speaker Verification Systems." *Proceedings of INTERSPEECH 2021*, pp. 2361-2365. Brno, Czechia.

- **Chapter 15. Conclusions and Future Work:** This last chapter contains the main conclusions that emerge from this thesis, as well as a proposal of future research lines.

1.5 Collaborations and Research Stays

All the research presented in this dissertation has been conducted within the Voice Input Voice Output (ViVoLab) group (University of Zaragoza, Spain), under the supervision of Dr. Antonio Miguel Artiaga. Additionally, I had the opportunity of benefiting from one research stay in the context of my PhD, which is listed below:

- September 2018 - December 2018: *STAR Lab, SRI International, California, United States.* This stay was an opportunity to discuss closely with Diego Castan, Mitchel McLaren and Mahesh Nwanda in the context of the analysis of language influences to recognize. Apart of getting familiar with the data to be analyzed, we sat up a proper framework for its analysis which is described in Chapter 10.

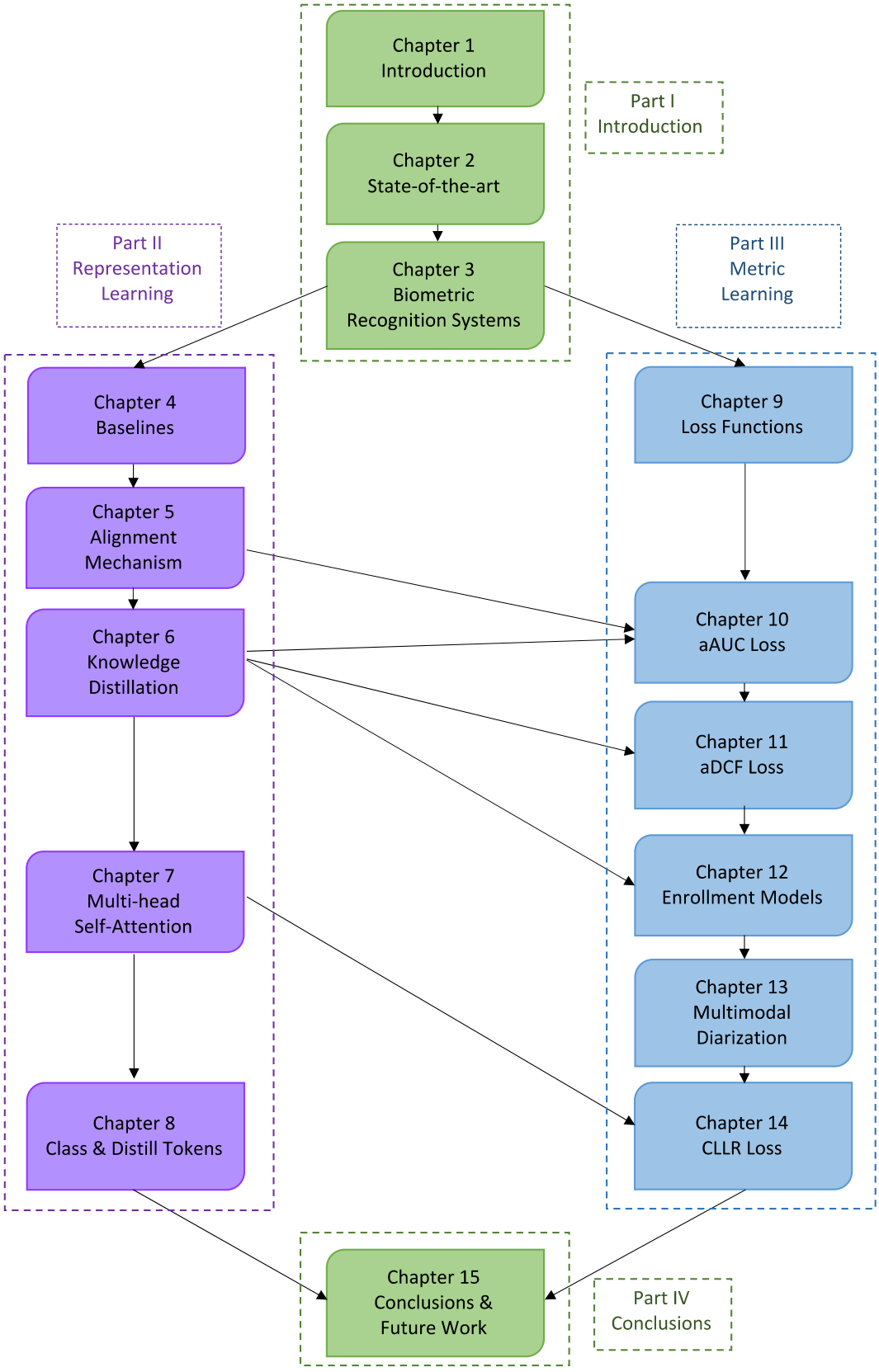


Figure 1.5: Conceptual map of this thesis showing the two stream of research developed in parallel.

2

State-of-the-art Face and Voice Recognition

In this chapter, we summarise in three state-of-the-art reviews the most important existing approaches in the literature from the fields of face recognition, speaker recognition and language recognition. During their development, the fields of speaker and face recognition have evolved in parallel, and sometimes approaches developed in one field have subsequently been implemented in the other. On the other hand, the evolution of speaker and language recognition has traditionally progressed together as they are closely related and share many common aspects. In general, the most relevant speaker recognition approaches have been adapted to the language recognition systems. To make the following reviews of the state of the art, we have taken into account some relevant reviews for each field: face recognition [14–20], speaker recognition [21–28] and language

recognition [29–31]. In addition, we have divided each of these state-of-the-art review into the same periods to facilitate the comparison between them.

2.1 State-of-the-art Face Recognition

2.1.1 Early Beginnings

The earliest studies on face recognition can be traced back to the 1950s in psychology [32], and 1960s in engineering literature [33]. In the latter, the first semi-automatic face recognition system was developed, which could extract usable feature points and calculate distances between two faces. Nevertheless, the research field on automatic face recognition is considered to have started in the 1970s [34, 35] where human intervention was not involved in the recognition process. In these initial attempts, face recognition was performed with feature-based approaches. These approaches used straightforward image processing techniques to extract a vector of important regions of the face. After that, these vectors were compared with a simple distance measure or using the nearest neighbour classifier to obtain the identity label.

2.1.2 The 1980s-2000s

During the early 1980s, the field of face recognition remained dormant. The slow progress made in these years was motivated by the fact that image capture was a difficult task. Therefore, few face image databases were available. However, since the late 1980s, research interest in this field grew significantly. One of the first approaches was a standard linear algebra technique applied to perform the face recognition process [36]. This technique was based on a statistical approach known as Principal Component Analysis (PCA) and as Karhunen-Loeve transform (KLT) in the pattern recognition literature [37]. Using this method, feature point selection and dimension reduction were applied on the face images to obtain representations of the face in a new lower-dimensional space. These face representations were known as eigenpictures or eigenfaces. Based on the above work, [38] was one of the first truly successful approaches to automatic face recognition using eigenfaces as classification features for face detection and identification. To carry out face recognition, a Euclidean distance was applied on the eigenfaces. Nevertheless, as this metric was too simple, [39] proposed to employ a probabilistic similarity measure.

The eigenface approaches showed some problems when the face images to be recognized had large variations in illumination, light direction and facial expression. Therefore, to solve these issues, another linear projection method was developed, which was based on Fisher's Linear Discriminant Analysis (LDA) [40, 41]. The classification features obtained with this method became known as Fisherfaces. Apart from this approach, several variations and extensions of the eigenfaces and Fisherfaces methods were proposed. These approaches slightly improved the way of dealing with illumination and expres-

sion variations. However, these methods still had one main drawback since the PCA and LDA techniques only model simple Euclidean distances between samples. Hence, these techniques failed to discover the underlying complex ground truth structure. Therefore, a generalization of PCA was proposed, which was Independent Component Analysis (ICA) [42]. ICA employed a higher-order statistics to have more power of representation than PCA.

PCA, LDA and ICA are the best known holistic or appearance-based methods for face recognition. Nonetheless, during the same decades, other types of feature extraction methods were developed. A widely employed local or feature-based approach was proposed in [43] which was Elastic Bunch Graph Matching (EBGM). This technique consisted of representing each face with a labeled graph. To generate it, the Gabor wavelet transform was applied around the chosen specific facial landmarks, called fiducial points. For the recognition process, the labeled graphs of each face image were compared with a Euclidean distance metric between equivalent landmarks. During these years, another model-based approach was proposed. This method was Active Shape Model (ASM) [44, 45] which consisted of flexible statistical models of the shapes of objects. To train this kind of models for recognition, a prior model with plausible location of shapes of objects was employed and when a new image was available, it was iteratively deformed to find the best match position between the new image and the model. On the other hand, simultaneously, another holistic method based on a deterministic discrete transform widely used for image compression was introduced for face recognition. This technique was Discrete Cosine Transform (DCT) [46] which was proposed since it was more efficient with the variations of illumination, occlusions, different scales, poses and rotations. In addition, this approach could be applied to obtain local or global features. In [47], DCT was computed for each block of the face image, and a set of coefficients was kept as a feature vector. Once these feature vectors were obtained, a Vector Quantizer (VQ) codebook of feature vectors was generated by using the k-means algorithm to cluster the data. Then, a minimum distance classifier was applied to find the closest features for recognition.

In parallel to the development of all the presented feature representations, several techniques were proposed to substitute the simple classifier commonly used. Given the success of Hidden Markov Models (HMM) [48] in speech recognition previously, the use of HMM as a classifier for face recognition was introduced in several works. In [49, 50], strips of raw pixels covering areas such as forehead, eye, nose, mouth and chin were extracted from a face image and converted into a chronological sequence. Then, the sequence was modeled with a linear left-right HMM model in which the number of states was defined by the areas covered with the strips. In function of the advantages of applying an HMM to the recognizer classifier, this approach was also developed with different feature extraction approaches, such as eigenfaces [51] and DCT [52]. Moreover, another kind of classifier was introduced, which was Support Vector Machine (SVM) [53]. SVM classifier could be used with different features as input such as raw pixels, PCA [54], ICA [55] or EBGM [56], and it was also considered one of the most effective algorithms

for classification problems due to the ability to define complex nonlinear classification boundaries.

2.1.3 The 2000s-2012s

In the early 2000s, another classifier successfully applied in the speech field was implemented to perform face recognition. Gaussian Mixture Model (GMM) [57, 58] was a probabilistic model that was composed of a combination of linear Gaussian models. This approach allowed more complex data to be represented by using a mixture that can handle variations in the data that a single Gaussian model can not capture. Furthermore, as occurs with previous classifiers, features of different types were employed to train the GMM classifier [59–61].

Focusing on the existing problems when face images had large variations, a new local appearance method was also created since this type of methods was more stable to local changes such as occlusions, misalignment and expression changes. This method was Local Binary Patterns (LBP) [62–64] which was a representative method in which the LBP operator was applied to transform the face images and divide them into small regions. Over these regions, histogram features were extracted, and Chi-square distances between the corresponding histograms were calculated.

In view of the advances against illumination and occlusion issues, new approaches were developed to address the problems of large pose changes from a probabilistic point of view. [65, 66] proposed a novel generative model to describe pose variations in facial data. This approach was based on a linear statistical model called Tied Factor Analysis model (TFA). TFA estimated the hidden variables representing identity and pose, defining the new identity space, using the Expectation Maximization (EM) algorithm. This recognition model was not based on distance comparisons, in this case, the likelihoods were calculated under the assumption that all images of the face of a person share the same vector in the identity space since the hidden identity variable was tied. As this previous probabilistic approach produced good results in face recognition, similar probabilistic techniques were developed. The first of them was Probabilistic Linear Discriminant Analysis (PLDA) [67, 68], which was a probabilistic version of Fisherfaces. PLDA was a generative model that projected data to a latent space where data of the same class were modeled with the same distribution. Thus, the distribution was able to represent the data, as well as the within-class variability of the data. A model similar to PLDA was developed independently for the speaker recognition task, which was known as Joint Factor Analysis (JFA). JFA was an evolution of the GMM representations that models and removes within-class variability using a low-dimensional subspace to obtain a more reliable class model. In addition, JFA employed tied hidden variables to model identities and channel variability. In [69], DCT coefficients were used as facial features, and JFA was applied to address the face recognition task. As the last approach following this probabilistic philosophy, in [70], an evolution of JFA was applied to face recognition. This evolution was called Total Variability Modelling (TVM) or also referred to as i-vector modelling, which

was based on a generative probabilistic model to represent the data in a low-dimensional space. TVM can be seen as a JFA with a single type of hidden variables. Therefore, this technique was proposed as a feature extractor, and in [70], a PLDA was employed as face classifier.

2.1.4 The 2012s and onwards

In previous decades, several attempts were made to introduce Neural Networks (NN) in face recognition but, these approaches did not achieve relevant success due to the limitation in available training data and computational resources. However, after the great success of AlexNet in the ImageNet competition in 2012 [71], deep learning based methods gained popularity. Mainly, these deep learning methods consisted of using deep Convolutional Neural Networks (CNN) to train powerful models for several vision tasks. In recent years, these deep CNN-based methods have become the dominant approach to address face recognition due to the availability of large training databases and advances in computation resources, such as CPU cores and GPUs. [72] was one of the first works where it was shown that deep CNN would be very successful in face recognition. This work is known as DeepFace and used the network architecture proposed in AlexNet with the softmax loss function. Apart from the AlexNet architecture, [72] also experimented with a siamese network [73]. This approach consisted of replicating the trained neural network twice without the final classification layer and processing two input images to predict whether both images belonged to the same person or not. Then, the decision process was based on a similarity metric [74].

Due to the relevant success achieved with DeepFace, research began to focus on improving architectures to extract features representing the face images. In [75], the Facenet model was defined to train a backbone architecture such as CNN or inception network [76]. Once this backbone network was trained, a backend network was incorporated using a triplet loss function as the final layer. The triplet network was an evolution of the siamese approach where instead of a single pair of images, three face images were processed to train the network and learn to determine whether each pair belonged to the same identity or not. In [77], an approach similar to [75] was developed. The main difference was the introduction of a very deep CNN to implement the backbone. Most of the previous approaches used typical activation functions, such as Sigmoid, Tanh, Rectified Linear Units (ReLU), etc. Nevertheless, [78] introduced a light CNN framework that presented Max-Feature-Map (MFM) instead of using ReLU activation to learn a more robust face representation. Increasing network depth led to difficulties in network training, such as vanishing gradients and convergence problems. Motivated by these issues, Residual Neural Networks (ResNet) were developed [79], which included shortcut connections within layers to enable cross-layer connectivity. In face recognition, the first work with these networks was [80, 81]. An evolution of ResNet was presented in [82], this novel architecture called Wide Residual Neural Networks (WideResNet) proposed to increase the width and decrease the depth of ResNet.

Most of the previous works had adopted to train neural networks using the same loss function, which is Cross-Entropy (CE) loss with softmax output units [71]. The choice of this loss function is also important to correctly train the architectures and obtain sufficiently discriminative representations. Different studies have shown that with CE loss, the separability of the representations is encouraged. However, this loss is not effective enough to enhance the intra-class compactness. Therefore, the design of new loss functions has been widely investigated in recent years to find the most suitable loss function for training deep learning systems. These efforts have focused on two lines of study, on the one side, the redesign of the identification loss function, and on the other side, the verification loss functions. The former is composed of CE loss combined with a complementary loss such as Center loss [83] or Ring loss (RL) [84], and other studies have focused on its variants such as Angular Softmax loss (A-Softmax) [81], CosFace [85], or Additive Angular Margin loss (ArcFace) [86]. While the latter is based on metric learning approaches such as triplet neural network [75] or contrastive loss [87]. Although these approaches had already been introduced, in recent years, this metric learning philosophy has gained more attention. For example, this thesis introduces the use of a new loss function based on the Area Under the ROC Curve (AUC) combined with triplet neural networks to improve the discriminative power of learned facial representations. The advantage of this loss function is that the triplet neural network is directly optimized with a cost function close to one of the desired performance measures.

2.2 State-of-the-art Speaker Recognition

2.2.1 Early Beginnings

The first works on speaker recognition date back to the early 1960s, when a physiological model of human voice production was developed [88]. This model and other parallel research laid the basis for understanding the speech analysis for representing in speaker recognition systems. Using this knowledge in [89], the first speaker recognition system known as Voiceprint Identification emerged. In this system, a visual representation of the speech signal called a spectrogram was inspected by pattern matching and scored by a human. During the same decade, the research to develop a fully automatic speaker recognition system was growing, and in 1963, the first automatic speaker recognition system was developed [90]. This system also relied on the correlation of spectrograms and the use of spectral pattern matching. As an evolution of these earlier works, a new approach proposed in [91] took into account formant frequencies, voice pitch period and speech energy to perform the speaker verification. The first successful systems were all text-dependent, but a breakthrough in text-independent occurred in the late 1960s when [92] proposed to apply cepstral analysis to obtain measurable features in the human voice for speaker recognition. The study presented in [93] demonstrated the improvement achieved using the cepstral analysis as a feature approach for text-dependent. In this

work, a Linear Predictor (LP) was also analyzed, and a nearest neighbour classifier was employed to perform the recognition.

2.2.2 The 1980s-2000s

Contrary to what happened with face recognition, the decade of the 1980s was relevant for the development of the field of speaker recognition. The great evolution of these systems came with non-parametric template matching methods, such as Dynamic Time Warping (DTW) and Vector Quantization (VQ). [94, 95] described the use of cepstral coefficients [96] and DTW, which was a dynamic programming method to align a pair of speech utterances. Once the pair of utterances were aligned, the cumulative distance was obtained as a score to determine the identity of the utterance. In [97], the DTW method was combined with the use of filter banks as features. On the other hand, a time-independent template method, which was VQ modelling, was developed and employed combined with LP analysis [98–100]. The VQ method represented the feature vectors of each speaker using the indices of a VQ codebook obtained by clustering techniques. To find the pattern matching score for recognition, an input vector was compared with the codewords in the VQ codebook, and the codeword with the minimum distance was selected. This method substantially alleviated the computational complexity of dynamic programming based systems since no temporal alignment was required. Nevertheless, the temporal information, which may be relevant for the success of text-dependent systems, was neglected.

As an alternative to template matching approaches, a second type of classifier based on parametric modelling methods was developed. The first of these classifiers was Hidden Markov Model (HMM) [48, 101–103], which was a stochastic process where a sequence of observations was produced by a Markov chain of a finite number of states. To model the observation generation process with HMM, a Probability Density Function (PDF) is employed and several alternatives can be selected. One of the probability models to generate the data given the state was usually a mixture of Gaussian distributions. Furthermore, HMM offered more flexibility and robustness to speech signal variability. Based on HMM, a special case of this model was proposed as a second parametric model for the speaker recognition task. To create this model, a single-state HMM without taking into account the information about transitions between different states was employed. Therefore, it was a more robust model and was equivalent to a simple Gaussian Mixture Model (GMM) [104–106]. GMM can also be considered as a soft version of the VQ model in which the clusters overlap. On the other hand, this GMM approach evolved in [107] which presented the training of a general GMM called Universal Background Model (UBM) or speaker-independent world model. Then, a Maximum a Posteriori (MAP) criterion was applied to UBM to adapt the means and derive specific speaker GMMs using a few data samples.

During the late 1990s, several attempts were made to find efficient features for speech signals that could replace the spectral features. To carry out this search, the first method

applied was one of the standard statistical techniques of the state-of-the-art of face recognition, which was Principal Component Analyasis (PCA) [108–111]. Applying PCA to the speaker recognition task, Eigenvoices were obtained and combined with Gaussian models to recognize the identities. On the other hand, Independent Component Analysis (ICA) method, which was a generalization of PCA, was also introduced for speaker recognition [112, 113]. Despite the efforts to find an effective substitute for spectral features, these statistical methods have been used in combination with spectral features but, they have never been employed alone to obtain features.

In addition, in 1996, the National Institute of Standards and Technology (NIST) began conducting periodic evaluations of speaker recognition systems [114]. These technology evaluations were known as Speaker Recognition Evaluation (SRE) and provided a common framework for evaluating text-independent speaker verification systems. Therefore, in the following years, the development of these evaluations led to focus research efforts on improving text-independent speaker systems. As a consequence, research on text-dependent speaker systems received less attention.

2.2.3 The 2000s-2012s

During the 2000s and early 2010s, some important breakthroughs were achieved in the field of speaker recognition. Mainly, these advances were for application in text-independent speaker verification systems. First, the Support Vector Machine (SVM) technique became a relevant part of speaker recognition systems since it was a powerful discriminative classifier that was able to model a non-linear decision boundary instead of using a simple threshold. In [115], LPC analysis was employed to obtain feature vectors to train SVM models for speaker recognition, while in [116], Mel-Frequency Cepstral Coefficients (MFCC) vectors were extracted and used to obtain GMM supervectors that were used as input for SVM models. The GMM supervectors were built by concatenating the means of the adapted mixture components of GMM. Since then, the GMM supervectors have become one of the reference methods for speaker recognition. Furthermore, in the latter work, the Nuisance Attribute Projection (NAP) technique was introduced to reduce the effects of speaker and channel variabilities present in the supervectors.

Following this idea of compensating for variabilities, Factor Analysis (FA) techniques began to be applied in speaker recognition [117]. One of the techniques developed was Joint Factor Analysis (JFA) [118] which represented the useful information with a latent factor of the channel and speaker variability subspace separately. This technique allowed to reduce the dimensionality of the representation and to compensate for the different variabilities. Later, a new technique derived from FA emerged as the dominant approach. This technique was an evolution of JFA modell ing and was known as Total Variability Modelling (TVM), or i-vectors approach [119]. In this work, the use of factor analysis as a feature extractor was proposed to obtain the i-vector representations. With this i-vector extractor, each utterance was represented in a single low-dimensional subspace called the total variability subspace as a fixed-length feature vector instead of the separate fac-

tors for speaker and channel variables. Then, a cosine distance or SVM was employed to obtain the scores for the verification process. To improve the scoring process, [120] introduced the Probabilistic Linear Discriminant Analysis (PLDA) model, which had already been successfully used in face recognition.

For text-dependent speaker systems, the above techniques needed some modifications to be employed. In [121], the extended GMM supervector combined with the SVM was replaced by an HMM supervector as the feature vector for the SVM classifier. On the other hand, in [122], a slight modification of JFA and standard i-vectors were employed. However, these works had limitations as the lack of sufficient data in existing databases did not allow the proper development of this type of system. Hence, the performances obtained with them were not too good. For this reason, a new database was created in 2012 with the name of RSR2015 database [123, 124]. This database reactivated research in this field, which had several applications with commercial potential.

2.2.4 The 2012s and onwards

Motivated by the outstanding results of deep learning for face and speech recognition, in this decade, many advances have also been carried out in speaker recognition following the same philosophy. These approaches based on discriminative Deep Neural Networks (DNNs) have achieved substantial success in text-independent systems. However, during these years, the evolution of text-dependent systems has been different because of the little development that occurred in the previous decade. Therefore, two different sections will be used to explain separately the most recent evolution of text-independent and text-dependent speaker verification systems.

Text-Independent Speaker Verification

In the context of text-independent speaker verification, the combination of i-vector extraction and PLDA [125, 126] has remained dominant during this decade. Nonetheless, several components have been progressively replaced by DNNs. Mainly, three main streams have been investigated to improve the i-vector framework. In the first instance, the fundamental idea proposed was to introduce a DNN to extract bottleneck features (BNF) instead of acoustic features or combined with them. This kind of DNN was trained for automatic speech recognition to discriminate phonemes [127, 128] or senones [129]. Once the BNF were extracted, these features were fed into the i-vector framework. Afterwards, DNN acoustic models were employed to generate the phonetic posteriors for the frame alignments instead of using GMM-UBM in the i-vector extractors [130]. The third stream of research proposed to replace the PLDA of the i-vector framework with a DNN [131].

Recently, more ambitious proposals similar to face verification architectures have been developed to train discriminative DNNs for speaker classification with a large number of speakers as classes and then extract embeddings from an intermediate layer by re-

duction mechanisms. After that embedding extraction, the verification score is obtained by a similarity metric such as cosine similarity or using PLDA. As input to train the following approaches, several features have been employed in the literature, such as spectrogram, filter-bank and MFCC. Also, a few works have directly taken raw waveforms in the time domain. The first approach based on this philosophy was the frame-level embedding called d-vector [132, 133]. The novelty in this work was the evolution of d-vector from frame-level embedding to utterance-level embedding. In [134], a standard Convolutional Neural Network (CNN) was employed to obtain the utterance-level embedding. While in [135, 136], a Time Delay Neural Network (TDNN) was introduced to extract the embeddings, which are called x-vectors. Another popular architecture from face recognition was ResNet which was used as an extractor to obtain the speaker embeddings [137]. Besides, recurrent neural networks, such as Long Short-Term Memory (LSTM) [138], and Gated Recurrent Units (GRU) [139], were inserted to improve CNN models. Finally, transformer networks are the most recent architectures introduced in speaker recognition [140]. Apart from the previous approaches using multi-class architectures, the triplet network paradigm has also been applied as a back-end for speaker recognition [141, 142].

As happened in face recognition, most of the initial deep learning systems were trained with the extended Cross-Entropy (CE) loss for multi-class classification. However, during the last few years, several loss functions have been designed to replace CE loss, especially in face recognition, and then applied in speaker recognition. Advances in this line of research have focused on two types: identification loss functions and verification loss functions. The former is mainly composed of CE loss combined with a complementary loss such as Ring loss (RL) [143] or Center loss [144], and other studies have focused on its variants such as Angular Softmax loss (A-Softmax) [145], or Additive Angular Margin Softmax loss (AAMSoftmax) [143]. While the latter is based on metric learning approaches as evolution of the triplet neural network such as contrastive loss [87, 137], partial AUC loss (pAUC) [146–148], NeuralPLDA [149] or angular prototypical [150].

Text-Dependent Speaker Verification

Due to the late development of text-dependent systems and the creation of a new database in 2012, the state-of-the-art of these systems during these years has been slightly different. The use of traditional techniques based on FA and deep learning approaches has overlapped. Traditional techniques with specific modifications have achieved good performance for this task. In [151], a modified version of JFA was proposed as Tied Mixture of Factor Analysis (TMFA), which uses the speaker and channel latent factor mechanism but, these factors were estimated in a different way. Another variant of JFA was presented in [152, 153] combined with PLDA scoring back-end. On the other hand, a set of HMMs was introduced to substitute GMM for training the i-vector extractor, and a PLDA was used to verify speaker identity [154–156]. Apart from using PLDA as a back-end, other works combined the i-vector extractor with SVM [157] or similarity metric [158].

Parallel to the previous research stream, deep learning approaches were applied to replace some parts of the traditional pipeline. Firstly, DNN was trained for phone classification, and the output of a narrow bottleneck layer was employed as input features for JFA or i-vector frameworks [159, 160]. In another approach, DNN was also trained for speech recognition, and the output was interpreted as posterior probabilities, which replaced the conventional GMM-UBM technique for JFA or i-vector extractors [159, 161]. Following the idea presented in [151], speaker and session variability were modeled by tied hidden variable with autoencoder DNNs [162]. In [151, 162], the systems developed needed to store a model per user which were adapted from a universal background model, and the evaluation of the trial was a likelihood ratio. One of the drawbacks of these approaches were the need to store a large amount of data per user and the speed of evaluation of trials since likelihood expressions were dependent on the frame length. Moreover, ambitious approaches were developed for these systems but, the systems have produced mixed results. DNN was trained to classify speaker identities, and frame-level embeddings (d-vector) were extracted to compare with the d-vector of each utterance and make a verification decision [163, 164]. d-vectors have provided good results when the task involves a large amount of private data and a single phrase. Motivated by the improvement achieved with x-vectors in text-independent systems, several unsuccessful attempts were performed to introduce this approach in text-dependent systems. One of the reasons this type of approach has been successful in text-independent systems is the availability of large public databases. However, unlike text-independent tasks, those existing large databases for text-dependent tasks are not always publicly available. Therefore, in tasks with more than one phrase and smaller databases, the lack of data may lead to problems with the use of deep architectures due to overfitting. For this reason, initially, these techniques based on the embedding extracted directly from a DNN or a ResNet have been shown ineffective [165, 166]. Using recurrent neural networks such as LSTM, an improvement was demonstrated in the case of large databases [167, 168].

Throughout this dissertation, several approaches have been proposed to solve these problems encountered in the development of such systems. First, DNNs successfully employed in the face and text-independent speaker verification tasks have been analyzed. In this analysis, it has been observed that the issues encountered when using them in text-dependent speaker verification have not only been motivated by the size of the databases, but also by the reduction mechanisms that are usually part of the architectures. Therefore, different approaches have been proposed to replace this kind of mechanisms such as the alignment mechanism [1–3] or multi-head self-attention layers of the transformer encoder [5]. Moreover, to mitigate the effect of having small training databases, other architectures based on the teacher-student philosophy [4] have been introduced for the first time for this task.

On the other hand, the slow progress of this research field and the problems motivated by the lack of large public databases have meant that the stream of research based on new loss functions developed in previous tasks has not been widely applied in text-dependent speaker verification. Hence, most of the existent approaches were trained with the traditional CE loss. Although some new loss functions have been proposed for text-dependent

speaker verification systems, such as a loss function similar to Triplet loss [167], Tuple-based End-to-End (TE2E) loss function [164] and Generalized End-to-End (GE2E) loss function [168]. Nevertheless, these loss functions were initially employed in a private database and difficult to compare. Approaches based on triplet neural networks were also developed on public databases [169]. However, these approaches trained the network architecture with the triplet loss function, which is not an objective loss function oriented to the goal task. Thus, in this thesis, a new loss function based on Area Under the ROC Curve (AUC) has also been applied to train the triplet neural networks. Furthermore, as already introduced in the reviews of the previous sections, another line of research has been developed in the second part of this thesis. This line of research is based on the development of alternative loss functions to CE loss more suitable for the verification task. The proposed loss functions are inspired by the final performance metrics, such as Detection Cost Function (DCF) and Cost of Likelihood Ratio (CLLR). Moreover, the implementation of these loss functions allows keeping the efficiency of using the same multiclass style of training as systems with CE loss employ [5, 8, 10].

2.3 State-of-the-art Language Recognition

2.3.1 Early Beginnings

Initial attempts were proposed in language recognition later than in face and speaker recognition. The reason was that many approaches used in language recognition come from speaker recognition and, in this case, the first speaker recognition systems date back to the 1960s. Thus, the first reports in automatic language recognition were made between 1973 and 1980 by Texas Instruments [170–173]. These reports explored the idea that the frequency of occurrence of certain reference sounds or sequences of sounds differed by language. With this motivation, these works extracted the reference sounds with an automatic segmentation technique based on spectral change. To decide the spoken language in the new speech utterances, the likelihood of the reference sounds was calculated. Hence, in these early approaches, the similarity of different spectral references was exploited to perform language recognition.

2.3.2 The 1980s-2000s

During these decades, great advances were produced, as occurred in speaker recognition. [174] was one of the first works developed to discriminate between languages using statistical inference techniques. In this work, an automatic approach based on acoustic-phonetic segmentation was employed to build the language recognition system. In another approach [175], a pattern analysis technique based on a polynomial decision function was designed and applied to the acoustic feature vectors extracted from the speech signal by Linear Predictive Coding (LPC) analysis. Until then, the available speech signals came only from read speech but, the next research approaches also started to address

recorded speech from radio under noisy conditions and spoken speech. Assuming the premise that prosodic features extracted from pitch and energy contours vary from language to language, a quadratic classifier could be applied to these features for language recognition in [176]. Besides, this work also introduced another technique to exploit the occurrence frequency of sounds using the formant frequency values, and a Vector Quantization (VQ) distortion measure was applied to make the language decisions. In [177], the VQ technique was used to characterize each language in combination with the acoustic features derived from LPC.

Motivated by the success achieved in speaker recognition, Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM) were also incorporated into the language recognition. [101] proposed the use of HMM for prosodic features, while in [178], an interesting comparison was made using VQ, HMM and GMM acoustic classifiers with Mel-Frequency Cepstral Coefficients (MFCCs) as feature vector. In [179], one of the first approaches employing n-gram features was developed, and the use of Perceptual Linear Predictive (PLP) coefficients for language recognition systems was also introduced. This work employed a phonetic classifier to recognize the phonemes of each language and then extracted unigram and bigram statistics. To argue that pitch estimation was more robust in noisy environments than spectral features, [180] built a system that used only features based on pitch estimates. These features were analyzed using Principal Component Analysis (PCA) and discriminant analysis. Moreover, in 1996, when NIST began speaker recognition evaluations, the Language Recognition Evaluation (LRE) was also launched to compare the performance of the language recognition system under the same conditions for all the participants.

2.3.3 The 2000s-2012s

In the early 2000s, GMM-UBM, a widely deployed approach in speaker recognition, was introduced as an alternative to the standard GMM system [181]. As an input feature in this work, MFCCs and their first delta parameters were employed. In 2002, one of the most important works in the language recognition field was presented in [182]. This work introduced the use of cepstrum data combined with some special derivatives. These derivatives were Shift Delta Cepstrum (SDC), which were the derivatives of a downsample version of the sequence, so long-term information was considered since the derivative coefficients where applied to non adjacent frames. Besides, a Gaussian back-end classifier was employed for recognition. Continuing the previous work, [183] proposed a variant of Gaussian back-end and applied the Support Vector Machine (SVM) classifier [184] for the first time in language recognition.

To compensate for the channel mismatches that still existed, a research stream for language recognition based on the application of Factor Analysis (FA) approaches started to be applied. The work presented in [185] was considered the first to use FA techniques. FA was employed to create a factor subspace that represents the distortions caused by inter-speaker variability within the same language, helping to compensate for these dis-

tortions. In [186], an eigenlanguage space was developed by analogy with the eigenface and eigenvoice approach. This approach was proposed to estimate language factors with PCA, which had low dimensionality. These factors were used as input features for an SVM classifier. Following with FA approaches, in [187], JFA was applied to model language and channel variabilities separately, and accurate acoustic language recognizers were built. One of the most popular approaches of the last decade in speaker recognition was also applied to the language recognition task. This approach was based on i-vector utterance representation [188,189]. In this approach, the system pipeline consisted of two main blocks: a front-end formed by a GMM-UBM along with a total variability subspace for i-vector extraction and a back-end. The i-vector was considered a low-dimensional representation of the GMM mean-supervector obtained by FA. After these two works, i-vectors became the most widespread approach, and researchers' efforts focused on the study of back-ends, which can be one of several existing generative and discriminative classification techniques to produce system scores. In the following years, several attempts to improve the back-end technique employed as an i-vector classifier led to the development of studies using all existing classifiers such as Logistic Regression (LR) [188], SVM [189], or variants of previous employed GB [190, 191]. However, Probabilistic Linear Discriminant Analysis (PLDA), which was the main back-end technique developed for face recognition and successfully applied in speaker recognition, was not as successful in language recognition [192].

2.3.4 The 2012s and onwards

Due to similarities in the research fields, recent advances in automatic speech recognition and speaker recognition techniques based on discriminative Deep Neural Network (DNN) have subsequently improved the technology applied to language recognition. At the beginning of this deep learning era, modern systems have substituted parts of the previous pipeline with DNNs to achieve greater robustness to varying conditions and audio duration. For instance, a significant step forward in language recognition was obtained by replacing acoustic features with DNN bottleneck representations [193]. These representations were used as input to the i-vector system. Another approach was proposed to replace GMM-UBM posteriors with DNN or Convolutional Neural Network (CNN) senone posteriors [191, 194, 195] within the i-vector framework.

At the same time, other more ambitious DNN approaches have been used, including end-to-end systems [196–198]. These techniques consisted of a DNN trained with a softmax layer at the output and a multiclass Cross-Entropy (CE) loss, whereby the network learns to classify languages. These end-to-end DNN systems are quite successful at this task, but the approach requires relatively large training datasets compared to the i-vector framework, and it is computationally more expensive. More recently, leveraging advances in speaker recognition, the use of DNN language embeddings to produce an utterance-level representation of speech has become commonplace [199]. The most employed embeddings were x-vectors and replaced i-vectors as front-end [200–202]. In this approach, once the DNN is trained, embedding representations are obtained from

one of the previous layers within the DNN, and these embeddings are evaluated using a back-end classifier like any traditional classifier or neural network (NN) [203].

As in deep learning approaches to face and speaker recognition systems, a new line of research has been initiated in recent years motivated by the limitations presented by CE loss. However, for the language recognition task, only the most successful loss functions from the speaker verification task have been applied. As for example, [144] introduced the combination of CE loss with Center loss. On the other hand, variants of CE loss have also been incorporated into the DNNs pipeline for language recognition, such as Angular softmax [144, 204] or Additive Angular Margin softmax [205]. Another widespread approach in face and speaker recognition tasks has been metric learning loss functions, especially the triplet neural networks. Nevertheless, this approach had not been employed for language recognition. For this reason, in this dissertation, we propose the use of this kind of network for language recognition. In addition, the AUC-based loss function applied for face and speaker verification has also been introduced for this task [7].

3

Biometric Recognition Systems

3.1 Introduction

As Chapter 1 introduced, biometric recognition systems are attracting a great deal of research interest since they have become a crucial part of day-to-day life. There are several tasks that can be performed by these systems, but in this thesis, we focus on the verification task to recognise faces, speakers and languages from image and audio samples. As described in Chapter 2, many approaches have been created to develop such systems to address this task using the different characteristics. The pipeline of these biometric recognition systems is very similar regardless of the biometric characteristic employed and consists of the parts shown in Figure 3.1. These systems typically have four different phases: training phase, development phase, enrollment phase and test phase. During the training phase, the implementation of a generic model is usually carried out. In this phase, the representation method is chosen and trained using the training data from the different databases including a large amount of different identities. Moreover, in the case of applying a trainable back-end, this step is also performed. The second stage is the development phase which is not always used. This phase allows obtaining the scores of the representation method or the back-end. After that, these scores could be incorporated into the last phase to perform score normalization and calibration. During the enrollment phase, the examples of the target identity are processed with the representation method and, if applicable, with the back-end. Once these examples are processed, in the test phase, they are compared with the test data, which are new examples also known as trial examples. From this comparison, the scores are obtained, and with these scores which can be corrected by the normalization or calibration methods, a final decision about the identity is made.

Thus, in this chapter, the main blocks that compose this type of system and also the most relevant approaches of the state-of-the-art for each block are presented.

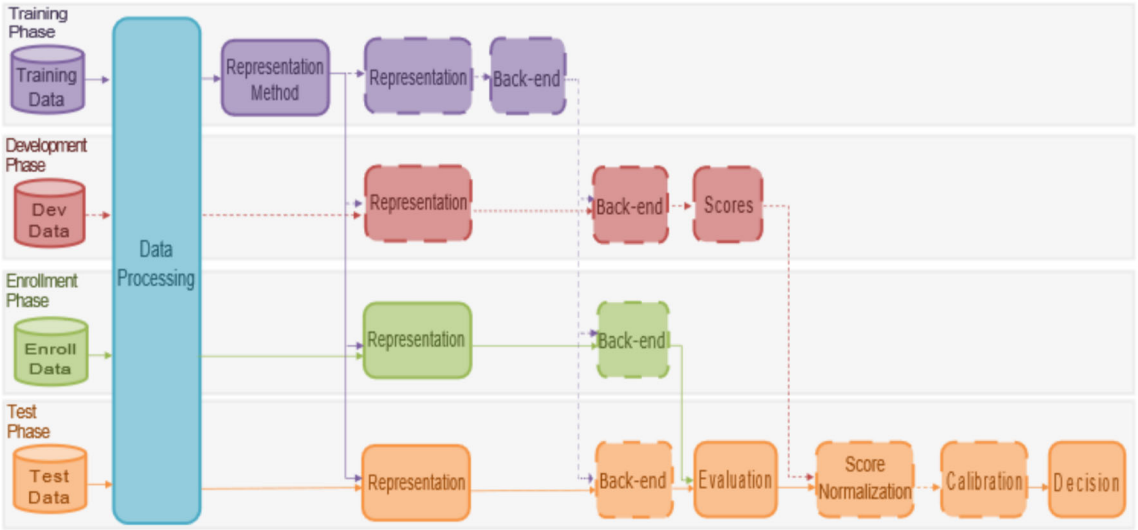


Figure 3.1: Components and phases of a biometric recognition system, the dashed lines indicate the blocks that are optional in this kind of system.

3.2 Data Processing

The first step of biometric recognition systems concerns acquiring the data and process them to obtain the most relevant information. Nowadays, acquiring image and audio samples is easier than capturing other biometric characteristics. It is made possible by the widespread use of technological devices to capture image and audio data. After that, the data is prepared to facilitate the training of the recognition methods. Depending on the biometric trait employed, this data processing is slightly different. Thus, throughout this section, we explain the specific process applied in function of the type of data available as Figure 3.2 depicts. First, when the data come from image or audio directly, and in the other case, whether the available data are video files.

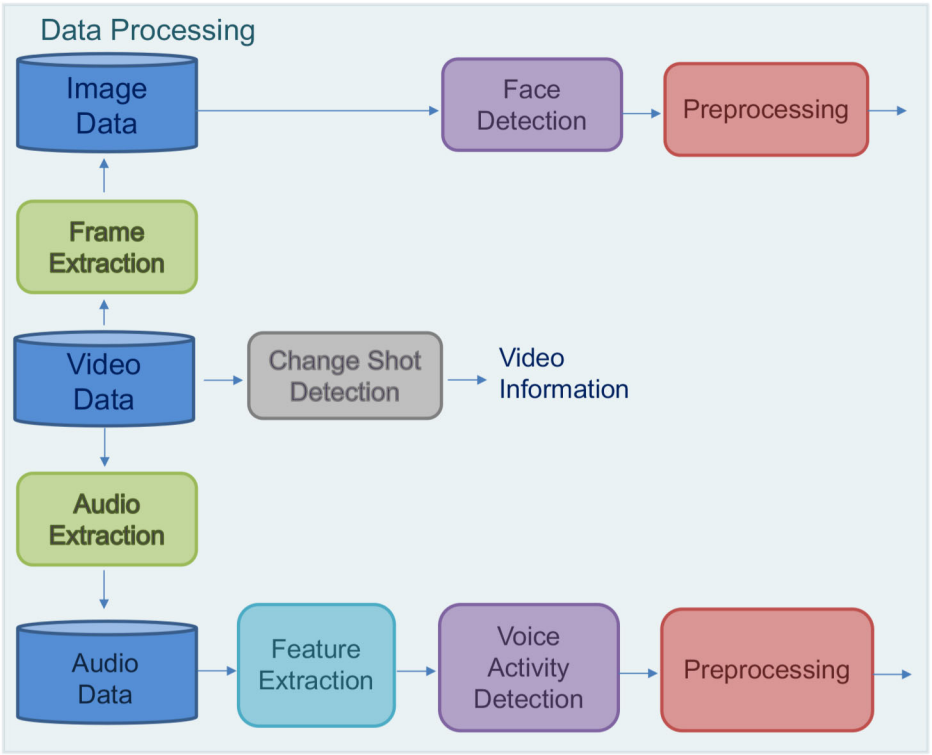


Figure 3.2: Data processing in function of the kind of data available.

3.2.1 Face Processing

This section explains the main steps that are followed to prepare face images when a face recognition system is to be developed. As described in Chapter 2, a large number of different features have been obtained over the years to represent a face image. These features have been classified into two lines of research: local or texture-based features and holistic or appearance-based features. The former is based on treating only some critical facial points and extracting local features to generate more details. In this type of features, the most employed approaches have been Elastic Bunch Graph Matching (EBGM), Scale Invariant Feature Transform (SIFT), and Local Binary Patterns (LBP). While the

latter features process the whole face and try to represent it in a low-dimensional space within this kind of features are Eigenfaces, Fisherfaces, Gabor filters, and Discrete Cosine Transform (DCT) features. Nonetheless, nowadays, none of these feature extraction techniques are commonly used when Deep Neural Networks (DNNs) are involved to carry out the recognition process. Since these deep learning based methods are considered a holistic approach that performs deep feature extraction and deep face recognition steps.

During the development of this dissertation, we have employed different architectures based on DNNs, as we will explain in the following sections. As input for each of these architectures, we have not applied any of the previous feature extraction techniques present in the literature. Instead, raw face images have been used to train the different architectures. Although we do not apply a feature extraction technique, several steps are involved in the process before training the DNNs that simplify the raw face images to facilitate the following process. The first of these steps is face detection which has to be done before continuing with the rest of the face verification pipeline. Face detection is a fundamental step because failures in this process could be crucial for the correct development in other parts of the face recognition system. To carry out this step, the face detector employed is a system of alignment and detection. After the detection step, a face preprocessing step is usually applied. This step can be different depending on the architecture used. The most common operations employed in this step are cropping and normalization, which help to make all the face images used to train the DNNs uniform. Finally, a data augmentation process is also applied to increase the amount of image data by generating slightly modified copies of the existing data [71, 206]. Moreover, the use of this process helps to reduce overfitting due to the lack of data in some cases and to generate more robust systems.

3.2.2 Audio Processing

In the case of working with audio files, the process followed is conceptually quite similar to that of face processing. Nevertheless, for developing the speaker and language recognition systems, a feature extraction step is usually applied. As already mentioned in Chapter 2, many types of features have been applied for decades to extract relevant information about speech samples and remove irrelevant and redundant information. Techniques to obtain these representations have been based on different groups of feature approaches: short-term spectral (Mel-Frequency Cepstral Coefficients (MFCC), Linear Predictive Coding (LPC)), prosodic and high-level features. In recent years, apart from approaches using these features, several works have appeared that use raw speech files or the short-time frequency transform (STFT) directly to train DNNs [139]. However, in this thesis, the most popular features are employed which are the short-term spectral features. Moreover, there are several variants of short-term spectral features which characterize the resonances of the vocal tract. The most widespread spectral features are MFCCs [96], but more recently, log Filter Banks (FBs) are becoming increasingly popular as intermediate signal representations between raw speech files and MFCCs to train

DNNs. For this reason, in the most advanced models of this thesis, FBs are also used as input features.

To extract the features, the process followed consists of performing a short-time analysis of the speech signals. The pipeline to compute these spectral features is depicted in Figure 3.3 which shows that the process to obtain both types of spectral features (MFCCs, FBs) involves somewhat the same procedure. This procedure consists of several stages. Initially, offset compensation, pre-emphasis filter and multiplication with hamming windows are applied on the frames of the speech signal to condition this signal for subsequent spectral analysis. Afterwards, the spectrum is calculated with the Fast Fourier Transform (FFT). Then, a set of triangular Mel-filters is applied on the spectrum to obtain the frequency bands. After that, the output spectrum is converted to a logarithmic scale, and the FB features are produced as the result of this conversion. Finally, if a Discrete Cosine Transform (DCT) is applied, the MFCC features are obtained. Once the features have been calculated, the log energy is concatenated to them, and, in the case of the MFCC, the derivatives are also usually computed and added.

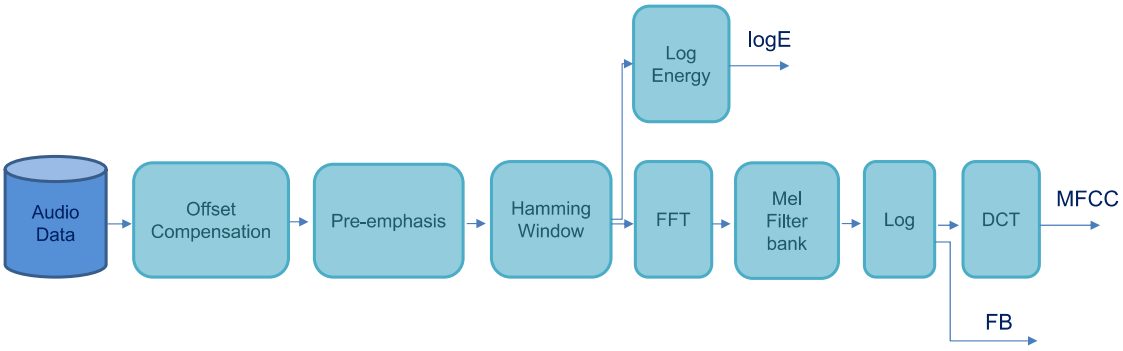


Figure 3.3: Audio feature extraction pipeline.

As shown in Figure 3.2, once the features have been extracted, a Voice Activity Detection (VAD) step is applied. This step is critical to the recognition task, as it detects speech in the audio samples and removes non-speech frames for optimal performance. There are different approaches to obtain this VAD, where the simplest one is an energy-based voice activity detector. This detector consists of setting a threshold, and when the energy of the signal is above the threshold, the VAD indicates speech activity. Other more recent VAD approaches have been proposed that rely on deep learning solutions [207]. Both approaches can be employed using the different types of features presented previously. After the VAD step, preprocessing is applied as in face processing. Including in this preprocessing, normalization and scaling techniques are usually applied. Apart from these traditional techniques, data augmentation techniques are also incorporated before training using DNNs.

3.2.3 Video Processing

Aside from working directly with image and audio samples, data can be derived from video files. In this case, as we observe in Figure 3.2, two first steps are performed to extract the image and audio. Once the images are extracted in the frame extraction step, the process applied on them is the same as directly using a dataset of face images. On the other hand, when audio extraction has been performed, the audio processing is also the same.

In addition to the above steps, whether video files are used and depending on the task performed, a shot change detection step is applied. This step allows generating extra information from the video files, such as the changes of shots and scenes. A simple and efficient approach to obtaining these changes consists of using a threshold-based detection method. This detector finds areas where the difference between two subsequent frames exceeds a threshold value. In the tasks of tracking people through the video such as Multimodal Diarization, this information helps to perform the tracking and clustering step since usually the source of the videos employed are television programs. These programs are composed of a huge variability in content characteristics and constant changes of shot and scenes.

3.3 Representation Methods Review

As Figure 3.1 depicts, once the data have been processed, a representation method is introduced to obtain a representation of each sample from the input data. A large number of approaches have been developed to address this task, as we presented in Chapter 2. There are two main types of representation methods: model-based and embedding-based approaches. The former are methods based on the creation of generative models, which will be able to represent all data, such as Hidden Markov Models (HMMs) or Gaussian Mixture Models (GMMs). While the latter are techniques that train systems which allow extracting an internal representation such as GMM Supervectors, i-vectors or embeddings of a neural network. The most widespread methods used as a starting point in this thesis are explained below.

3.3.1 Hidden Markov Model

The Hidden Markov Model (HMM) [48] is a generative probabilistic model that represents a set of stochastic sequences (such as face regions, audio concepts or segments) as a Markov chain with a finite number of states, Q. Each state generates observations following a random process which we characterize by its Probability Density Function (PDF). The elements that characterize a HMM are:

- A: the state transition probability matrix where a_{ij} represents the probability of moving from state q_i to state q_j.
- B: for discrete observation variables, the emission probability matrix defined by the probability density function associated with each state where b_j denotes the probability that the observation is generated from state q_j.
- Π: the vector with the initial state probabilities of each state π_i.

HMMs could be classified according to the type of output distribution model into two categories. The first is discrete HMMs, where the observations are discrete elements. On the other hand, in continuous HMMs, the states are represented by continuous observation density functions. The most common way to model observation probabilities is using a mixture of Gaussian distributions:

$$b_q(o) = \sum_{c=1}^{C} w_{qc} \cdot N(o|\mu_{qc}, \Sigma_{qc}) \tag{3.1}$$

where w_{qc} is the mixture weight for the cth mixture in state q, and $N(o|\mu_{qc}, \Sigma_{qc})$ is a multivariate Gaussian distribution with mean μ_{qc} and covariance matrix Σ_{qc}.

To create this representation model, initially, the emission and transition probabilities are chosen randomly and the initial state probabilities are uniformly distributed. Then, this kind of model is trained using Expected Maximization (EM) or Baum-Welch algorithm to compute and update iteratively the elements characterizing HMM. This process has been explained to train a single HMM. However, it can be repeated to obtain a model for each class. Afterwards, using the estimated model parameters and given an observation sequence, the likelihood of the observation sequence is calculated with the forward-backward algorithm. Moreover, the most probable state sequence associated with the given observation sequence can be found with the Viterbi algorithm.

3.3.2 Gaussian Mixture Model

Like the previous HMM, Gaussian Mixture Model (GMM) [106] is also a generative model. In concrete, HMM can be viewed as the general model, which includes GMM as a specific HMM with a single-state with a mixture of Gaussian distributions. This kind of model has been the most widespread since it is a more robust parametric model due to the fact that it ignores the temporal dependence, so the transition probability matrix is not necessary. Furthermore, previous works have demonstrated that recognition rates were strongly correlated with the number of mixture components and not so much with the number of states. Therefore, GMM is defined as a parametric probability density function which is composed of a weighted sum of individual Gaussians without the dependence on any state since there is only one:

$$p(o|\theta) = \sum_{c=1}^{C} w_c \cdot N(o|\mu_c, \Sigma_c) \tag{3.2}$$

where w_c is the mixture weight for the cth mixture, and $N(o|\mu_c, \Sigma_c)$ is a multivariate Gaussian distribution with mean μ_c and covariance matrix Σ_c which can be expressed as:

$$N(o|\mu_c, \Sigma_c) = \frac{1}{(2\pi)^{D/2}|\Sigma_c|^{1/2}} \cdot \exp\left(-\frac{1}{2} \cdot (o - \mu_c)^T \cdot (\Sigma_c)^{-1}(o - \mu_c)\right) \tag{3.3}$$

To estimate the GMM parameters, the process followed is simpler as for the training of HMMs, since there is not necessary to estimate the alignment, and the Maximum Likelihood (ML) criterion is iteratively optimized by using the EM algorithm. Although for maximization, the logarithm of the likelihood is usually employed instead of likelihood. The use of these models for some tasks quickly became the dominant approach. Nevertheless, as the number of classes increased, building a class-specific GMM with a large number of mixture components is not a good choice, as it requires too much time to create all class-specific models and a large memory storage to save them. To solve this issue, the GMM combined with the Universal Background Model (UBM) approach was developed. Using the GMM-UBM approach, a general UBM model is trained. Afterwards, to train the enrol models the amount of data is low, therefore the UBM parameters are adapted to be robust against data scarcity following the Maximum a Posteriori (MAP) approach. MAP adaptation is defined as:

$$\begin{aligned} \mu_c^{MAP} &= \frac{\sum_{n_c=1}^{N} x_{n_c} + \tau \cdot \mu_c}{n_c + \tau} \\ &= \frac{n_c}{n_c + \tau} \frac{\sum_{n_c=1}^{N} x_{n_c}}{n_c} + \frac{\tau}{n_c + \tau} \cdot \mu_c, \end{aligned} \tag{3.4}$$

where τ is a fixed relevance factor, and n_c are the weights of Gaussian components. Hence, the MAP adapted mean parameter is computed for each class n as:

$$\mu_c^n = \rho \cdot f + (1 - \rho) \cdot \mu_c, \tag{3.5}$$

where ρ is the adaptation coefficient defined as $n_c/n_c + \tau$, and f is the mean estimation. With this adaptation, a class-dependent GMM is obtained.

3.3.3 From Representation Models to Vectors

The methods presented so far are model-based techniques to obtain models that represent the data. However, since the early 2000s, the embedding paradigm has gained prominence. This paradigm consists of creating models that constrain the class information into a restricted space where a compact vector represents each class. Regarding this paradigm, initially, the GMM supervector derived from the adapted GMM-UBM emerged. This supervector is a high-dimensional representation of a sample file that is obtained by concatenating the mixture means of the class adapted GMM-UBM to form a GMM mean supervector.

$$s = [\ \mu_1^n,\ \mu_2^n,\ ...,\ \mu_C^n\ ], \tag{3.6}$$

where μ_c^n represents the mean vector of cth Gaussian in the GMM for each class n.

After that, another relevant approach based on the representation vector instead of models is the i-vector approach. The i-vectors apply the Factor Analysis (FA) techniques and employ a low-rank projection matrix known as Total Variability Matrix (TVM) to reduce the dimension of GMM supervectors. Nonetheless, in this dissertation, the i-vector approach has not been applied.

3.3.4 Convolutional Neural Network

Within the embedding-based methods, the latest approach to emerge was the deep learning representation methods. In these methods, typically, a deep learning system is trained, and an internal representation is extracted for each sample file. This representation is what is known as embedding. Several variants of deep learning representation techniques have been the main approach employed in this dissertation.

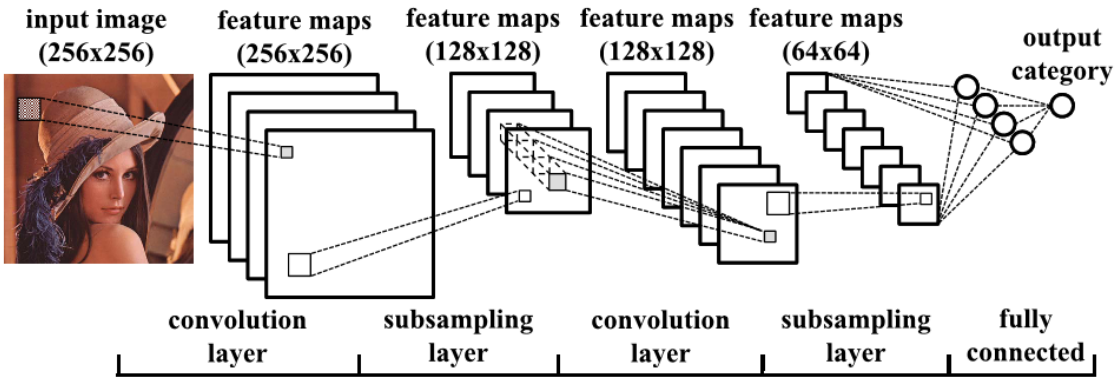


Figure 3.4: CNN architecture example.

Deep learning methods are based on Artificial Neural Network (ANN). ANNs are non-linear models inspired by the structure and information processing of the brain. These networks consist of three types of layers: input layer, one or more hidden layers, and output layer. When these networks have many hidden layers, these models are called Deep Neural Networks (DNNs). DNNs can capture complex relationships between input and output. Within DNNs, we can find numerous types of neural networks. One of the most popular architectures is Convolutional Neural Network (CNN) [208, 209]. Traditionally, standard CNNs have as input a 2D matrix of values, although other implementations exist to work with 1D or 3D matrices depending the data used. From the structural point of view, the typical CNN is composed of convolutional layers, pooling layers, fully connected layers and a loss layer, as can be seen in Figure 3.4. During this thesis, we have worked on the different parts of these CNNs. Therefore, we will now provide a general introduction of them, and throughout the different chapters, we will explain the specific modifications or developments made in each part.

Convolutional Layer

The convolutional layer is the core building block of these networks. With these layers, the main idea lies in applying a convolution operation in place of a general matrix multiplication between the input signal and a filter or kernel to generate a feature map. This layer is not connected to all input values because this leads to impracticable computational complexity when the input to the CNN is high-dimensional data. Instead, neurons are connected only to a small local region of the input, which is the local receptive field. Then, the filter is applied as a sliding window across the entire input signal, and each part corresponds to a different neuron in the layer. Using this kind of layer with two-dimensional data, the filter slides over the width and height of the data, as the example in Figure 3.5 depicts, and spatial and temporal dependencies are captured. Hence, the main goal of this layer is to extract relevant features of the data. This data is usually derived from real-world data, so the distribution of the data is too complex to model it linearly and simple linear approaches are ineffective. However, the convolutional layer is a linear operation. Therefore, an additional activation function is usually included to introduce nonlinearity after each convolutional layer. The activation function is an element-wise operation such as Sigmoid, Hyperbolic Tangent (Tanh) or Rectified Linear Unit (ReLU). Moreover, CNNs are usually composed of several convolutional layers and the composition of these type of linear (convolution) and nonlinear operations gives the network great modelling capacity for complex data. Hence, depending on the point of the architecture, the features are different in each layer. In the initial layers, low-level features are extracted, such as edges, color, etc. While in the advanced layers, high-level and more abstract features are obtained, such as shapes, faces, etc.

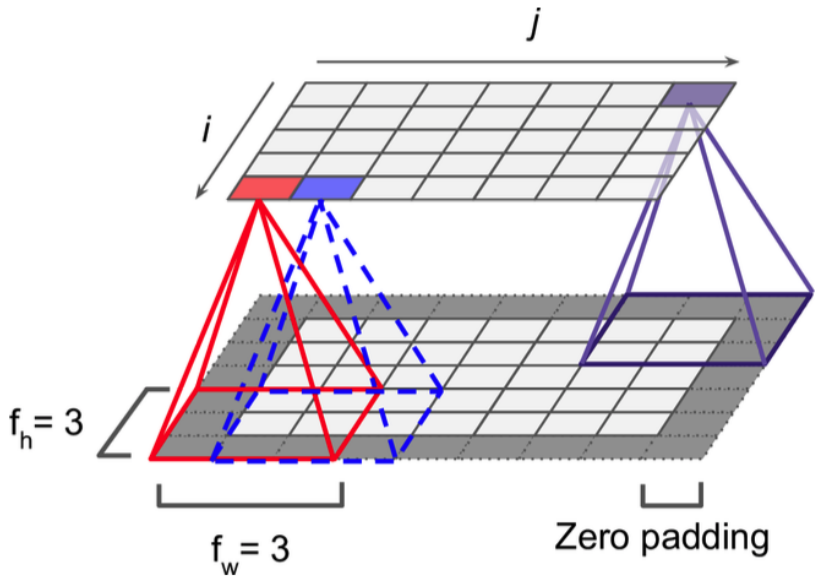


Figure 3.5: Convolutional layer example [210]. In this example, the filter of size 3x3 is applied to the input data to capture temporal and spatial dependencies, and the relevant features of data of each region are extracted as output.

Pooling Layer

Typically, after the convolutional layer, a second layer called pooling or subsampling layer is employed. The pooling layer aims to reduce the processing time by summarizing the statistics of nearby regions and preserving the most important information in them. For this purpose, this layer performs a subsampling of the input data in order to reduce the dimensionality of each feature map coming from the previous convolutional layer. This reduction allows making the input representations smaller and more manageable, which also involves a smaller number of parameters and computations in the subsequent layers of the network. Likewise, the risk of overfitting is reduced. Furthermore, this layer also introduces robustness against a certain level of invariance to small transformations, translations and distortions in the input data, so the dominant features are extracted. The most common reduction methods are average and max pooling. An example of how this max pooling method works is shown in Figure 3.6. Using this method, the maximum value on each feature map is obtained and used as output of this layer. Moreover, this selection of the maximum values of each region reduces the size of the output representation.

These pooling layers interleaved with the convolution layers is a widely used approach in image recognition tasks, while depending on the task performed, it could be necessary to keep one of the dimensions invariant. For this reason, in some implementations, the pooling layer is only applied once all convolution layers have been employed and before the use of the fully connected layer, usually in this case, it is called global pooling if the size of the receptive field kernel involves the entire signal.

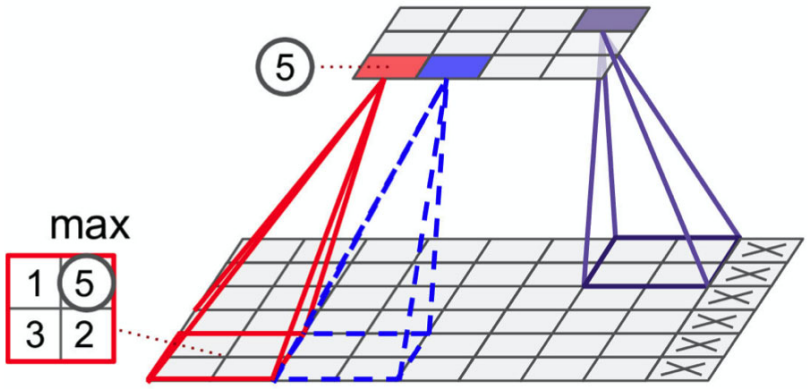


Figure 3.6: Pooling layer example using maximum method [210]. This type of layer operates on each feature map to obtain the maximum value, so that the dominant features are kept and the size of the representation in the output is reduced.

Fully Connected Layer

After several convolution layers, activations and pooling layers, a reduction or flatten is applied to convert the matrix of feature maps into vectors without losing information. Once the reduction or flatten is performed, a fully connected or dense layer is used to connect all the neurons in the previous layer to each of the neurons in the next layer. Thus, this layer is a linear combination of all the inputs using a weight matrix and an optional bias parameter where the output is expressed as:

$$o = W^T \cdot x + bias, \tag{3.7}$$

where W^T is the weight matrix of the linear layer and $bias$ is the bias parameter vector. The purpose of this layer consists of learning the non-linear combination of the high-level features from the previous spatial layers and uses this combination of features for the final classification.

Loss Layer

Finally, after performing representation learning in many layers, CNN systems have a loss layer or objective function. This loss layer determines the function to be optimized with that network. The most traditional approach has been based on employing a softmax activation function after the last fully connected layer, and then, Cross-Entropy (CE) loss is obtained. CE loss is calculated as:

$$L_{CE} = -\sum_{i}^{m}\sum_{j}^{N} y_{ij} \cdot \log(\hat{y_{ij}}), \tag{3.8}$$

where y_{ij} is the ground truth class label with $i \in \{1, ..., m\}$ where m is the number of samples and $j \in \{1, ..., N\}$ where N is the number of classes, and $\hat{y_{ij}}$ is the predicted probability extracted from the output of the last fully connected layer. Nevertheless, this approach, although it is optimal for classification, has several problems in obtaining more general representations for biometric comparison. As we introduced in Chapter 2, a very active research line is currently being widely investigated to solve these issues. For this reason, in the third part of this thesis, an extensive study related to the loss functions of neural network architectures for biometric comparisons has been developed.

3.3.5 Residual Neural Network

Since in 2012, CNNs began to gain more attention, a lot of more powerful variants based on CNNs have been proposed, such as AlexNet [71], VGGNet [211], GoogLeNet Inception [76], ResNet [79], UNet [212] or EfficientNet [213]. These approaches are mainly focused on introducing more and more CNN layers into the architectures. Nevertheless, [79] studied the effect of degradation that these deeper architectures start to show.

Initially, this degradation was the result of overfitting, but a wide analysis showed that it was not the cause of performance degradation. Instead, the optimization function, the initialization of the network, or the problem of vanishing or exploding gradients could be blamed.

To solve these issues, in [79], a new architecture called Residual Neural Network (ResNet) was also created and achieved high relevance. ResNets are based on convolutional layers and residual path additions, which are also known as skip connections or shortcut connections. The basic block diagram of the ResNet architecture with the skip connection is depicted in Figure 3.7. This connection allows the input signal of the residual block to be added directly to the output. In this way, the effect of vanishing gradients is reduced, and the training convergence improves. Moreover, the use of these blocks increased the ability to train much deeper networks than was previously possible.

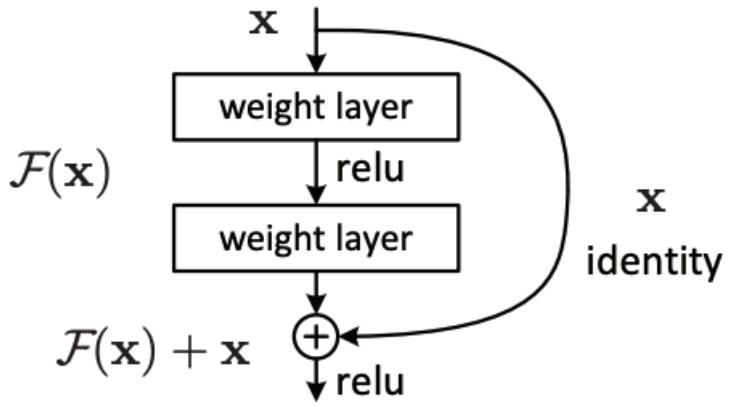


Figure 3.7: Residual block [79]. This block has a skip connection that allows information to flow more easily from one layer to the next layer which helps to reduce the effect of vanishing gradients.

3.4 Back-end Approaches Review

For the back-end and evaluation blocks in Figure 3.1, many popular algorithms have been employed throughout history. As with representation methods, these algorithms can be categorized into two main groups: distance-based and trainable-based. Distance-based approaches, such as Euclidean distance or cosine similarity, are directly applied to obtain the scores of the representations extracted from the representation method. On the other hand, the trainable methods consist of different techniques in which a model is trained using generative approaches such as Gaussian back-end and PLDA, or with discriminative approaches such as SVM and NN back-end. The biometric recognition systems may or may not introduce one of these trainable methods, and only these systems are evaluated using a distance-based approach. As in the previous section, the approaches employed in this thesis are briefly described below.

3.4.1 Cosine Similarity

The easiest approach to employ as a back-end is cosine similarity. This similarity is a metric that measures the normalized product between two representation vectors. These vectors are known as enrollment and test vectors in recognition systems, and when these vectors come from a DNN are often called embeddings. To determine the similarity, the cosine of the angle between the two vectors is calculated. In order to calculate the cosine value, no extra training is required. Using this metric, the degree of similarity of the enrollment and test vectors is computed as:

$$s(e_e, e_t) = \frac{e_t^T \cdot e_e}{\|e_t^T\| \cdot \|e_e\|}, \tag{3.9}$$

where $\|e_e\|$ is the norm of the enrollment vector, and $\|e_t^T\|$ is the norm of the test vector, where the euclidean norm applied to these vectors is $\|e\| = \sqrt{\sum_d e_d^2}$. Whether this similarity value is close to 1, it indicates that the two vectors are very similar. Whereas two orthogonal vectors would have a similarity close to 0.

3.4.2 Weighted Gaussian Back-end

The previously explained similarity is a straightforward and an efficient way to evaluate the systems. However, depending on the task and the approach used to obtain the representation vectors, a trainable back-end can also be incorporated to improve the results. The generative Gaussian Back-end (GB) is one of the most common. In this back-end, each class is modeled by a Gaussian distribution, defined by a class-dependent mean and a full covariance matrix shared across all classes. These Gaussian models are used to compute the likelihood of a sample for each of the modeled classes. As usually, the back-end training data is likely to be imbalanced, a modification of the Gaussian Back-end [191] was presented to normalize for the imbalance by appropriately weighting the examples during the computation of the means m_l and covariance S of the model. The Weighted GB (WGB) is thus defined by,

$$m_l = \frac{\sum_{i|l_i=l} w_i \cdot e_i}{\sum_{i|l_i=l} w_i} \tag{3.10}$$

$$S = \frac{\sum_l \sum_{i|l_i=l} w_i \cdot (e_i - m_l)^T \cdot (e_i - m_l)}{\sum_i w_i}, \tag{3.11}$$

where e_i is the representation vector with $i \in \{1, ..., m\}$ and m is the total number of examples, w_i is the weight assigned to each example, and l_i is the class present in the example. The weights are computed such that all samples from a class are weighted equally ($w_i = w_j$ if $l_i = l_j$) and the sum of their weights is identical across classes ($\sum_{i|l_i=l} w_i$ is the same for all l).

3.4.3 Probabilistic Linear Discriminant Analysis

Another generative approach has been widely used in several tasks, which is called Probabilistic Linear Discriminant Analysis (PLDA). PLDA is a probabilistic variant of Linear Discriminant Analysis (LDA) that can be used with more complex data than LDA. This technique is also similar to models based on Joint Factor Analysis (JFA). The PLDA modelling can be expressed as

$$e_{ij} = \mu + V \cdot y_i + U \cdot x_{ij} + \epsilon_{ij}, \tag{3.12}$$

where μ is the class-independent mean vector, V is the eigenclass matrix, y_i is the class factor, U is the eigenchannel matrix, x_{ij} is the channel factor with $j \in \{1, ..., m\}$ and m is the total number of examples, and ϵ_{ij} models the residual variability (intra class).

Once this model is trained, it can be used as a binary detector to determine whether pairs of examples are of the same class or different. For scoring pairs of examples, the verification score is calculated using the likelihood ratio of the form:

$$s(e_e, e_t) = \frac{p(e_e, e_t|H_1)}{p(e_e|H_0) \cdot p(e_t|H_0)}, \tag{3.13}$$

where e_e is the vector of enrollment sample, e_t is the vector of test sample, H_1 represents the hypothesis of both vectors coming from the same class, and H_0 is the hypothesis of both vectors belonging to different classes.

3.4.4 Support Vector Machines

The generative back-end techniques aforementioned may struggle to separate similar classes because they are not trained to separate them explicitly. For this reason, in [184], the Support Vector Machines (SVM) kernel-based classifier were introduced as discriminative technique. The SVM classifier focuses on finding the optimal plane which maximizes the margin between classes. This classifier is built from the sum of a kernel function as

$$f(e) = \sum_i w_i \cdot y_i \cdot K(e, e_i) + b, \tag{3.14}$$

where e_i is the representation vector with $i \in \{1, ..., m\}$ and m is the total number of examples, w_i are the learned coefficients, y_i are the target values, and $K(e, e_i)$ is the kernel function used to discriminate classes. One of the most employed the kernel function is the radial basis function (RBF).

The SVM is by nature a binary classifier, so to use the SVM with multiclass data, a “one vs. all" strategy can be employed. This strategy consists of training class-dependent SVMs that target one class against the pool of all other classes. In this manner, the margin between the target class and the other classes will be largely defined by closely related classes to the target class.

3.4.5 Neural Network Back-end

Apart from the techniques introduced above, a discriminative model based on a feed-forward Neural Network (NN) approach is another option for training a back-end. Recently, the NN approach has been widely extended as a representation method to obtain the vectors characterizing each sample. Nevertheless, several traditional techniques are still used for some tasks to extract representation vectors, so in those cases, the NN can be employed as a back-end over the representation vectors extracted from the previous model. Even sometimes, special NN configurations [75, 149] are applied as back-end with vectors extracted from another DNNs, as we will present and analyze in later chapters.

3.5 Score Normalization

Before the final verification step, a score normalization is usually applied to conclude the system. The motivation for this score normalization is the large variations that can be observed in the scores obtained with different back-end techniques. Therefore, the need arose to compensate for them to reduce variability, and several normalization techniques have been developed. These techniques allow to improve the quality of the final scores, but depending on the previous models, they could be optional to apply in the cases where this variability is not very large. Among all normalization techniques, Zero Normalization (Z-Norm) and Test Normalization (T-Norm) [214] have been the most popular until the introduction of the PLDA paradigm. PLDA produces symmetric scores, so another normalization technique was extended, which is Symmetric Normalization (S-Norm) [215]. Nowadays, this S-Norm is one of the most common score normalization applied and can be calculated as:

$$score\_norm = \frac{score - \mu_e}{\sigma_e} + \frac{score - \mu_t}{\sigma_t}, \tag{3.15}$$

where $score$ is the original score, μ_t and σ_t are the mean and standard deviation of the scores obtained from the evaluation of test vs. development files, and μ_e and σ_e are the mean and standard deviation of the scores obtained from the evaluation of enroll vs. development files.

3.6 Calibration

As the score normalization step, this calibration step is not always applied. However, when it is introduced into the system, it is the last step before the final decision process. In this step, the scores generated by the back-end are calibrated by transforming the scores into proper likelihoods using a linear logistic regression as described in [216, 217],

$$\log(LR) = \beta_0 + \beta_1 \cdot scores. \tag{3.16}$$

Using this transformation, these scores are converted into well-calibrated Log-Likelihood Ratios (LLRs). Apart from obtaining well-calibrated LLRs, this process also chooses the optimal threshold for detection. Thus, this step ensures optimal performance for the operating point of the application.

3.7 Decision Making

Once the scores comparing the enrollment and test samples have been obtained, and these scores have been normalized and calibrated whether both steps are employed, the last step to complete the verification process consists of making a decision. In verification systems, this final decision is taken by comparing the scores to an empirical threshold (Ω) and is a binary decision: acceptance or rejection [218]. Thus, as Table 3.1 shows, this kind of system produces two types of decision hits and errors [219, 220] which are depicted in Figure 3.8:

- True Positive (TP), correct detection or hit: which refers to when a sample of a true identity or target is above the threshold and is correctly accepted.
- True Negative (TN) or correct rejection: which is related to the correct rejection of an impostor identity.
- Type I error, False Alarm (FA), False Positive (FP), False Match (FM) or False Acceptance (FA): which refers to when an impostor is incorrectly accepted.
- Type II error, Miss, False Negative (FN), False Non-Match (FNM) or False Rejection (FR): which is related to the incorrect rejection of a true identity or target.

		Truth	
		Positive - Target	Negative - Impostor
Decision	Accept	True Positive (TP)	False Positive (FP)
	Reject	False Negative (FN)	True Negative (TN)

Table 3.1: Decision hits and errors types.

Based on these hits and errors, the probability of occurrence rates can be defined as Table 3.2 shows:

- Sensitivity or True Positive Rate (TPR): which is the number of TPs in relation to the total number of target identities.

		Truth	
		Positive - Target	Negative - Impostor
Decision	Accept	Sensitivity: $TPR = \frac{TP}{TP + FN}$	False Alarm Rate: $FPR = \frac{FP}{FP + TN}$
	Reject	Miss Rate: $FRR = \frac{FN}{TP + FN}$	Specificity: $TNR = \frac{TN}{FP + TN}$

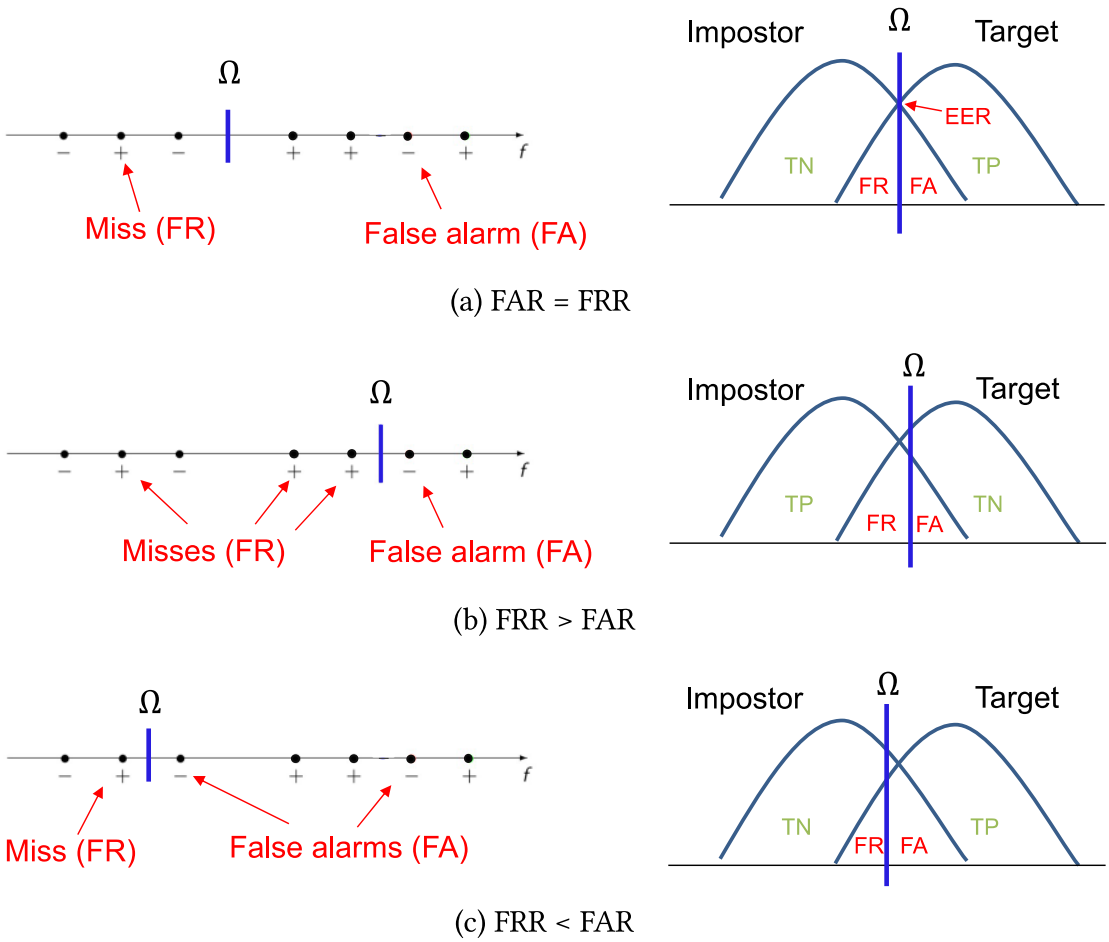
Table 3.2: Probabilities of hits and errors rates.

- Specificity or True Negative Rate (TNR): which corresponds to the number of TNs with respect to the total number of impostors.
- Probability of False Alarms, False Alarm Rate (FAR) or False Positive Rate (FPR): which is defined by the number of times of a FA occurs in relation to the number of impostors.
- Probability of Misses, Miss Rate or False Rejection Rate (FRR): which is the average number of times that the scores are smaller than Ω and a miss is produced with respect to the number of target identities.

Therefore, the main challenge of this step is to position the decision threshold. The selection of the threshold is what relates the system to the operating point of interest in terms of the application, and there are three different types of cases. Firstly, when FAR = FRR as in Figure 3.8(a), we are at the point known as Equal Error Rate (*EER*). *EER* is an operating point that is frequently used as a measure of the discrimination capability of the system for many applications, especially commercial ones [221]. However, *EER* may not be the best choice for some applications, so there are alternative operating points [222]. For example, if Ω is set to a high value for a high-security application, the number of false alarms may decrease even though a higher number of false rejections occur [223]. Thus, for systems where the cost of accepting an impostor is high, the threshold is chosen such that FRR > FAR as in Figure 3.8(b). On the other hand, if the application of the system requires a lower number of false rejections, such as forensic evidence evaluation, Ω will be set lower, but this involves that the number of accepted impostors increases, so in this case, FRR < FAR as in Figure 3.8(c).

3.8 Performance Metrics

As described in the previous section, the main objective of these systems is to determine whether each access attempt is made by a target or by an impostor person. Hence, when



(a) FAR = FRR

(b) FRR > FAR

(c) FRR < FAR

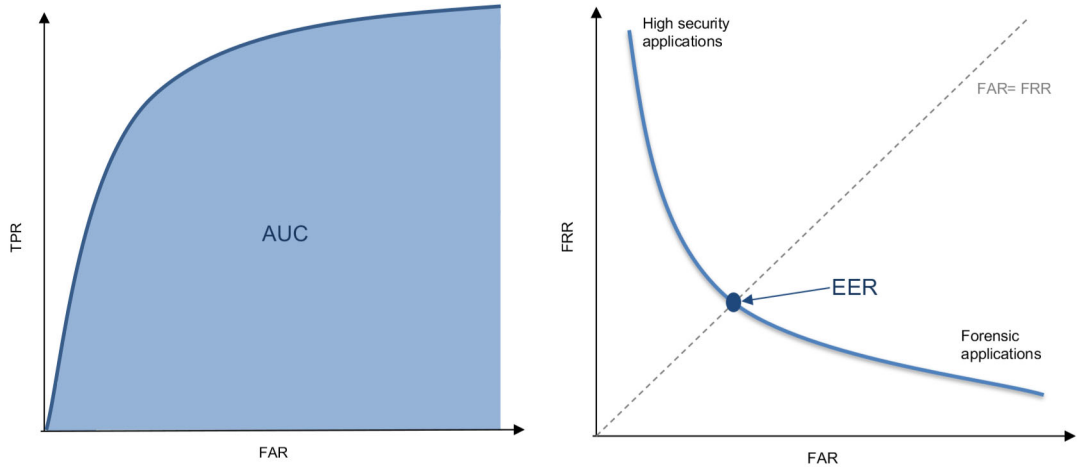
Figure 3.8: Decision errors based on the decision threshold (Ω) for speaker verification systems. Different possible cases depending on the amount of each type of error. (a) Case where FAR is equal to FRR. (b) The number of FRR are greater than FAR. (c) Greater number of FAR than FRR.

these systems are created, they try to minimize decision errors. Thus, to analyze and evaluate the behaviour of the biometric recognition systems proposed in this dissertation, we have measured the performance using six metrics. Moreover, it is common to represent the performance using two types of curves that depicts the trade-off between decision errors.

3.8.1 Receiver Operating Characteristic Curve and Detection Error Trade-off Curve

The first curve used to represent system performance is the Receiver Operating Characteristic curve (ROC) which is shown in Figure 3.9(a). The ROC curve is a graphical plot that has been widely adopted to show performance in biometric systems. The aim of this representation is to analyze the power of the systems to detect as many TPs as possible and minimize FPs. Therefore, in this curve, the relation between the sensitivity or TPR and FPR or FAR is shown on a linear scale.

On the other hand, the Detection Error Trade-off curve (DET) is a variant of the previous ROC curve. The DET curve represents in a visual plot the decision errors sweeping the threshold over all operating points. Non-linear warping has been applied to this curve on both axes, which spreads out the plot and produces curves that are closer to straight lines if the scores are better calibrated. In this case, instead of plotting the correct detection rate, this plot represents the trade-off between the probability of false alarms or FAR and the probability of misses or FRR as Figure 3.9(b) depicts.



(a) ROC curve and AUC

(b) DET curve and EER

Figure 3.9: (a) Left: Example of ROC curve and AUC of this curve. (b) Right: Example of DET curve, *EER* operating point, and also different application operating points.

3.8.2 Area Under ROC Curve and Equal Error Rate

The previous curves allow representing the overall capabilities of the systems at the different operating points. Nevertheless, a single performance number is usually employed to represent the capability of the systems. One of the most used metrics to obtain it is the Area Under ROC Curve (*AUC*) [224]. *AUC* measures the probability that samples from the target and non-target classes, m^+ and m^-, are ranked correctly based on the previously obtained scores. This metric is expressed as,

$$AUC = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} \mathbb{1}(s_i > s_j), \tag{3.17}$$

where s_i is the score of target class, s_j is the score of non-target class, and $\mathbb{1}()$ has a value equal to '1' whenever $s_i > s_j$, and '0' otherwise.

Another widely spread metric is Equal Error Rate (*EER*) [216, 225] which measures the discrimination ability of the system to separate true identities from impostors. *EER* represents the operating point at which the probability of FA is equal to the probability of FR as marked on the DET curve in Figure 3.9(b). This operating point is widespread in commercial applications, as it provides a general idea of the performance regardless of the number of samples in each class.

3.8.3 Detection Cost Function

As explained in the decision making section, the operating point is different according to the desired application of the system. Then, measuring only the *EER* value may not be the best option to show the performance. For this reason, Detection Cost Function (*DCF*) [226] is calculated and is usually shown in combination with *EER*. *DCF* is an application-dependent cost function to measure the discrimination and calibration power of the system. This function is based on the measure of the cost of detection errors in terms of a weighted sum of the FA and FR probabilities for some decision threshold and a prior probability of observing true (target) or impostor (non-target) identities. In this metric, two different values can be measured: the minimum and the actual detection cost (*minDCF* and *actDCF*). *minDCF* is defined as the optimal value of *DCF*. While *actDCF* is the computed cost for a fixed threshold obtained from the development data during the calibration step. To calculate this function, the general expression is defined as:

$$DCF = C_{fa} \cdot (1 - P_{tar}) \cdot P_{fa}(\Omega) + C_{miss} \cdot P_{tar} \cdot P_{miss}(\Omega), \tag{3.18}$$

where P_{tar} is the target prior probability, C_{fa} is the cost of false alarms, $P_{fa}(\Omega)$ is the probability of false alarms for a given threshold Ω, C_{miss} is the cost of misses, and $P_{miss}(\Omega)$ is the probability of misses for a given threshold Ω. In addition, depending on the application and the desired operating point, there are two main configurations in the bibliography for P_{tar}, C_{fa} and C_{miss} parameters that have been used to evaluate the systems in this thesis: NIST detection costs 2008 (*DCF08* [227]) and 2010 (*DCF10* [228]).

3.8.4 Log-Likelihood Ratio Cost

Log-Likelihood Ratio Cost (*CLLR*) [216, 226] is an application-independent measure of the discrimination and calibration performance of the system. This metric can be seen as a generalization of *DCF* since *CLLR* is defined as an integral over all possible operating points of *DCF*. As occurs with *DCF* metric, *CLLR* has two different values to measure: minimum Log-Likelihood Ratio Cost (*minCLLR*) and actual Log-Likelihood Ratio Cost (*actCLLR*). *minCLLR* indicates the optimal *CLLR*. While *actCLLR* represents the value obtained during the calibration step. The integral of *CLLR* is calculated as,

$$CLLR = \int_{\Omega} DCF(\Omega)\, d\Omega, \tag{3.19}$$

where Ω is the decision threshold, and $DCF(\Omega)$ is the *DCF* value at each operating point. However, to measure this value, the following analytic expression is employed:

$$CLLR = \frac{1}{2\log 2}\left(\frac{\sum_{y_i \in y_{tar}} \log(1 + exp(-s_{tar}(x_i, y_i)))}{N_{tar}} + \frac{\sum_{y_i \in y_{non}} \log(1 + exp(s_{non}(x_i, y_i)))}{N_{non}}\right) \tag{3.20}$$

where s_{tar} are the scores of target class, and s_{non} are the scores of non-target class.

3.8.5 Diarization Error Rate

Apart from the above metrics, another metric has been used to measure the performance of multimodal diarization systems. As reference metric to evaluate these systems, Diarization Error Rate (*DER*) is usually employed. However, in this thesis, *DER* is obtained slightly different than the original metric since it also takes into account the measurement of the identity assignment errors. To analyze better the results obtained with the *DER* metric, this metric can be decomposed in the three terms of error:

- *Probability of misses (MISS)*: which indicates the segments where the target identity is presented but the system does not detect it.
- *Probability of false alarm (FA)*: which illustrates the number of errors due to the assignment of one enrollment identity to a segment without identity known.
- *Identity error (ID)*: which reflects the segments assigned to enrollment identities different from the target identity.

3.9 Experimental Framework

This section presents the databases used in this thesis to develop and evaluate the different biometric recognition systems. A key element for the evaluation of the performance of these systems is the availability of large databases, which nowadays are growing more and more in both size and variety. Therefore, this section introduces the databases employed during this dissertation for face, text-dependent speaker and language recognition, and also for multimodal diarization.

3.9.1 Face Verification Datasets

The development of face verification systems has been carried out using three main databases.

Labeled Faces-in-the-Wild database

Many face databases have existed over the years, but the desire to study face recognition systems in unconstrained situations led to the creation of Labeled Faces-in-the-Wild (LFW) [229, 230]. This dataset was collected from the web in 2007 with different variations such as changes in posture, facial expression, race, gender, illumination and other parameters. Moreover, the number of identities increased with respect to the previous existing databases. The LFW dataset contains 5.749 different celebrity identities with 13.233 facial images.

CASIA-WebFace database

The need for a large-scale dataset motivated the development of the CASIA-WebFace dataset [231] that was built in 2014. The facial images to create this dataset were selected from the IMDb website. This dataset is composed of 494.414 images of 10.575 individuals. In many previous works, it has been employed as a training set due to this dataset is one of the largest public corpora for the face recognition task.

Mobile biometrics database

The mobile biometrics (MOBIO) database [232] is a bi-modal, face and speaker database containing video recordings from mobile devices collected in 2010. This database was the first captured almost exclusively on mobile phones. It contains 152 individuals with 52 females and 100 males, where each person has 12 sessions with over 61 hours of audio-visual data. To enroll the people, 5 facial images per identity were employed. Using this database, the systems have to cope with the different variabilities caused by acquisition in a mobile environment.

3.9.2 Text-Dependent Speaker Verification Datasets

To carry out the speaker verification experiments, two different text-dependent speaker verification datasets have been used.

RSR2015 database

Initially, we have used the RSR2015 text-dependent speaker verification dataset [124] for experiment development as it was the only one composed of several phrases and with publicly available train and test sets for text-dependent phrase-based speaker verification. This dataset consists of three different parts with speech samples from 157 males and 143 females. For each speaker and part, there are 9 sessions in which 30 different phrases are uttered. Furthermore, this corpus is divided into three speaker subsets: background (bkg), development (dev) and evaluation (eval). We develop our experiments with Part I and Part II, and employ the bkg and dev data (194 speakers, 94 females/100 males) for training. The evaluation part is used for enrollment and trial evaluation. Part I contains 30 phonetically balanced pass-phrases. Part II is based on 30 short control commands which have a strong overlap of lexical content. Moreover, this dataset has three evaluation conditions, but in this work, we have only evaluated the most challenging condition, which is the Impostor-Correct case where non-target speakers pronounce the same phrase as target speakers. This condition is also the most commonly employed in text-dependent speaker verification.

DeepMine database

The above database has been widely used to develop text-dependent speaker verification systems for several years, but it is not very large. For text-dependent speaker verification tasks, existing large databases are usually not publicly available. To address this issue, the multipurpose DeepMine dataset has been recently released [233]. The DeepMine dataset was released in 2019 and was created mainly to provide a large-scale database for text-dependent speaker verification purposes. This database is composed of three different parts with English and Persian phrases. For the experiments with this database, we employ the selected files from Part 1, which are used in Task 1 of the Short-duration Speaker Verification (SdSV) Challenge 2020 [234]. Part 1 corresponds to text-dependent part and consists of 5 Persian phrases and 5 English phrases. This part contains about 100.000 files from 963 female and male speakers.

3.9.3 Language Verification Datasets

Language Recognition Evaluation

In this thesis, the language recognition experiments have been performed with the database that was provided by the National Institute of Standards and Technology (NIST) for the Language Recognition Evaluation (LRE) 2009 [235]. Specifically, we have used a division of LRE09 development data that was proposed in [236]. The 49 original LRE09 development languages with 41793 files are used for training the different systems. For testing, we use test segments of the 23 target languages, each cut to consist of just 8 seconds of speech. We focus on the closed-set language recognition problem in which there are no out-of-set examples since all test examples belong to one of the 23 target languages. Also, we use the evaluation part of the LRE15 [237] corpus and the LRE17 [238] corpus to perform a final test since these databases are composed of clusters of closely related languages, useful for dialect discrimination.

3.9.4 Multimodal Diarization Datasets

In the case of multimodal diarization, images and audios will be extracted from the following video databases to perform diarization and identity assignment on them.

RTVE 2020 Challenge

The RTVE 2020 Challenge is part of the 2020 edition of the Albayzin evaluations [239, 240]. This dataset is a collection of several broadcast TV shows in Spanish language covering different scenarios. To carry out this challenge, the database provides around 40 hours of shows from Radio Televisión Española (RTVE), the Spanish public Radio and Television. The development subset of the RTVE2020 database contains two of the parts

of the RTVE 2018 database ($dev2$ and $test$ partitions) which are formed by four shows of around 6 hours. Furthermore, this subset also contains a new development partition with nine shows of around 4 hours. The evaluation or test set consists of fifty-four video files of around 29 hours in total with timestamps of speakers and faces. Enrollment data is also provided for 161 characters with 10 images and a 20-second video of each character.

Part II

Representation Learning

4

File Level Representation using Deep Neural Network Embeddings

4.1 Motivation

In the first part of this thesis, the theoretical background of biometric recognition systems has been introduced. As already mentioned, these systems are attracting a great deal of research interest in which many biometric attributes can be used. Among all the multiple options, this thesis is based on face and voice recognition, as they have become a crucial part of day-to-day life. Since, nowadays, most of the personal devices have a camera and a microphone, which allow capturing face and audio easily. This fact has led to these systems being used for different applications. Although the technology behind this kind of systems is quite mature and has undergone a breakthrough in recent

years with the spread of AI algorithms, especially Deep Neural Networks (DNNs), there are still some issues to be solved, such as overfitting problems due to limited data that generate problems to generalize to unseen data. Moreover, in some cases, applying techniques with impressive results in a specific task to another task may not produce the expected behaviour. Therefore, in this thesis, we propose different approaches to solve these problems. However, to introduce these approaches, we first need to create two reference systems similar to the main state-of-the-art approaches. Hence, in this chapter, we present face and text-dependent speaker verification systems that will serve as baseline systems throughout this thesis.

The remainder of this chapter is organized as follows: the baseline architectures for the face verification system are described in Section 4.2. Section 4.3 presents the experimental results for the face verification systems. The initial architectures used as baselines for text-dependent speaker verification systems are introduced in Section 4.4. Finally, the text-dependent speaker verification results and analysis of the results are described in Section 4.5.

4.2 Baseline Architecture for Face Verification

To establish the baseline system for face verification, a data processing step is initially carried out. In this step, first, the face images are passed through a face detector. This detector is a system of alignment and detection based on a DNN called Multi-task Cascaded Convolutional Networks (MTCCN) [241]. We have used this implemented system as it is an effective and proven method for face detection, which is necessary to do before proceeding with the rest of the face verification pipeline. Subsequently, the detected images are used as input to train the neural network architecture. To perform this training, at the beginning of the process, the images are randomly cropped to a size of 160×160 pixels, mean and variance normalization are applied, and data augmentation methods are employed. Aside from applying a random crop, a random horizontal flip and Random Erasing (RE) [206] are also used as data augmentation in this case. The RE data augmentation technique will be explained in detail later. Once the images are processed, the neural network as representation method is trained to optimize a multi-class classification objective loss function such as the traditional Cross-Entropy (CE) loss. As first neural network, a widespread architecture in the image field has been employed. This architecture is a WideResnet [82] which is an evolution of the Resnet architecture [79] and has a general structure as shown in Figure 4.1. The developed architecture is composed of twenty-five convolutional layers. Moreover, this kind of architecture is based on a neural network with wide residual blocks with two-dimensional convolution (2D convolution) layers, a pooling layer typically using a global average operation, a fully-connected (linear) layer, and finally a loss layer such as softmax. The detailed information of each layer and its dimensions can be found in Table A.1 of Appendix A. Once the training of this architecture is completed after 150 epochs using Stochastic Gradient Descent (SGD) optimizer with a learning rate that decreases from 10^{-1} to 10^{-5}, file level representations

known as embeddings are extracted from the pooling layer to carry out the verification process. Typically, for the verification process, a simple cosine similarity is applied to obtain the scores and the final performance metrics are calculated. To implement this system, the CASIA-WebFace dataset is employed for training and the MOBIO dataset is used for evaluation.

On the other hand, the development of the previous system was limited in depth and width due to the computational resources available. Nevertheless, contemporaneously with the start of this thesis, several powerful state-of-the-art systems were implemented and made available in public repositories. Therefore, we have established a second baseline using a pretrained model based on Inception ResNet [242] with more than one hundred layers [75]. This network was also trained for a classification task on the CASIA-WebFace dataset with the same general structure as Figure 4.1 depicts. Hence, we decided to use this system to improve the initial baseline results, as the embeddings extracted from this model were previously tested in a verification task with the LFW database to check their discriminative ability with impressive results. Using this network, the processing of the face image is similar to the previous one, mean and variance normalization is applied, and it is resized to 160 × 160 pixels by applying a central cropping. Next, the processed images are passed through a trained model to obtain an embedding representation of each file to perform the verification process, for which a cosine similarity is also employed.

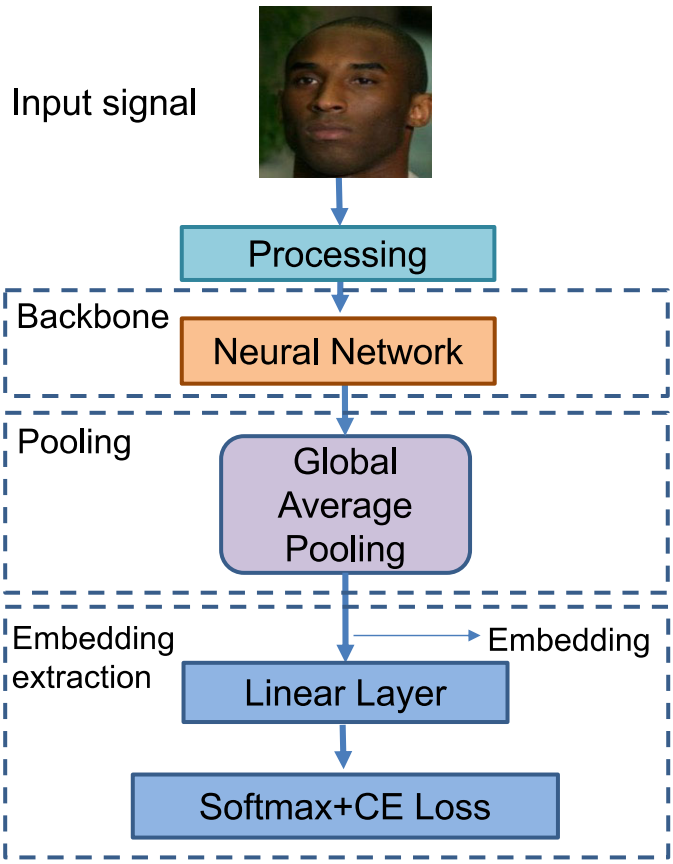


Figure 4.1: Baseline system for face verification.

4.3 Results of Face Verification Baseline Systems

In this section, the results obtained with both systems are presented. Table 4.1 shows the performance in terms of Equal Error Rate (*EER%*), Area Under the ROC Curve (*AUC%*), minimum Detection Cost Function 08 (*minDCF08*), minimum Detection Cost Function 10 (*minDCF10*) and minimum Log-Likelihood Ratio Cost (*minCLLR*) for the female and male data from the MOBIO dataset. Furthermore, we also include the results for this dataset using the simple Convolutional Neural Network (CNN) with only nine layers developed in [243]. The introduction of this result allows us to show more clearly the effect of training a deep system with CASIA-WebFace to improve performance. Note that both baseline systems introduced in this thesis are powerful architectures that show a relevant improvement over a system with a simpler architecture. This effect confirms the need to use larger architectures when the training data is large to achieve good performance.

Table 4.1: Experimental results on MOBIO [244] eval set, showing *AUC%*, *EER%*, *minCLLR*, *minDCF08*, and *minDCF10*. These results were obtained to compare the baseline systems for face verification.

	Architecture				**Female**		
Type	**Nº Layers**	**Nº Param.**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
CNN	9	32.31M	11.89	94.91	0.492	0.752	0.403
WideResnet	25	43.31M	4.72	99.12	0.210	0.505	0.166
Facenet	> 100	140M	**2.14**	**99.76**	**0.086**	**0.201**	**0.079**
	Architecture				**Male**		
Type	**Nº Layers**	**Nº Param.**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
CNN	9	32.31M	7.04	98.11	0.273	0.608	0.243
WideResnet	25	43.31M	1.22	99.91	0.060	0.148	0.049
Facenet	> 100	140M	**0.59**	**99.94**	**0.036**	**0.154**	**0.028**
	Architecture				**Female+Male**		
Type	**Nº Layers**	**Nº Param.**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
CNN	9	32.31M	8.34	97.38	0.342	0.9674	0.289
WideResnet	25	43.31M	2.31	99.75	0.107	0.317	0.087
Facenet	> 100	140M	**1.05**	**99.92**	**0.051**	**0.184**	**0.043**

4.4 Baseline Architecture for Text-Dependent Speaker Verification

In view of the results achieved with the powerful architectures in the previous section, we decided to implement a text-dependent speaker verification system using a similar WideResNet architecture for the backbone part. This type of architecture had also been successfully applied for text-independent speaker verification [137]. Moreover, in text-dependent speaker verification, an attempt was carried out to introduce these convolutional architectures in [166]. However, that work did not achieve the same good results as simple architectures with only dense layers [165] for the RSR2015 database, which is the text-dependent speaker verification dataset initially employed in this thesis. In addition,

since this database employed to create this system is smaller than the CASIA-WebFace dataset, we have also proposed to compare performance with a second baseline system that is composed of a straightforward architecture with only a few convolutional layers. To evaluate these architectures, cosine similarity is employed as a metric. This metric is simple and allows us to perform a fast evaluation of the trials using a matrix structure, which reduces the time required for the entire evaluation. The structure of the system architecture for both baseline systems is the same as shown in Figure 4.2 where the only difference lies in the number and type of layers used in the backbone which are described in Tables A.2 and A.3 of Appendix A.

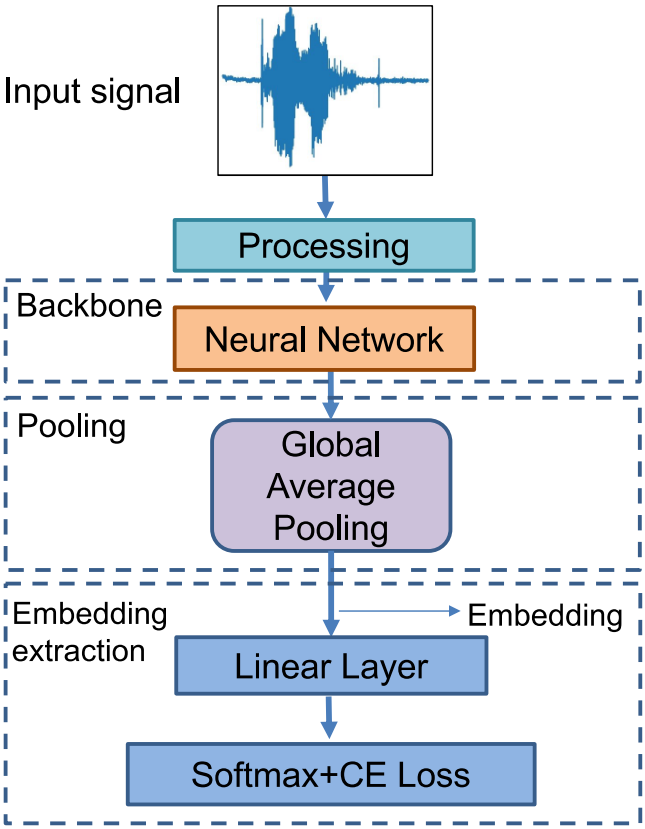
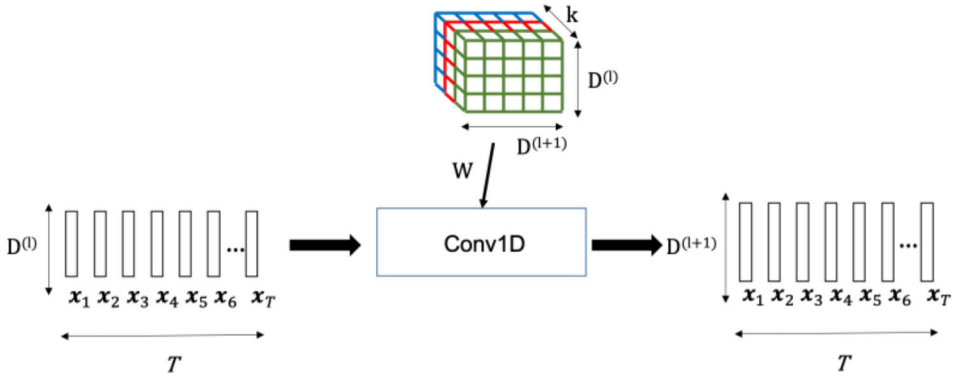


Figure 4.2: Baseline system for speaker verification.

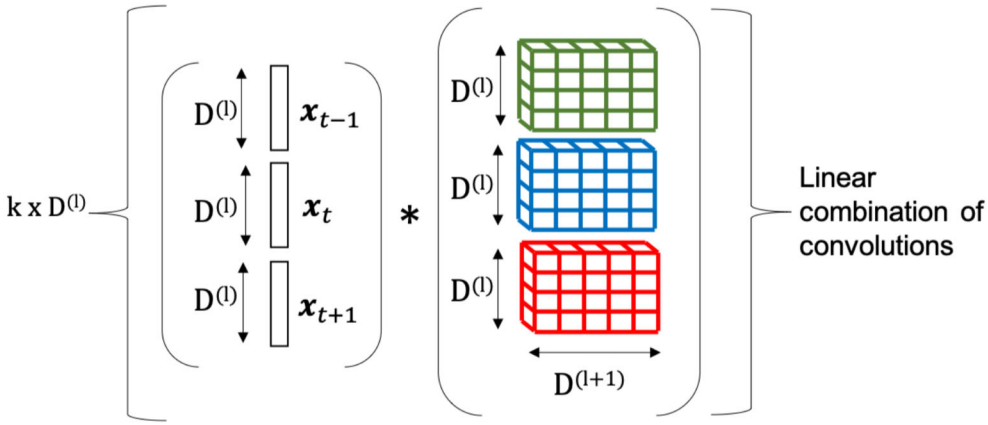
In the processing step in both systems, the raw signals are processed to calculate a set of features composed of 20 Mel-Frequency Cepstral Coefficients (MFCCs) [96, 245] with their first and second derivatives as features to obtain a final input dimension of 60 using a window length of 25 ms and an advance of 10 ms. Then, an energy based Voice Activity Detector (VAD) is used on them. These features are employed as input to train the WideResNet and simpler architectures. On the other hand, we have made some modifications to these architectures with respect to the one used for the face verification system. The first modification is the replacement of the 2D convolution layers by one-dimensional convolution (1D convolution) layers since in this case, the input signals to the network are audio samples. Furthermore, a new data augmentation technique is introduced in the audio processing step. This new technique is known as Random Erasing (RE) [206]. Both modifications will be explained in detail below.

4.4.1 1D Convolution Layer

Regarding the layers applied to create the backbone of text-dependent speaker verification systems, we have proposed the use of 1D convolution layers instead of dense layers or 2D convolution layers as in the face verification system. Our proposal is to operate on the temporal dimension to add context information to the process, and at the same time, all channels are combined in each layer. The context information added depends on the size of the kernel k used in the convolution layer.



(a) Operation with 1D Convolution.



(b) Example of the convolution operation.

Figure 4.3: Operation with 1D Convolution layers, (**a**) general pipeline of this operation; (**b**) example of how k context frames from input are multiplied by the weight matrix W and the output is equivalent to a linear combination of convolutions.

The operation of the 1D convolution layers is depicted in Figure 4.3. To efficiently train this type of layers, it is desirable that all input signals have the same length in order to concatenate them and pass them to the network. It is possible to obtain this fixed length by applying an interpolation transformation or by padding the input signals with zeros. In this thesis, we have used an interpolation to have all of them with the same dimensions ($D^0 \times T$) where D^0 is the acoustic feature dimension, and T is the temporal dimension. Then, the signal used as input to the layer and its context, the previous and the subsequent

frames indicated by the kernel size, are processed by the 1D convolution layer, where they are multiplied frame by frame with the corresponding weight matrix to produce a new vector sequence. The result of this operation is that each frame and its context are linearly combined to create the output signal. During the training process with these convolution layers, the value of the temporal dimension (T) is kept until the pooling block, while the acoustic feature dimension changes when it passes through the network layers in function of the combination of channels in each layer. As we can see in Figure 4.3, each convolution layer will have feature vectors of dimension $D^{(l)}$ as input and $D^{(l+1)}$ as output. The resulting dimension of the signal processed through the network will be $D = (D^{(0)}, D^{(1)}, ..., D^{(l)}, ..., D^{(L)})$, where L is the total number of backbone layers.

4.4.2 Random Erasing

Focusing on the idea of adding some extra variability to the input signals, the use of convolution layers allows us to introduce a new data augmentation method for the speaker verification task which is the Random Erasing (RE) data augmentation [206]. This method was developed to improve the ability to generalize in image tasks using 2D convolution layers. However, we have used a modified version of this method, one-dimensional RE (1D RE), to apply it combined with 1D convolution layers. Figure 4.4 depicts a random perturbation applied on F consecutive time frames at a random position in the sequence. The number F of corrupted frames is drawn from an uniform distribution, $N \sim U(F_{min}, F_{max})$, where F_{min} is the minimum number of corrupted frames, and F_{max} is the maximum number.

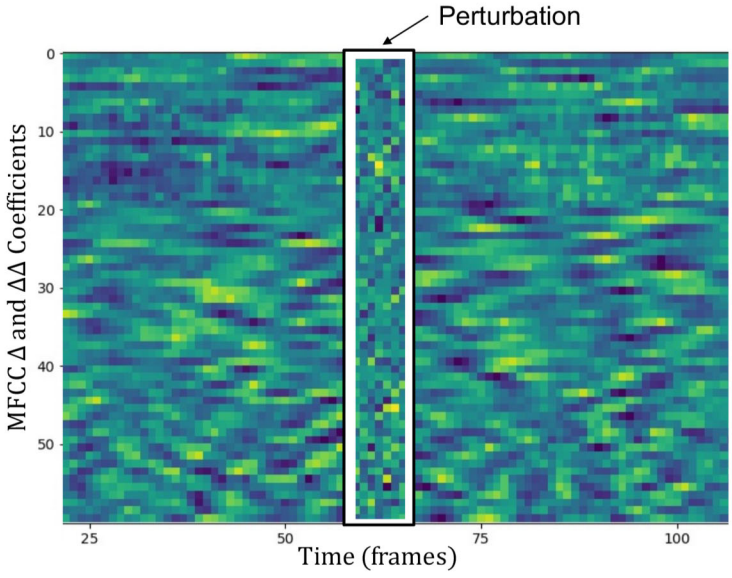


Figure 4.4: 1D Random Erasing transformation applied over input sample after the feature extractor and the padding and normalization transformations.

4.5 Results and Analysis of Text-Dependent Speaker Verification Baseline Systems

This section presents the results of the two baseline systems for the text-dependent speaker verification task. In Table 4.2, we show *EER%*, *AUC%*, *minDCF08*, *minDCF10* and *minCLLR* results with both systems trained on the bkg subset for females, males and both partitions of the eval subset together from the RSR2015-Part I dataset. We have found that these approaches do not perform well for this task, regardless of network depth. It is noteworthy that the smaller architecture achieves slightly better results. Nevertheless, these results are worse than expected, as DNNs combined with traditional approaches such as Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs) or i-vectors [154, 159] exhibit *EER* values around 1%.

Table 4.2: Experimental results on RSR2015-Part I [124] eval set, showing *AUC%*, *EER%*, *minCLLR*, *minDCF08*, and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the baseline systems for text-dependent speaker verification.

	Architecture				**Female**		
Type	**Nº Layers**	**Nº Param.**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
WideResnet	25	21.51M	11.09	95.74	0.526	0.968	0.378
CNN	3	0.02M	**9.11**	**97.04**	**0.453**	**0.959**	**0.315**
	Architecture				**Male**		
Type	**Nº Layers**	**Nº Param.**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
WideResnet	25	21.51M	12.74	94.79	0.619	0.975	0.417
CNN	3	0.02M	**8.67**	**97.22**	**0.460**	**0.963**	**0.306**
	Architecture				**Female+Male**		
Type	**Nº Layers**	**Nº Param.**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
WideResnet	25	21.51M	12.02	95.23	0.578	0.972	0.401
CNN	3	0.02M	**8.88**	**97.14**	**0.458**	**0.963**	**0.311**

To analyze what may be happening, we have computed the cosine similarity of the embeddings of the samples in the evaluation subset. Due to the computational cost of calculating this similarity for the entire subset, we have depicted only the first ninety samples in Figure 4.5. In view of this representation, it seems that embeddings with good potential representation ability have been obtained, but it does not match the numerical performance in Table 4.2. Therefore, we decided to analyze some more examples of test embeddings.

For this purpose, we have calculated the cosine similarity matrix shown in Figure 4.6. To compute this matrix, we have used the embeddings of the first female speaker pronouncing the thirty phrases composing the RSR2015-Part I dataset, and also, the second female speaker pronouncing the first twenty phrases. In this illustration, we observe that the embeddings extracted from the DNN after the global average pooling produce high cosine values when comparing the embeddings for the same identity and phrase. On the other hand, whether we pay attention to the cosine similarity of the comparison of the second speaker pronouncing the same phrase as the first speaker, we can see that the

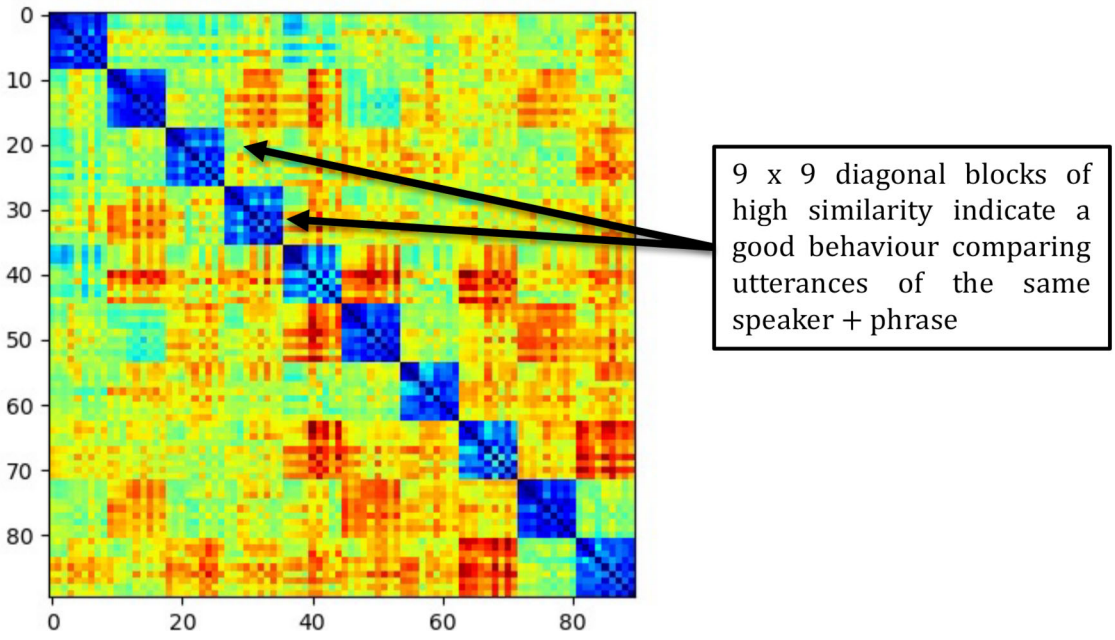


Figure 4.5: Matrix of cosine similarity with the first female speaker pronouncing the first ten phrases.

value is also high. Therefore, the final verification process can be affected by the confusion generated by different speakers pronouncing the same phrase. This effect could be the reason for the poor performance of the system.

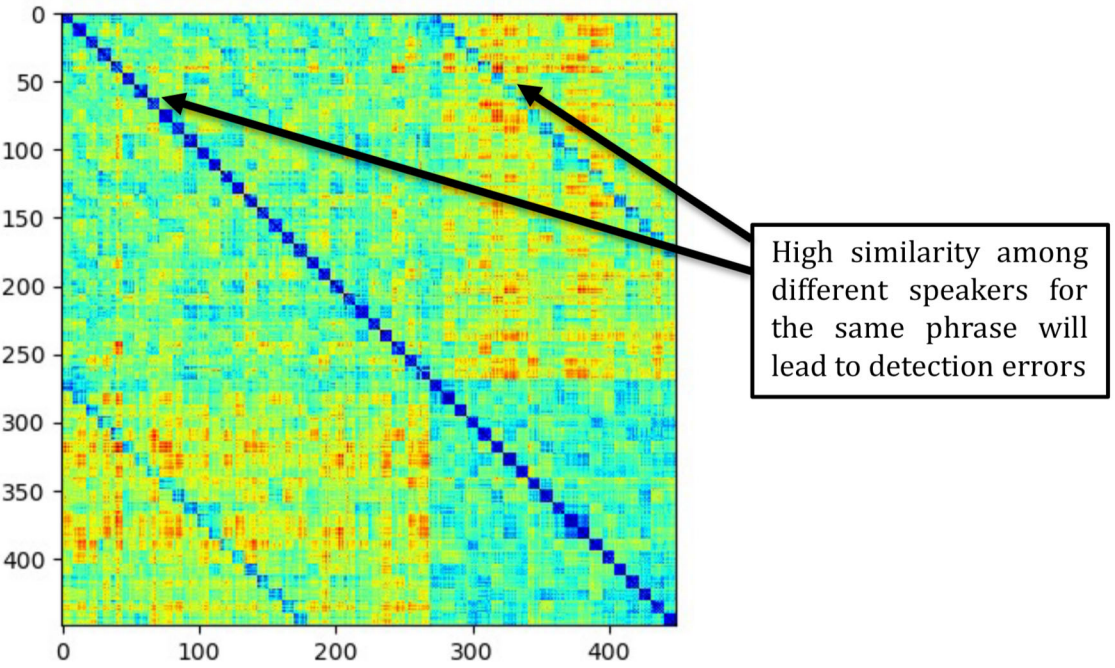


Figure 4.6: Matrix of cosine similarity with more embeddings where problems to distinguish between two different female speakers pronouncing the same phrase are shown.

Unfortunately, the results and the cosine similarity matrices depicted show that the application of this kind of system does not work efficiently on text-dependent tasks with

small training databases. A possible cause of inaccuracy in text-dependent tasks could be derived from the use of the global average pooling mechanism as a representation of the utterance. Since for text-dependent speaker verification tasks, the uttered phrase is a relevant piece of information to correctly determine identities and the system has to detect a match in speaker and phrase to be correct, as we will show in the next chapter.

5

Deep Neural Network Supervectors

5.1 Motivation

As described in Chapter 4, the use of techniques based on discriminative Deep Neural Network (DNN) combined with a global average pooling mechanism does not work efficiently for text-dependent speaker verification tasks. Most text-independent speaker verification approaches to obtain speaker representations or embeddings from a whole utterance employ this type of approaches in which temporal information is reduced by calculating the average across frames of internal representations of the network. Thus,

this type of systems only maintain the information of who is speaking and they may not capture phonetic information for the final identification process. This approach may neglect the order of phonetic information because in the same phrase the beginning of the sentence may be totally different from what is said at the end. An example of the relevance of keeping this order is the case where the system asks the speaker to utter digits in a random order. In that case, a mean vector would fail to capture the combination of the order of the uttered digits and speaker. For that reason, we show that it is important to keep this information for the identification process in text-dependent speaker verification tasks, and not only the information of who is speaking, since the order of the phonetic information of the uttered phrase is a relevant part of the identity information.

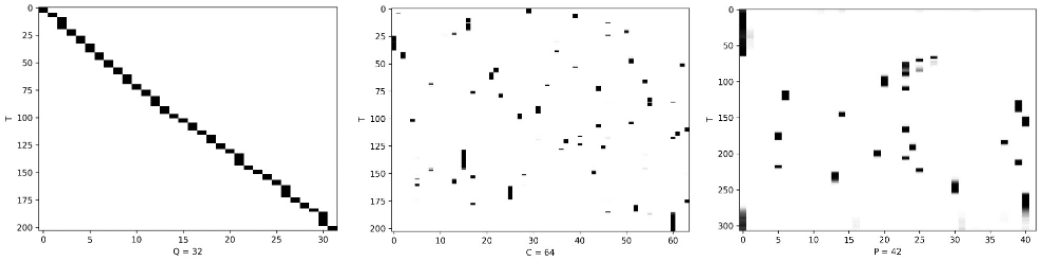
To solve this problem, this thesis proposes a new architecture that includes a phonetic phrase alignment method as a key component of the mechanism to obtain the vector representation of a DNN. Unlike previous works, we substitute the average of internal representations across time used in other neural network architectures [165, 166] by a differentiable alignment mechanism applied as a new internal layer to keep the temporal structure of each utterance. We show how the alignment can be applied in combination with a DNN acting as a backbone to encode speaker and phrase information in a supervector for each utterance thanks to the DNN and the specific states of the supervector. A supervector is a concatenation of lower-dimensional vectors of each specific state into a higher-dimensional vector. As we will show, the application of both sources of information in the process of defining the supervector provides better results in the experiments performed with the RSR2015 database compared to previous approaches.

Therefore, the aim of this chapter is to explore different approaches to generate signal representations or embeddings introducing a differentiable alignment mechanism into the DNN models which preserve the subject identity and uttered phrase. In the following section, we present the alignment mechanism and the two techniques employed in this thesis. Later on, we will describe the way to use this mechanism as an internal layer in the architecture. Afterwards, the results for these approaches will be introduced. Finally, we explain the conclusions about this alternative to the global average pooling mechanism.

5.2 Alignment Mechanism

Due to the poor performance of the results achieved in the previous chapter for this task with DNNs combined with global average pooling, we have developed a differentiable alignment mechanism for DNNs to replace the usual pooling because of the importance of the phrases and their temporal structure in this type of task. Since the same person does not always pronounce a phrase at the same speed or in the same way due to differences in the phonetic information, it is usual that there exists an articulation and pronunciation mismatch between two speech utterances compared even from the same person. Thus, for the success of the text-dependent speaker verification task, a new mechanism is necessary to encode the phrase and speaker information of the audio file into a represen-

tation vector. This vector has to be able to keep the order of the phonetic information, so for this reason, different alignment mechanisms have been proposed to use their states or components to maintain this information in the representation vector. Therefore, using the alignment mechanism, the representation obtained has a common mechanism with the traditional supervector in speaker verification. It has the advantage of being discriminative against differences in phonetic information, which is necessary in our task, and, at the same time, it is robust against the different sources of variability produced when an utterance is pronounced. There are several alternatives to use as an alignment mechanism such as Hidden Markov Model (HMM), Gaussian Mixture Model (GMM) or DNN posteriors. An example of the alignment for each of these mechanisms can be seen in Figure 5.1. In Figure 5.1(a), the relation between time and states of an HMM is shown. On the other hand, Figure 5.1(b) depicts the relation between time and components of a GMM. Finally, Figure 5.1(c) represents the relation between time and DNN posteriors. The two alternatives employed in this thesis will be explained below.



(a) HMM (b) GMM (c) DNN posteriors

Figure 5.1: Examples of alignment for the different existent approaches. (a) Left: HMM alignment. (b) Center: GMM alignment. (c) Right: DNN Posteriors alignment.

5.2.1 Hidden Markov Model Alignment Mechanism

The first technique employed during this dissertation is a phrase HMM-based alignment mechanism [48, 246]. The general explanation of this type of model was detailed in Section 3.3.1. Using this approach, the knowledge of phonetic information is not necessary for the training process, so we can easily train an independent HMM with no skip states for each different phrase. Unlike HMMs generally used in Automatic Speech Recognition (ASR), whose states are associated with phonemes or context-dependent units, we have used one HMM per phrase with a fixed number of states and these models are trained in an unsupervised way. Therefore, we do not need to use annotations to define the states of the model from the transcribed sequence of phonemes for each phrase. Furthermore, this kind of alignment mechanism has a left-to-right architecture and employs the Viterbi algorithm to provide a decoded sequence in which all states of HMM have correspondence with at least one frame of the utterance, so no state is empty.

The process followed to add this frame-to-state alignment to our system is detailed below. Once the models for alignment are trained, we obtain a sequence of decoded states $\overline{q}=(q_1, ..., q_t, ..., q_T)$ where q_t indicates the decoded state at time t with $q_t \in \{1, ..., Q\}$ and

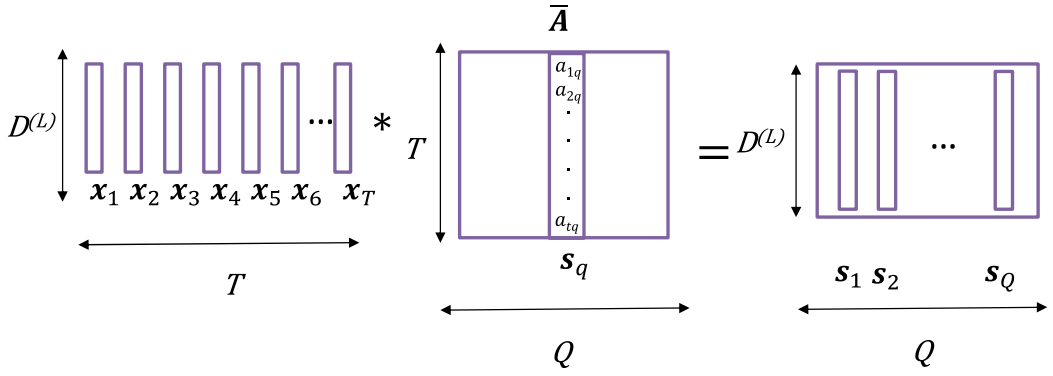


Figure 5.2: Process of alignment, the input signal x is multiplied by an alignment matrix A to produce a matrix with vectors s_Q which are then concatenated to obtain the supervector.

Q is the number of states. After that, these decoded sequence vectors are converted into a matrix with ones at every state q corresponding to the decoded sequence at time t and zeros in the remaining states, which makes possible to use them directly inside of the neural network. As a result of this process, we have the alignment matrix $A \in \mathbb{R}^{T \times Q}$ with its components $a_{tq_t}=1$ and $\sum_q a_{tq}=1$ which means that only one state is active at the same time.

For example, if we train an HMM with 4 states and we obtain a vector v and apply the previous transformation, the resultant matrix A would be:

$$v = [1,1,1,2,2,3,3,4] \longrightarrow A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

Once this process is over, as we can see in Figure 5.2, the alignment matrix is provided externally to the network for each utterance, giving the supervector as the output of a matrix product as one additional layer after the last layer L. The use of this matrix facilitates differentiation and enables to backpropagate the gradients to train the neural network. Therefore, each column of the output matrix is a linear combination of the sequence of vectors, and its derivative is straightforward. Matrix multiplication assigns the sum of the corresponding frames to each state vector, resulting in a supervector if we consider them concatenated. Then, speaker verification is performed with this supervector. This process can be expressed as a function of the output, $x_{dt}^{(L)}$, of the last layer L with dimensions $(D^{(L)} \times T)$ and matrix of alignment of each utterance $\bar{A}$ with dimensions $(T \times Q)$:

$$x_{dq}^{(L+1)} = \frac{\sum_t x_{dt}^{(L)} \cdot a_{tq}}{\sum_t a_{tq}} = \sum_t x_{dt}^{(L)} \cdot \bar{a}_{tq}, \tag{5.2}$$

where $x_{dq}^{(L+1)}$ is the supervector of the layer $(L+1)$ with dimensions $(D^{(L)} \times Q)$, where there are Q state vectors of dimension $D^{(L)}$ and we normalize with the number of times the state q is active, and $\bar{a}_{tq}$ are the normalized weights defined as $a_{tq}/\sum_t a_{tq}$.

5.2.2 Gaussian Mixture Model with Maximum A Posteriori Alignment Mechanism

The second proposed approach is the use of a GMM to generalize the previous method. GMM was introduced in detail in Section 3.3.2. Like the previous alignment mechanism, the phrase transcription is not needed to train a model for each phrase. Moreover, this method provides more flexibility since a single frame might not correspond to a single component in the mixture. They can be distributed, and there might be empty components for the whole sequence.

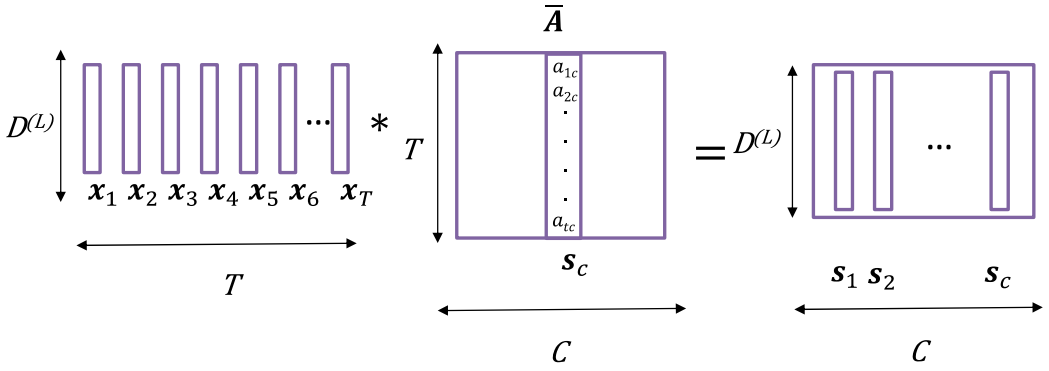


Figure 5.3: Process of alignment with GMM, the input signal x is multiplied by an alignment matrix A to produce a matrix with vectors s_C which are then concatenated to obtain the supervector. Note that we use the C to refer at the components of the GMM.

The philosophy of this frame-to-components alignment process is similar to the one described in the previous section, and it is depicted in Figure 5.3. However, in this case, we obtain the GMM alignment from the posterior probability of the hidden variables. Matrix $\bar{A}$ with dimensions $(T \times C)$ where C are the components of GMM is built by assigning the posterior probabilities of the Gaussian component c at time t to the elements of the matrix $a_{tc} = \gamma_t(c) = P(Z = c|\mathbf{x}_{dt}^{(L)})$:

$$x_{dc}^{(L+1)} = \frac{\sum_t x_{dt}^{(L)} \cdot a_{tc}}{\sum_t a_{tc}} = \sum_t x_{dt}^{(L)} \cdot \bar{a}_{tc}, \tag{5.3}$$

where $x_{dc}^{(L+1)}$ is the supervector of the layer $(L+1)$, a_{tc} are the weights of the component c, $x_{dt}^{(L)}$ is the input to this layer L, and $\bar{a}_{tc}$ are the normalized weights defined as $a_{tc}/\sum_t a_{tc}$. However, often, most of the posteriors are close to zero. To avoid the loss of performance

due to the sparness of matrix A, we add a MAP adaptation [106,107] that slightly modifies this process. To define how this layer operates, we employ the following expression:

$$x_{dc}^{(L+1)} = \frac{\sum_t x_{dt}^{(L)} \cdot \gamma_t(c) + \tau \cdot \mu_{dc}^{(b)}}{\sum_t \gamma_t(c) + \tau}, \tag{5.4}$$

where $\tau > 0$ is the relevance factor, $\gamma_t(c)$ is the posterior probability of Gaussian component c, and $\mu_{dc}^{(b)}$ is the running mean per component of the mixture that will be updated each batch b in a similar manner to a Batch Normalization layer [247]:

$$\mu_{dc}^{(b)} = (1 - \rho) \cdot \mu_{dc}^{(b-1)} + \rho \cdot f, \tag{5.5}$$

where ρ is the adaptation coefficient, and f is the mean estimation using the batch data samples.

The MAP adaptation ensures that the components with a low count of activations in a sequence will be assigned the mean value of the corresponding supervector section, $\mu_{dc}^{(b)}$, making the system more robust even in the case a component is not activated.

5.3 Deep Neural Network based on Alignment

As a first approximation to check that the above alignment layer works better than the extraction of the embedding from the global average pooling, we could apply this mentioned layer directly on the acoustic features to obtain the traditional supervector. However, we expect to improve this baseline result, so we propose to add some layers as a backbone network previous to the pooling part and train them in combination with the alignment mechanism. For this purpose, two different architectures are developed. In addition, we have added the architecture created in Chapter 4 to remark the changes introduced. These architectures are depicted in Figure 5.4 and are designed as follows:

- *Architecture A.* The architecture depicted in Figure 5.4(a) is the approach used in many recent verification systems, in which a DNN is trained for the multi-class classification which we developed in Chapter 4, and a global average pooling is applied to extract the embedding representation. Once the system is trained, an embedding is extracted for each enrollment and test file, and then a cosine similarity metric is applied on them to obtain the verification scores.

- *Architecture B.* As described in Chapter 4, the previous architecture with this kind of pooling is not the most suitable approach for text-dependent speaker verification tasks. For this reason, we substitute the global average pooling by the differentiable alignment mechanism explained in the above section, which allows us to keep the temporal structure of the utterance. Thus, we use the system in Figure 5.4(b) to verify that these alignment mechanisms work better than the global average pooling. This architecture consists of applying the alignment mechanism as a layer directly on the acoustic features, so we get the traditional supervector per each audio file.

This supervector can be seen as a mapping between an utterance and the state components of the alignment, which allows encoding the phrase information.

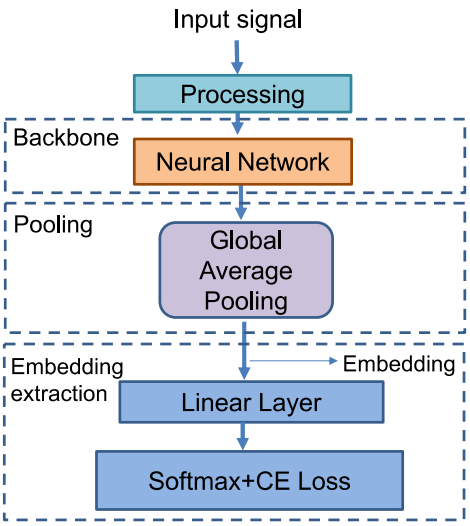
- *Architecture C.* Considering the good behaviour of the alignment layer and the possibility to propagate gradients, we can use this layer as a link between the backbone and loss parts, allowing us to train the whole system to optimize any cost function we decide. Thus, we create the architecture depicted in Figure 5.4(c) by combining the architecture type B with a backbone network, and a flatten layer is used to link with the last layer to train the system for multi-class classification with the CE loss function. To implement the backbone network, different configurations of the 1D convolution layers introduced in Chapter 4 have been used. For the verification process, once our system is trained, a neural network supervector is extracted from the flatten layer for each enrollment and test utterance, and then a cosine similarity is applied on them to produce the verification scores.

5.4 Experiments and Results

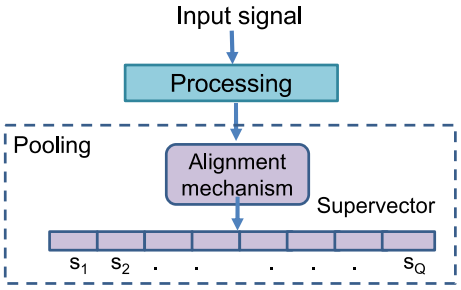
5.4.1 Experimental Setup

To validate the introduced approaches, we have used the RSR2015-Part I and II database presented in Chapter 3. As the audio processing step in these experiments, a set of features composed of 20 Mel-Frequency Cepstral Coefficients (MFCCs) with their first and second derivates are employed as features to obtain a final input dimension of 60. Then, an energy based voice activity detector is used on them. These features are employed as input to train the alignment mechanisms and also as input to the DNN. To train two different alignment mechanisms, the bkg partition of the RSR2015 dataset has been employed. Moreover, both were trained to obtain a model per phrase without the needed to know the phrase transcription. On the one hand, HMM models have been trained using a left-to-right model with no skip states, which has been developed with models of 10, 20, 40 and 80 states for each phrase. On the other hand, a 64 component GMM has been trained per phrase. With these models, we can extract the alignment information for use inside our DNN architecture. The lack of data may lead to DNN training problems, so we try to avoid a possible overfitting in our models with the Random Erasing (RE) data augmentation method explained in Chapter 4 which is applied on the input features. Furthermore, we train the neural networks using an Adam optimizer [248] with a learning rate of 10^{-4} during 300 epochs that is more robust than Stochastic Gradient Descent (SGD).

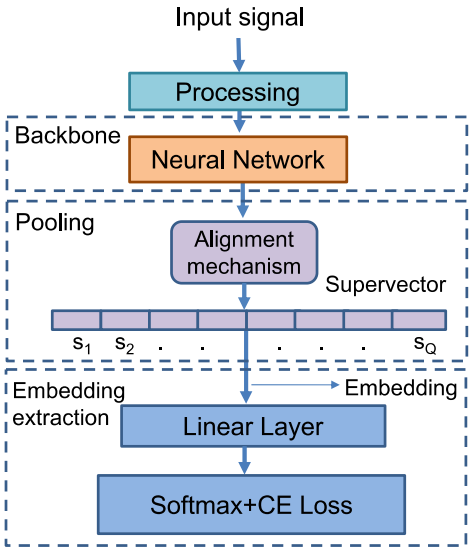
Along this chapter, two set of experiments have been performed to analyze the impact of using the two different alignment mechanisms proposed to replace the global average pooling. In addition, we have studied the effect of training with more data and also, using different configurations of the DNN architecture. To train all alternatives of the



(a) Architecture type A



(b) Architecture type B



(c) Architecture type C

Figure 5.4: The architectures developed to check the effectiveness of our proposed alignment mechanism, 5.4(a) the architecture type A is trained for multiclass classification using a traditional global average pooling mechanism. In 5.4(b) the architecture type B, the acoustic features are aligned directly to obtain the supervector. The supervector is composed of Q vectors s_Q for each state or component. 5.4(c) the architecture type C is trained for multiclass classification using the alignment mechanisms.

DNN architecture, CE loss has been employed, and to test a cosine similarity is applied on the embeddings. The diverse experiments have been performed using Pytorch [249].

5.4.2 Analysis of the Training Data and the Number of States using HHM Alignment with RSR2015-Part I and Part II

In the first set of experiments, we have analyzed the replacement of the usual global average pooling mechanism by a differentiable alignment based on HMMs as the pooling mechanism. In addition, we have also evaluated the interaction of different backbone architectures with the alignment and the utterance duration by means of several HMM sizes.

We compare a backbone using an average pooling with similar philosophy as [165, 166] which is detailed in Table A.3 in Appendix A, and the acoustic feature input directly aligned with the HMM alignment; thus, we obtain the traditional supervector. In these experiments, we have used the traditional supervector as the reference system instead of an i-vector based system due to the limitations of i-vectors for text-dependent tasks in the RSR2015 database, as pointed out in [250, 251], where the authors conclude that these types of datasets do not have enough data to properly train the traditional i-vector extractors. Additionally, we have evaluated several backbone architectures with four different configurations which are detailed in Appendix A in Tables A.4, A.5 and A.6: one convolutional layer with a kernel of dimension 1 equivalent to a dense layer but keeping the temporal structure and without adding context information, one convolutional layer with a kernel of dimension 3, three convolutional layers with a kernel of dimension 3, and four convolutional layers with a kernel of dimension 3. These configurations for kernel size and number of layers have been selected as simply as possible, since, as we mentioned in Chapter 4, DNNs have not achieved good results in this task. Finally, after all experiments, we have extracted embeddings using average global pooling or supervectors as a combination of backbone and alignment with a flatten layer, and, with them, we have obtained speaker verification scores using a cosine similarity metric without any normalization technique.

RSR-Part I

The analysis introduced above has been carried out on both parts of the RSR2015 dataset. First, we focus on RSR-Part I which is composed of fixed pass-phrases, e.g.,*"Only lawyers love millionaires"*. In these experiments, we study the behaviour of our system when we vary the number of backbone layers, the training data and the states of the HMM. In Tables 5.1, 5.2 and 5.3, we show Equal Error Rate (*EER%*) and minimum Detection Cost Function 10 (*minDCF10*) results with the different architectures trained on the bkg subset for females, males and both partitions together. As we shown in Chapter 4, the approach with a global average pooling mechanism to extract embeddings does not perform well for this text-dependent speaker verification task. It seems that this type of embeddings does not correctly represent the information to achieve discrimination between the correct speaker and the correct phrase simultaneously. For that reason, we seek an alternative to improve this discriminative ability, so we change this typical mechanism to a new alignment layer inside the DNN that achieves a relative improvement of 92.14% in terms

of *EER%*. In addition, we have evaluated the performance of our system as a function of whether we vary the number of states used to model each phrase. Given the length of the phrases in Part I, we observe that it is better to employ a larger number of states to correctly model each phrase.

Table 5.1: Experimental results on RSR2015-Part I [124] eval set, where *EER%* and *minDCF10* are shown. These female results were obtained by training only with a bkg subset and by varying the number of states of the HMM.

Architecture					**Results**	
Type	**Backbone**		**Pooling**		**Female**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	11.20	0.981
B	–	–	*HMM*	10	3.14	0.273
C	1*C*	1			1.75	0.276
	1*C*	3			1.60	0.250
	3*C*	3			1.72	0.315
	4*C*	3			2.31	0.384
B	–	–	*HMM*	20	1.81	0.219
C	1*C*	1			1.42	0.192
	1*C*	3			1.17	0.178
	3*C*	3			1.14	0.181
	4*C*	3			1.03	0.194
B	–	–	*HMM*	40	1.35	0.210
C	1*C*	1			1.16	0.175
	1*C*	3			1.04	0.170
	3*C*	3			0.86	0.157
	4*C*	3			1.09	0.198
B	–	–	*HMM*	80	1.70	0.235
C	1*C*	1			1.46	0.197
	1*C*	3			1.27	0.189
	3*C*	3			**0.84**	**0.153**
	4*C*	3			1.04	0.210

Nevertheless, the *EER%* results were still quite high, so we decided that the results could be improved by training with the bkg and dev subsets together. In Tables 5.4, 5.5 and 5.6, we can see that, if we use more data to train our systems, we achieve better performance, especially on deep architectures with more than one layer. This improvement is observed for both architectures. This fact remarks on the importance of having a large amount of data to be able to train deep architectures. Moreover, we have carried out an experiment to illustrate this effect. In Figure 5.5, we show that, if we gradually increase the amount of data used for training, the results progressively improve. In addition to

Table 5.2: Experimental results on RSR2015-Part I [124] eval set, where *EER%* and *minDCF10* are shown. These male results were obtained by training only with a bkg subset and by varying the number of states of the HMM.

Architecture					**Results**	
Type	**Backbone**		**Pooling**		**Male**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	12.13	0.991
B	–	–	*HMM*	10	7.18	0.375
C	1*C*	1			2.58	0.305
	1*C*	3			2.31	0.293
	3*C*	3			2.78	0.405
	4*C*	3			3.28	0.502
B	–	–	*HMM*	20	3.00	0.220
C	1*C*	1			1.37	0.190
	1*C*	3			1.07	0.172
	3*C*	3			1.42	0.272
	4*C*	3			1.48	0.271
B	–	–	*HMM*	40	1.57	0.164
C	1*C*	1			0.98	0.150
	1*C*	3			**0.77**	**0.138**
	3*C*	3			1.01	0.209
	4*C*	3			1.27	0.253
B	–	–	*HMM*	80	1.73	0.181
C	1*C*	1			1.19	0.157
	1*C*	3			0.88	0.148
	3*C*	3			0.88	0.187
	4*C*	3			1.18	0.271

that, we can see in Figure 5.5b that the alignment mechanism makes the system more robust to the size of training data. To make this representation, we have used the alignment mechanism with 40 states. In addition, the shaded areas in Figure 5.5 depict the standard deviation of the *EER%* values obtained from each system trained three times.

Moreover, DET curves of these experiments have been depicted in Figure 5.6. These curves represent the best systems for each pooling configuration for the female+male experiments. As it can be noticed in these representations, the DET curves of systems trained with the HMMs of 40 or 80 states have similar behaviour.

For illustrative purposes, we also represent our high-dimensional supervectors in a two-dimensional space using T-Distributed Stochastic Neighbor Embedding (t-SNE) [252] which preserves distances in a small dimensional space. In Figure 5.7(a), we show this representation for the architecture that uses the global average to extract the embed-

Table 5.3: Experimental results on RSR2015-Part I [124] eval set, where *EER%* and *minDCF10* are shown. These female+male results were obtained by training only with a bkg subset and by varying the number of states of the HMM.

Architecture					Results	
Type	**Backbone**		**Pooling**		**Female+Male**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	11.70	0.989
B	–	–	*HMM*	10	6.37	0.433
C	1*C*	1			2.66	0.347
	1*C*	3			2.29	0.309
	3*C*	3			2.45	0.378
	4*C*	3			3.00	0.460
B	–	–	*HMM*	20	3.20	0.334
C	1*C*	1			1.88	0.295
	1*C*	3			1.51	0.255
	3*C*	3			1.34	0.236
	4*C*	3			1.34	0.235
B	–	–	*HMM*	40	2.03	0.289
C	1*C*	1			1.56	0.277
	1*C*	3			1.20	0.234
	3*C*	3			0.98	0.194
	4*C*	3			1.23	0.240
B	–	–	*HMM*	80	2.25	0.311
C	1*C*	1			1.67	0.283
	1*C*	3			1.43	0.260
	3*C*	3			**0.92**	**0.187**
	4*C*	3			1.17	0.249

dings, while in Figure 5.7(b) we represent the neural network supervectors of our best system. As we can see in the second system, the representation is able to cluster the examples of the same person, whereas in the first method is not able to cluster together the examples of the same person. On the other hand, in both representations, the data are auto-organized to show on one side the examples of female identities and on the other side the examples of male identities.

Furthermore, we illustrate in Figure 5.8 the same representation as in the previous figure, however in this case, we represent the embeddings and neural network supervectors of the thirty phrases of the female identities. With this depiction, we prove something that we had already observed in the previous verification experiments, as the embeddings of the global average architecture are not able to separate between the same identity with

Table 5.4: Experimental results on RSR2015-Part I [124] eval set, showing *EER%* and *minDCF10*. These female results were obtained by training with bkg+dev subsets and by varying the number of states of the HMM.

Architecture					**Results**	
Type	**Backbone**		**Pooling**		**Female**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	9.11	0.959
B	–	–	*HMM*	10	3.14	0.273
C	1*C*	1			1.66	0.252
	1*C*	3			1.43	0.227
	3*C*	3			1.29	0.234
	4*C*	3			1.56	0.285
B	–	–	*HMM*	20	1.81	0.219
C	1*C*	1			1.30	0.181
	1*C*	3			1.15	0.168
	3*C*	3			0.83	0.133
	4*C*	3			0.99	0.189
B	–	–	*HMM*	40	1.35	0.210
C	1*C*	1			1.17	0.182
	1*C*	3			1.07	0.152
	3*C*	3			**0.59**	**0.105**
	4*C*	3			0.74	0.158
B	–	–	*HMM*	80	1.70	0.235
C	1*C*	1			1.32	0.216
	1*C*	3			1.17	0.178
	3*C*	3			0.65	0.111
	4*C*	3			0.80	0.165

different phrase and the same identity with the same phrase which is the base of the text-dependent speaker verification task.

RSR-Part II

The second part of these experiments evaluates the performance of the system proposed in RSR-Part II which is based on short commands with a strong overlap of the lexical content of different commands, e.g., *"Volume up"* and *"Volume down"*. The results obtained with Part II are shown in Table 5.7. In this set of experiments, the phrases are shorter, so we can see that we need fewer states to model them. However, the performance obtained with models of 10 and 20 states is similar since some phrases might require more states

Table 5.5: Experimental results on RSR2015-Part I [124] eval set, showing *EER%* and *minDCF10*. These male results were obtained by training with bkg+dev subsets and by varying the number of states of the HMM.

Architecture					Results	
Type	**Backbone**		**Pooling**		**Male**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	8.67	0.963
B	–	–	*HMM*	10	7.18	0.375
C	1*C*	1			2.30	0.265
	1*C*	3			2.01	0.256
	3*C*	3			2.08	0.287
	4*C*	3			2.41	0.383
B	–	–	*HMM*	20	3.00	0.220
C	1*C*	1			1.29	0.163
	1*C*	3			1.05	0.140
	3*C*	3			1.11	0.190
	4*C*	3			1.32	0.255
B	–	–	*HMM*	40	1.57	0.164
C	1*C*	1			0.98	0.144
	1*C*	3			0.78	**0.118**
	3*C*	3			0.71	0.157
	4*C*	3			0.89	0.215
B	–	–	*HMM*	80	1.73	0.181
C	1*C*	1			1.00	0.137
	1*C*	3			0.78	0.129
	3*C*	3			**0.61**	0.154
	4*C*	3			0.83	0.198

than others. For example, phrases like *"Turn on coffee machine"* are best modelled with 20 states, while others like *"Call sister"* only need a 10 state model to be characterized.

Furthermore, as we expected, the overall performance is worse, as the system suffers from the lexical similarity of short commands. Thus, this part is more challenging than Part I, as we can also see from other previous works [155–157].

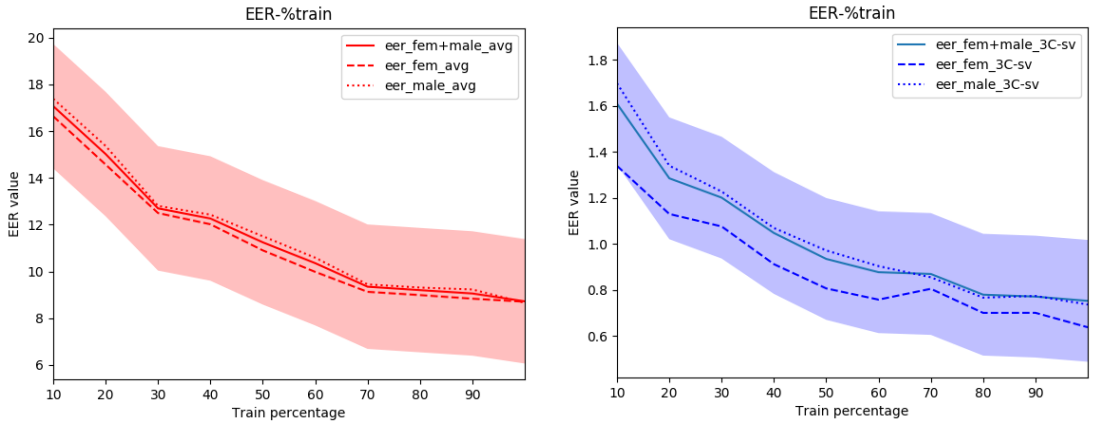
In addition to the above table, Figure 5.9 depicts the DET curves. These representations show the performance for the female+male experiments with the best systems for each pooling configuration. We can observe that the results with the HMMs of 10 or 20 states are practically the same.

Table 5.6: Experimental results on RSR2015-Part I [124] eval set, showing *EER%* and *minDCF10*. These female+male results were obtained by training with bkg+dev subsets and by varying the number of states of the HMM.

Architecture					Results	
Type	Backbone		Pooling		Female+Male	
	Layers	Kernel	Type	States	EER%	minDCF10
A	$3C$	3	avg	–	8.88	0.963
B	–	–	HMM	10	6.37	0.433
C	$1C$	1			2.49	0.335
	$1C$	3			2.13	0.298
	$3C$	3			1.82	0.275
	$4C$	3			2.12	0.345
B	–	–	HMM	20	3.20	0.334
C	$1C$	1			1.79	0.279
	$1C$	3			1.51	0.239
	$3C$	3			1.03	0.179
	$4C$	3			1.22	0.238
B	–	–	HMM	40	2.03	0.289
C	$1C$	1			1.55	0.267
	$1C$	3			1.24	0.225
	$3C$	3			0.73	**0.142**
	$4C$	3			0.86	0.195
B	–	–	HMM	80	2.25	0.311
C	$1C$	1			1.60	0.297
	$1C$	3			1.34	0.252
	$3C$	3			**0.68**	0.144
	$4C$	3			0.84	0.187

5.4.3 Comparing Global Average Pooling with GMM as Alignment Pooling Mechanism with RSR2015-Part I and Part II

As the second set of experiments, we compare architecture A that makes use of a global average pooling (avg) [165, 166], architecture B which applies the GMM alignment to the feature input directly, and the GMM alignment combined with the backbone network using architecture C. We have employed for the backbone network of architecture C, a convolutional network with three layers (CNN) and a kernel of dimension 3 which gave us the best performance in the previous section for the HMM alignment mechanism and is described in Table A.5 in Appendix A.



(a) Average embeddings (b) Neural Network Supervectors

Figure 5.5: Results of *EER%* varying train percentage where standard deviation is shown only for both gender independent results. (**a**) average embeddings; (**b**) neural network supervectors.

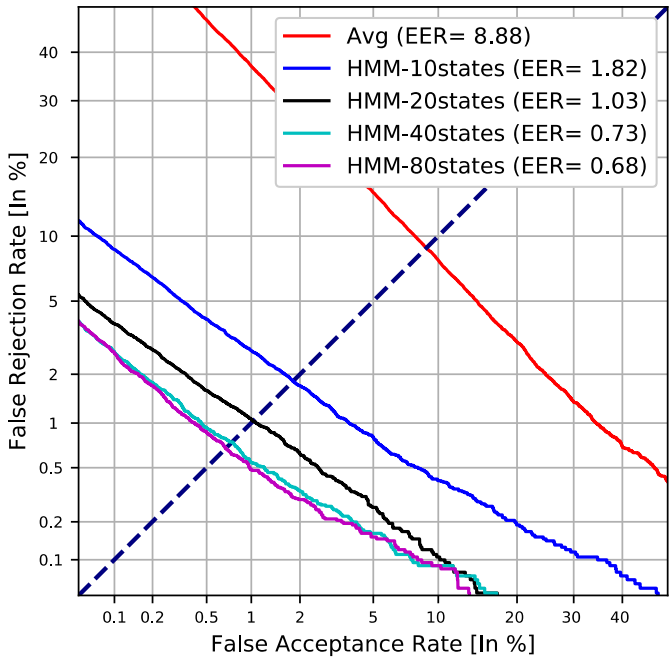
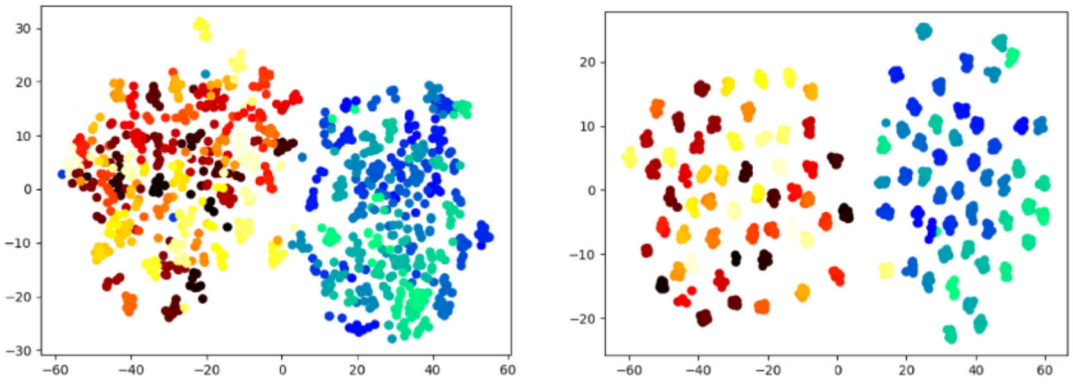


Figure 5.6: Detection Error Trade-off (DET) curve for female+male results on RSR2015-Part I of the best systems for each pooling configuration.

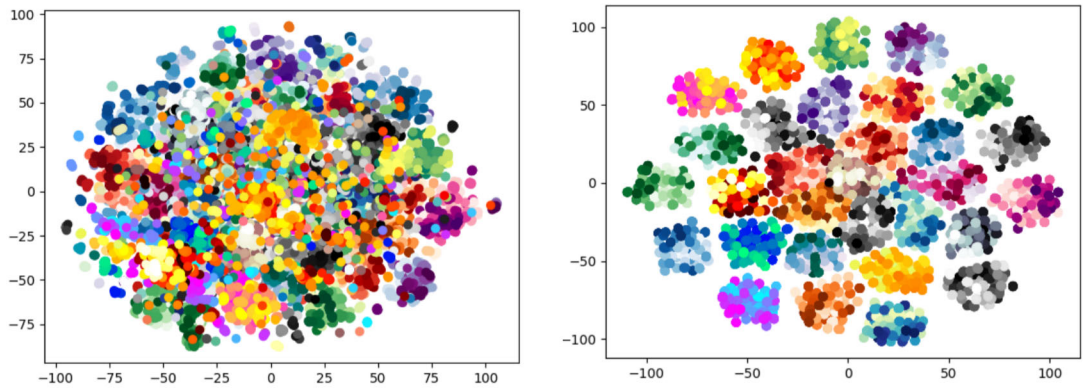
RSR-Part I

In Table 5.8, we can see *EER%*, *AUC%*, *actDCF10*, *minDCF10*, *CLLR*, and *minCLLR* results for these experiments. To obtain the *actDCF10* and *CLLR* results, we have carried out a calibration step for which we have employed the dev subset. As we showed, the embeddings of the global average reduction mechanism do not correctly represent the relevant speaker and phrase information due to the importance of keeping the temporal structure of phrases in the neural network supervectors with the alignment layer within the DNN architecture. We can also observe that architecture *C* which combines a backbone net-



(a) Average embeddings (b) Neural Network Supervectors

Figure 5.7: Visualizing Mean embeddings vs Neural Network Supervectors for 1 phrase from male+female using t-SNE, where female is marked by cold color scale and male is marked by hot color scale.



(a) Average embeddings (b) Neural Network Supervectors

Figure 5.8: Visualizing Mean embeddings vs Neural Network Supervectors for 30 phrases from female using t-SNE. Each phrase is marked by one different color scale.

work with the alignment mechanisms improves the performance achieved by applying only the alignment mechanism as in architecture *B*. In addition to the previous table, we depict the DET curves in Figure 5.10. These curves show the results for the female+male experiments. These representations demonstrate that the approach with the best system performance is architecture *C* with the backbone combined with GMM with MAP as the alignment mechanism.

Table 5.7: Experimental results on RSR2015-Part II [124] eval set, showing *EER%* and *minDCF10*. These results were obtained by training with bkg+dev subsets and by varying the number of states of the HMM.

Architecture					Results	
Type	**Backbone**		**Pooling**		**Female**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	11.18	0.982
B	–	–	*HMM*	10	7.99	0.706
C	1*C*	1			4.76	0.590
	1*C*	3			4.16	0.547
	3*C*	3			3.91	**0.540**
	4*C*	3			4.82	0.642
B	–	–	*HMM*	20	10.06	0.745
C	1*C*	1			6.44	0.671
	1*C*	3			5.57	0.618
	3*C*	3			**3.90**	0.542
	4*C*	3			5.00	0.654
B	–	–	*HMM*	40	18.13	0.820
C	1*C*	1			11.60	0.800
	1*C*	3			9.52	0.769
	3*C*	3			5.55	0.646
	4*C*	3			7.56	0.761

Architecture					Results	
Type	**Backbone**		**Pooling**		**Male**	
	Layers	**Kernel**	**Type**	**States**	**EER%**	**minDCF10**
A	3*C*	3	*avg*	–	11.81	0.981
B	–	–	*HMM*	10	9.47	0.674
C	1*C*	1			4.70	0.556
	1*C*	3			**4.31**	**0.539**
	3*C*	3			4.97	0.654
	4*C*	3			6.13	0.709
B	–	–	*HMM*	20	10.01	0.731
C	1*C*	1			5.56	0.601
	1*C*	3			5.01	0.555
	3*C*	3			4.86	0.645
	4*C*	3			6.23	0.721
B	–	–	*HMM*	40	18.16	0.815
C	1*C*	1			10.53	0.750
	1*C*	3			8.83	0.704
	3*C*	3			6.76	0.683
	4*C*	3			9.52	0.815

Architecture					Results	
Type	Backbone		Pooling		Female+Male	
	Layers	Kernel	Type	States	EER%	minDCF10
A	3*C*	3	*avg*	–	11.59	0.981
B	–	–	*HMM*	10	9.79	0.781
C	1*C*	1			5.47	0.649
	1*C*	3			4.94	0.614
	3*C*	3			4.68	0.615
	4*C*	3			5.73	0.688
B	–	–	*HMM*	20	11.06	0.803
C	1*C*	1			7.02	0.740
	1*C*	3			6.17	0.687
	3*C*	3			**4.61**	**0.607**
	4*C*	3			5.88	0.696
B	–	–	*HMM*	40	18.43	0.856
C	1*C*	1			11.74	0.847
	1*C*	3			9.95	0.819
	3*C*	3			6.59	0.679
	4*C*	3			8.93	0.794

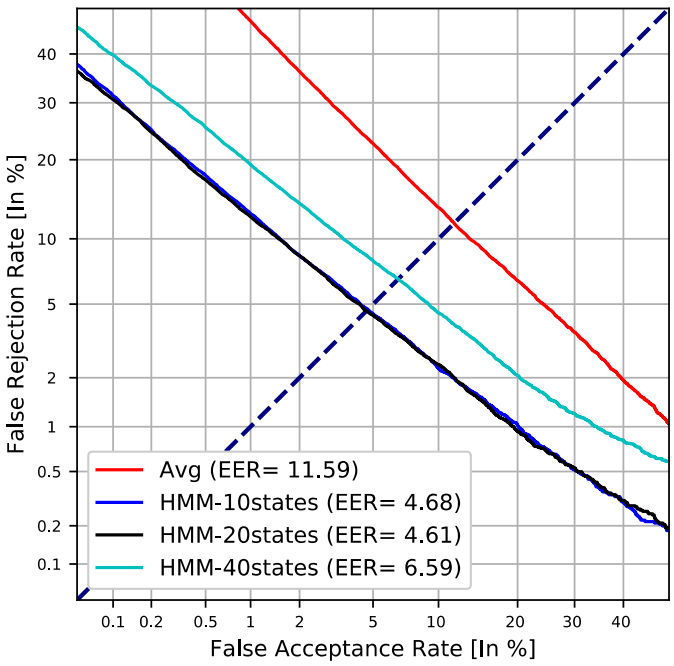


Figure 5.9: Detection Error Trade-off (DET) curve for female+male results on RSR2015-Part II of the best systems for each pooling configuration.

Table 5.8: Experimental results on RSR2015-Part I [124] eval set, showing *AUC%*, *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the global average pooling networks and the neural networks with GMM alignment technique.

Architecture				Female			
Type	**BB**	**Pool.**	**BE**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
A	*CNN*	*avg*	*CE*	9.11	97.04	0.959/1.744	0.315/0.404
B	–	*GMM*	–	1.44	99.79	0.240/0.276	0.065/0.085
C	*CNN*	*GMM*	*CE*	**0.79**	**99.95**	**0.174/0.210**	**0.032/0.041**

Architecture				Male			
Type	**BB**	**Pool.**	**BE**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
A	*CNN*	*avg*	*CE*	8.67	97.22	0.963/2.978	0.306/0.387
B	–	*GMM*	–	1.55	99.77	0.244/0.301	0.062/0.130
C	*CNN*	*GMM*	*CE*	**0.99**	**99.94**	**0.227/0.269**	**0.040/0.042**

Architecture				Female+Male			
Type	**BB**	**Pool.**	**BE**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
A	*CNN*	*avg*	*CE*	8.88	97.14	0.963/2.581	0.311/0.393
B	–	*GMM*	–	1.74	99.77	0.283/0.302	0.067/0.109
C	*CNN*	*GMM*	*CE*	**0.92**	**99.94**	**0.204/0.271**	**0.037/0.040**

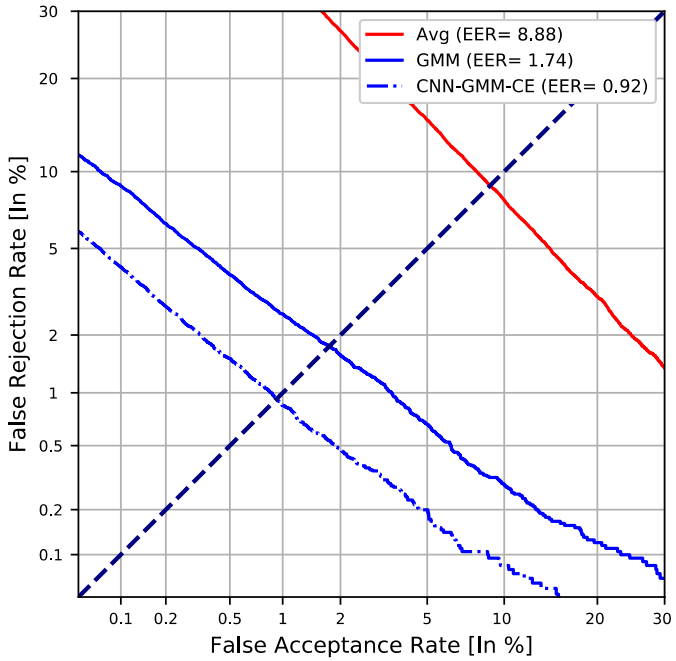


Figure 5.10: DET curves for female+male results on RSR2015-Part I of the different approaches employed for the backbone and alignment mechanism.

RSR-Part II

The results obtained with Part II are shown in Table 5.9 and Figure 5.11. In this second part, phrases are shorter with overlapped lexical content of short commands such as *"Call brother"* or *"Call sister"*. As we expected, the overall performance is worse since the system suffers from the lexical similarity of the different phrases. Therefore, this part is

more challenging than the first part, as we can also see in previous experiments. However, the system performance achieved with all architectures follows the similar trend to the results of Part I, and the best system is also formed by architecture *C* with an alignment mechanism based on GMM and MAP.

Table 5.9: Experimental results on RSR2015-Part II [124] eval set, showing *AUC%*, *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the global average pooling networks and the neural networks with GMM alignment technique.

Architecture				Female			
Type	**BB**	**Pool.**	**BE**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
A	*CNN*	*avg*	*CE*	11.18	95.57	0.982/5.772	0.386/0.570
B	–	*GMM*	–	5.91	98.40	0.734/0.784	0.221/0.238
C	*CNN*	*GMM*	*CE*	**3.86**	**99.28**	**0.652/0.668**	**0.149/0.157**

Architecture				Male			
Type	**BB**	**Pool.**	**BE**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
A	*CNN*	*avg*	*CE*	11.81	95.02	0.981/4.131	0.409/0.664
B	–	*GMM*	–	5.74	98.52	0.738/0.786	0.214/0.239
C	*CNN*	*GMM*	*CE*	**5.53**	**98.71**	0.789/0.810	**0.202/0.225**

Architecture				Female+Male			
Type	**BB**	**Pool.**	**BE**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
A	*CNN*	*avg*	*CE*	11.59	95.26	0.981/4.826	0.400/0.618
B	–	*GMM*	–	6.35	98.26	0.770/0.820	0.236/0.254
C	*CNN*	*GMM*	*CE*	**4.86**	**98.96**	0.727/0.744	**0.181/0.198**

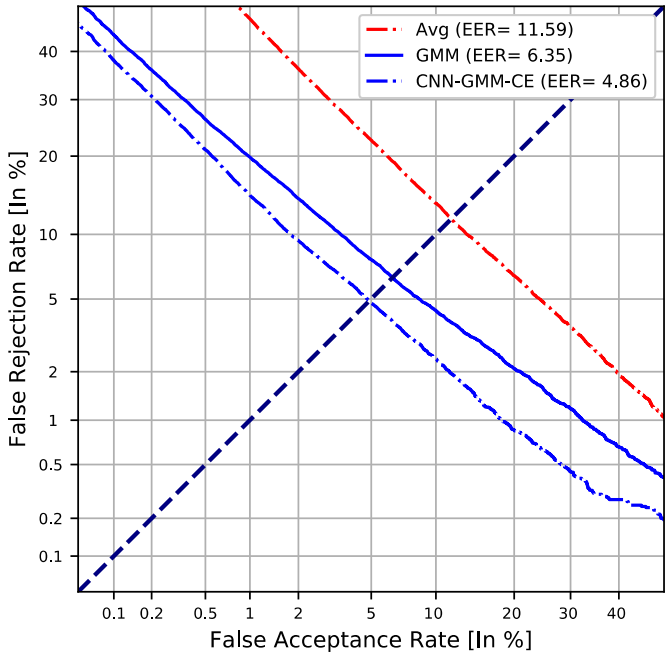


Figure 5.11: DET curve for female+male results on RSR2015-Part II of the different approaches employed for the backbone and alignment mechanism.

5.5 Conclusions

This chapter presents a new method for adding an alignment layer inside DNN architectures to encode the meaningful information of each utterance into a neural network supervector, allowing us to conserve the relevant information that we use to verify the speaker identity and the correspondence with the correct phrase. We have evaluated the models in the RSR2015-Part I and Part II text-dependent speaker verification database. The results confirm that the alignment as a layer within the architecture of DNN is an interesting approach since we have obtained competitive results with a straightforward and simple alignment technique that has low computational cost such as HMM, and with a more powerful and flexible technique such as GMM combined with MAP. Moreover, we have analyzed the effect of using different number of HMM states in funcion of the types of phrases which compose the dataset. Finally, we have also observed the improvement achieved by increasing the size of the training data.

6

Knowledge Distillation with Teacher-Student Architectures

6.1 Motivation

Apart from the issue of phonetic information presented in the previous chapter, another relevant issue is the availability of enough data for a given task. The impact of this other relevant problem could also be seen in Chapter 5 [1–3] with the experiments in which we trained using more data with respect to the initial experiments. As aforementioned, for face verification, language recognition and text-independent speaker verification, the amount of data to train the deep learning systems is very large. Whereas in text-dependent speaker verification tasks, there are large databases [164] but the amount

of publicly available training data is limited. For this reason, when a Deep Neural Network (DNN) similar to the approaches used in text-independent speaker verification task is trained using these smaller datasets, this kind of training leads to produce overconfident predictions, and is not able to generate enough discriminative representations for the new data.

Motivated by the previous reasons, the ability to generalize properly to unseen new data during the training process is a research focus for current speaker verification systems based on DNNs. These systems are typically trained to optimize an objective function on training data such as multi-class classification [132, 135]. A training process that only considers this strategy may optimize performance on training data without being able to generalize to new data. Furthermore, when the amount of training data is limited, for instance such as the RSR2015 database [124], overconfident predictions are produced. The use of several models trained on the same data has been proposed to decrease overfitting by averaging their predictions in a manner similar to a montecarlo sampling strategy, which allows the ensemble model to approximate the uncertainty in the parameters. The general idea behind the ensemble model is shown in Figure 6.1. However, this process can be cumbersome and too computationally expensive since several intermediate models have to be stored and evaluated at inference time to obtain the final ensemble model prediction, which requires a large number of computing resources. To solve this problem, a possible alternative is the use of an architecture that follows the philosophy of the Knowledge Distillation (KD) approach [253, 254], also known as Teacher-Student architecture. This approach consists of training two networks at the same time. First, the teacher network produces soft speaker labels given by their predicted posterior probability that are used to train the second network, i.e. the student network. With this training strategy, the student network is able to better capture the variability in the predictions produced by the teacher network during training in a compact model. Thus, this architecture provides more robustness to parameter changes during training and overfitting due to the lack of data. In addition, the evaluation is more efficient since a single model needs to be computed at inference time.

Therefore, this chapter presents the modifications applied on the original KD, which are based on the approach proposed in [253], to develop a teacher-student architecture for the text-dependent speaker verification task. The architecture consists of leading the student network to produce the same predictions as the teacher network. We also introduce the use of the Random Erasing (RE) data augmentation strategy [206] to add variability to the input of the networks. This technique has improved the generalization ability of neural networks in image tasks. The combination of these proposals in our system allows better modelling the parameter uncertainty with a compact model, without the needed to average the predictions of several trained models. Hence, we achieve a more robust system and contribute to handle a potential overfitting issue.

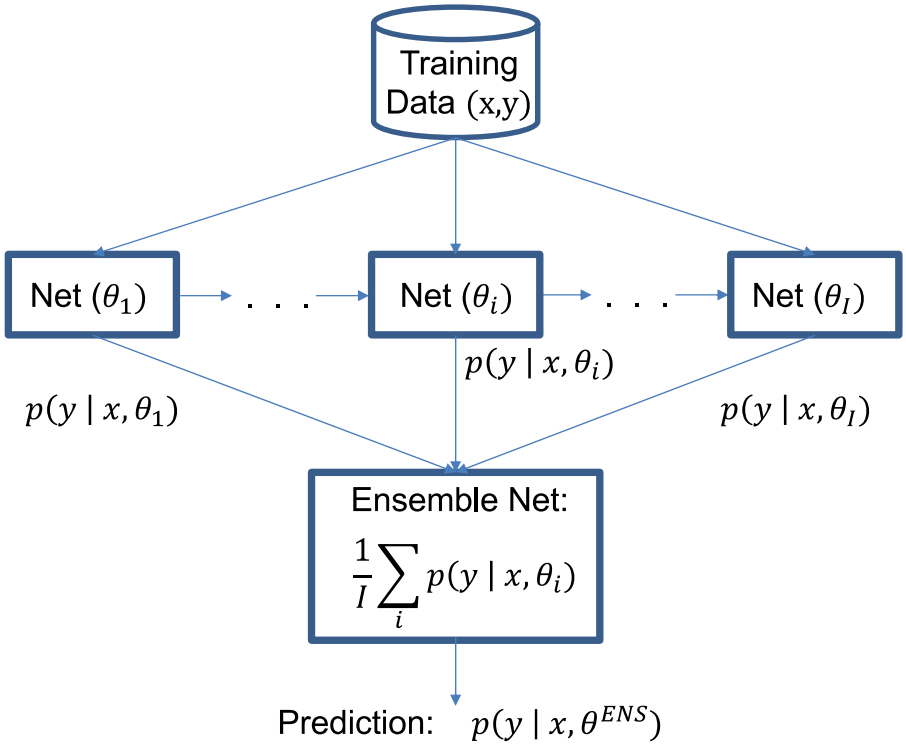


Figure 6.1: Ensemble models of several intermediate models to obtain the final prediction.

6.2 Knowledge Distillation

Knowledge Distillation (KD) approach, also called Dark Knowledge (DK), is originally a model compression framework [253], where two DNNs are implemented. These networks are often referred to as teacher and student networks. The use of the term "distillation" indicates the process made to transfer learning from the teacher network to the student network. Thus, KD can be considered as a transfer learning approach, where a teacher network is used to transfer knowledge to a student network which learns to make the same predictions. KD approaches have been successfully applied in many tasks such as image recognition [255], speech recognition [256–258], language recognition [259,260] and keyword spotting [261]. A traditional approach employing this transfer learning method consists of using a classifier output distribution instead of category labels during training in the context of model compression or distillation [262–264]. It has also been used in a domain adaptation context [265], where the source domain is employed in teacher network training, and the target domain in student network training.

With this philosophy, the teacher network produces the posterior probabilities which can be interpreted as soft speaker identity labels. These labels are used to train the student network. This second network can explore the knowledge learned by the teacher with the objective of transferring uncertainty. This learning is made possible by the use of soft labels instead of hard labels. Therefore, the student network is expected to mimic the predictions of the teacher if we drive them to produce similar posteriors. To achieve this, the Kullback-Leibler Divergence (KLD) between the student and teacher prediction

distributions is minimized. KLD measures the distance between two probability distributions, in our case $p_T(y_i|x)$ and $p_S(y_i|x)$:

$$\begin{aligned} KLD(p_T||p_S) &= \sum_{i=1}^{I} p_T(y_i|x) \cdot \log \frac{p_T(y_i|x)}{p_S(y_i|x)} \\ &= \sum_{i=1}^{I} p_T(y_i|x) \cdot \log\left(p_T(y_i|x)\right) - \sum_{i=1}^{I} p_T(y_i|x) \cdot \log\left(p_S(y_i|x)\right) \\ &= H(p_T) - CE(p_T, p_S), \end{aligned} \tag{6.1}$$

where $H(p_T)$ is the entropy which is independent of student probability distribution, so it can be considered as a constant value, and $CE(p_T, p_S)$ is the traditional Cross-Entropy (CE) loss between both distributions. Therefore, as objective function to minimize during the DNN training, KLD loss can be formulated as,

$$KLD = -\sum_{i=1}^{I}\sum_{j=1}^{J} p_T(y_i|x_j) \cdot \log\left(p_S(y_i|x_j)\right) + const, \tag{6.2}$$

where i and j are the speaker and utterance indices, x_j is the input signal, $p_T(y_i|x_j)$ is the output posterior probability of the label y_i from the teacher model (Label Distribution), $p_S(y_i|x_j)$ is the output posterior probability of the label y_i from the student network for the same example (Prediction Net), and $const$ which is defined in [254].

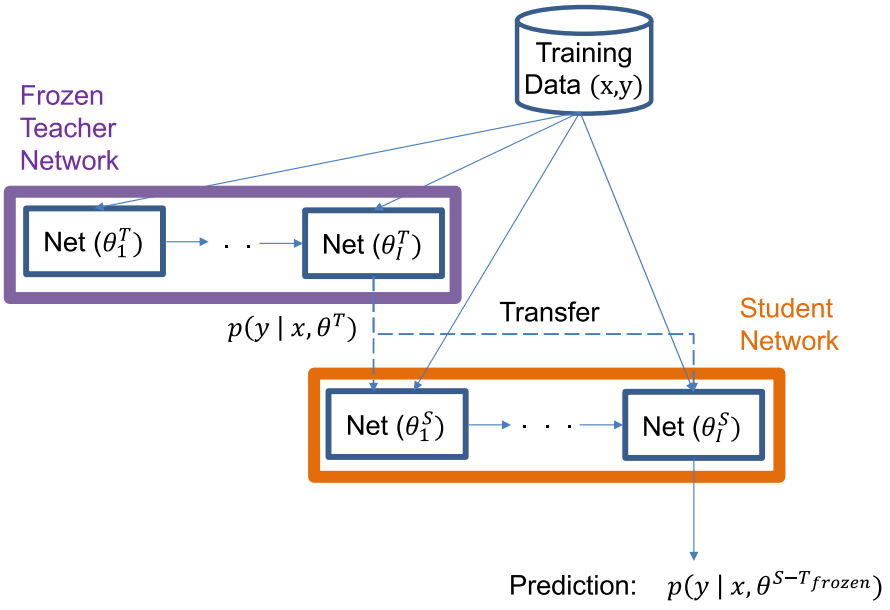


Figure 6.2: Original Knowledge Distillation approach.

Many KD methods focus on training the student network by using a pre-trained teacher model [259,264]. The weights of the teacher network are usually frozen to reduce the complexity, due to most approaches employ the same training data for both networks. Therefore, the soft speaker labels of the teacher network are fixed and are obtained from the last training step, as we can observe in Figure 6.2 which depicts the general structure

of this type of approach. Thus, the student network has to approximate the uncertainty in the model parameters from the same labels during the whole training process. The use of a frozen network also involves that part of the ability to better approximate the uncertainty in the parameters using the diversity of models as in ensemble models is lost. Nevertheless, a new version of the KD method based on a Bayesian approach (BDK) was presented in [254]. This approach showed how by applying a Bayesian estimation method, such as Stochastic Gradient Langevin Dynamics (SGLD) [266] which iteratively adds noise to the parameters and applies data augmentation to the inputs, the optimization training process of the teacher network was used to provide soft prediction labels to the student network while both were trained simultaneously and it was possible to improve the performance of the system and gets reliable probabilistic predictions.

6.3 Proposed Knowledge Distillation Approach

In this thesis, we propose an adaptation of the KD approach [253] using some of the interesting ideas in [254]. We have implemented a teacher-student architecture that allows us to provide robustness to our speaker verification system during the training process. Unlike previous works using models with frozen weights in the teacher network, we propose to train both networks at the same time which produces different prediction labels as the training progresses. The modifications introduced in the overall structure can be seen in Figure 6.3. Thus, this architecture provides robustness to the neural network training using the above strategy. Furthermore, in order to obtain more robustness, we have also substituted the classical Stochastic Gradient Descent (SGD) by the Adam optimizer [248].

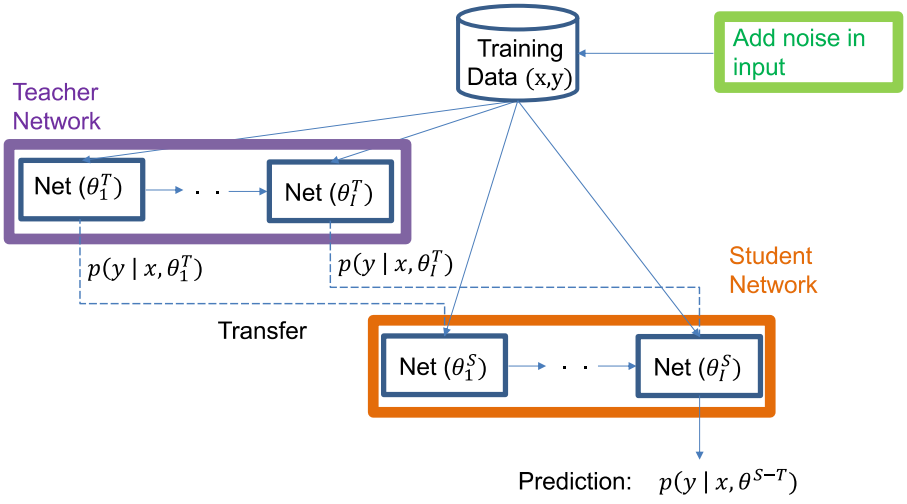


Figure 6.3: Proposed Knowledge Distillation approach.

Following the approach presented in [254], we have also used a perturbation strategy to sample data different from the training set. In this case, we have added the RE data augmentation technique explained in Chapter 4 to introduce more variability or noise in the input signals. Therefore, the general loss in (6.2) has two distributions that we

have to choose in function of these modifications. As we can see in Table 6.1, the loss of teacher network ($Loss_T$) is calculated using the true Label Distribution and the Prediction Network (i.e. the standard CE loss). For the student loss ($Loss_S$), we propose two different alternatives:

- BDK: The first alternative consists of applying the same perturbation (RE1) to the input of the student network as that used in the teacher network.
- BDK2: The second alternative adds a different perturbation (RE2) to the input of the student network. With this modification, the architecture measures uncertainty on different samples than those used to train the teacher network.

Table 6.1: Teacher and Student Losses for the two alternatives.

DK method	Loss	Label Distr.	Predic. Net
BDK	$Loss_T$	y_i	$p_T(y_i\|RE1(x_j))$
	$Loss_S$	$p_T(y_i\|RE1(x_j))$	$p_S(y_i\|RE1(x_j))$
$BDK2$	$Loss_T$	y_i	$p_T(y_i\|RE1(x_j))$
	$Loss_S$	$p_T(y_i\|RE2(x_j))$	$p_S(y_i\|RE2(x_j))$

6.4 Teacher-Student Architecture

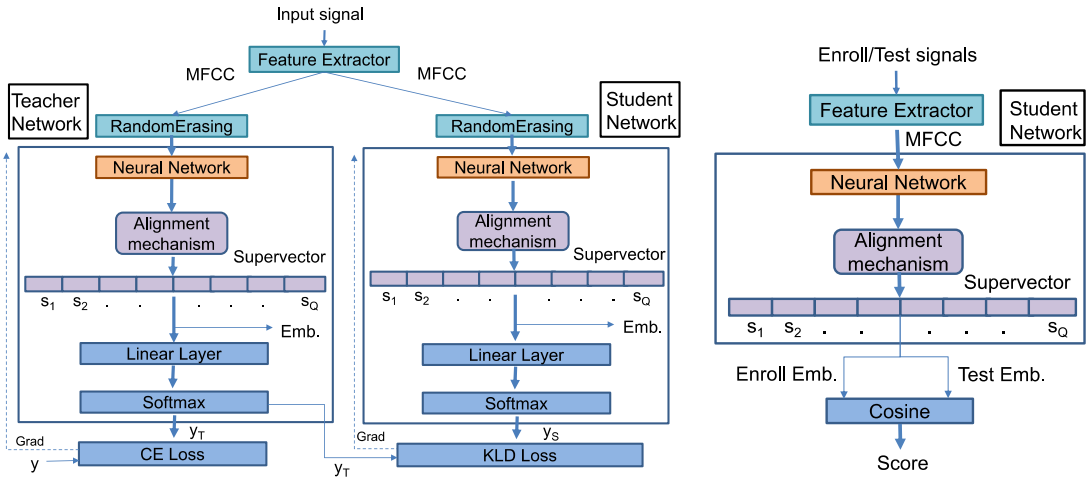
To develop the teacher and student networks, we have employed the same approaches for each part presented in Chapter 5 to create the architecture. Figure 6.4(a) represents the structure of the system used for training, our KD architecture. On the one hand, we can see how the soft predictions of the teacher network (y_T) are used to optimize CE loss and backpropagate the gradients through the teacher network. On the other hand, the same soft predictions (y_T) are employed to teach the student network to mimic these predictions using KLD loss. Hence, to train this architecture, we employ the following two loss expressions for teacher and student networks:

$$Loss_T = CE(y, y_T), \tag{6.3}$$

$$Loss_S = KLD(y_T, y_S), \tag{6.4}$$

where CE is Cross-Entropy loss, y_T is the posterior probability of the teacher network, y_S is the posterior probability of the student network, and y are the ground truth labels. In function of the two possibilities explained in Table 6.1, y_S and y_T values will be obtained using one alternative of the Label Distribution and Prediction Network or the other. Figure 6.4(b) depicts the testing phase, where only the student network without the last

layers is employed to test the system. Thus, our speaker verification system extracts the embeddings and applies a cosine similarity to perform the verification process.



(a) Training phase (b) Testing phase

Figure 6.4: (a) Left: Teacher-student training phase, where the dashed line indicates the process of backpropagation of the gradients from each network. (b) Right: Teacher-student testing phase, where the last layers are replaced by a cosine similarity to compare the embeddings extracted from the student network.

6.5 Experiments and Results

6.5.1 Experimental Setup

As the experimental setup to evaluate the improvement achieved with this architecture using RSR2015-Part I and II, we have employed a set of features composed of 20 dimensional Mel-Frequency Cepstral Coefficients (MFCCs) stacked with their first and second derivates as input to train the alignment mechanism and the DNN. For the pooling part of the architecture, two alignment mechanisms based on those developed in the previous chapter are used to create the teacher-student architecture. To train the alignment mechanisms, the bkg partition of the RSR2015 dataset has been employed. Moreover, both were trained to obtain a model per phrase without the needed to know the phrase transcription. On the one hand, HMM models have been trained using a left-to-right model of 40 states for each phrase. On the other hand, a GMM of 64 componenent has been trained per phrase. From these models, we extract the alignment information to be used in the alignment mechanism of our architecture.

In this chapter, two sets of experiments have been performed to show the benefits of using teacher-student architectures. First, we have studied the effect of introducing this architecture (*CNN (BDK)*) in case of using the different alignment mechanisms presented in Chapter 5. This architecture is described in Tables A.7 and A.8 in Appendix A. The second set of experiments has been carried out to evaluate the two different al-

ternatives for introducing the perturbation into the system. We compare an architecture that uses a single convolutional neural network (*CNN*) with the RE augmentation, the first alternative proposes to train the system using the same input for both networks in the teacher-student architecture (*CNN (BDK)*), and the other alternative uses different RE perturbations on the inputs of both networks (*CNN (BDK2)*). Additionally, we have evaluated the three systems with different layer configurations: two, three, and four convolution layers with a kernel of dimension 3.

6.5.2 Results using a Single Network versus the use of a Teacher-Student Architecture

In this first set of experiments, the basic network created in Chapter 5 is replicated using a teacher and student version, following the Bayesian Dark Knowledge (BDK) approach [253,254]. With this philosophy, a teacher network produces soft speaker identity labels that are used to train a student network. In this case, the RE data augmentation is applied to the input of the teacher network which adds variability to the labels prediction. The student network is trained using these labels, and we are able to better capture the data variability introduced. Thus, this architecture is used to provide robustness to the training of the neural network of the backbone part. Moreover, this approach has been evaluated for RSR2015-Part I and II using both HMM and GMM with MAP as the alignment mechanism.

RSR-Part I

For Part I of RSR2015, Table 6.2 contains Equal Error Rate (*EER%*), Area Under the ROC Curve (*AUC%*), actual Detection Cost Function 10 (*actDCF10*), minimum Detection Cost Function 10 (*minDCF10*), Log-Likelihood Ratio Cost (*CLLR*), and minimum Log-Likelihood Ratio Cost (*minCLLR*) results for these experiments. As we observe, the *CNN (BDK)* approach combined with the proposed frame-to-components alignment mechanism based on GMM with MAP provides an additional performance improvement. In addition to the previous table, we depict the DET curves in Figure 6.5. These curves show the results for female+male experiments. These representations demonstrate that the approach with the best system performance is the architecture with the backbone based on BDK and GMM with MAP as the alignment mechanism. Furthermore, it is worth noting that in the case of using HMM as pooling, it is better not to apply this kind of approach.

RSR-Part II

The results obtained with Part II are shown in Table 6.3 and Figure 6.6. In this second part, phrases are shorter with overlapped lexical content of short commands such as *"Call brother"* or *"Call sister"*. As we could see in Chapter 5, the overall performance is worse, since the system suffers from the lexical similarity of the different phrases. Thus, this

Table 6.2: Experimental results on RSR2015-Part I [124] eval set, showing *AUC%*, *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the different neural networks with both alignment techniques.

Architecture			Female			
BB	**Pool.**	**Loss**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
CNN	*HMM*	*CE*	**0.59**	**99.95**	**0.105/0.113**	**0.027/0.030**
CNN(BDK)			0.73	**99.95**	0.125/0.132	0.029/0.042
CNN	*GMM*	*CE*	0.80	99.95	0.174/0.210	0.032/0.041
CNN(BDK)			**0.51**	**99.98**	**0.128/0.135**	**0.021/0.025**

Architecture			Male			
BB	**Pool.**	**Loss**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
CNN	*HMM*	*CE*	**0.71**	**99.96**	0.157/0.180	**0.030/0.033**
CNN(BDK)			0.79	99.94	**0.137/0.143**	0.033/0.037
CNN	*GMM*	*CE*	0.99	99.94	0.227/0.269	0.040/0.042
CNN(BDK)			**0.78**	**99.96**	**0.149/0.215**	**0.031/0.039**

Architecture			Female+Male			
BB	**Pool.**	**Loss**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
CNN	*HMM*	*CE*	**0.73**	**99.95**	**0.142**/0.172	**0.030/0.032**
CNN(BDK)			0.80	**99.95**	0.149/**0.153**	0.033/0.036
CNN	*GMM*	*CE*	0.92	99.94	0.204/0.271	0.037/0.040
CNN(BDK)			**0.66**	**99.97**	**0.135/0.156**	**0.027/0.033**

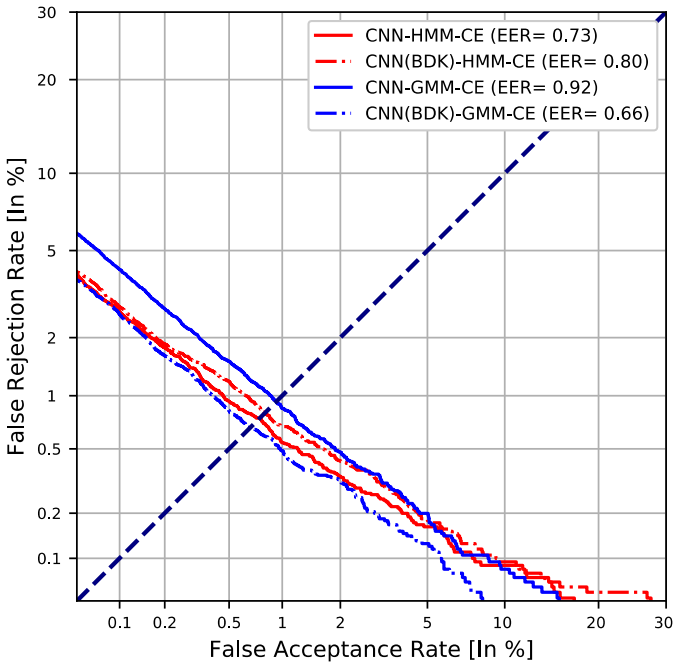


Figure 6.5: DET curve for female+male results on RSR2015-Part I of the different systems for each pooling configuration.

part is more challenging than the first part. However, the system performance achieved with all architectures follows the same trend as the results of Part I, and the best system

is also formed by the BDK architecture with an alignment mechanism based on GMM and MAP.

Table 6.3: Experimental results on RSR2015-Part II [124] eval set, showing *AUC%*, *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the different neural networks with both alignment techniques.

Architecture			**Female**			
BB	**Pool.**	**Loss**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
CNN	*HMM*	*CE*	**5.55**	**98.67**	0.646/0.821	**0.205**/**0.234**
CNN(BDK)			5.99	98.53	**0.616**/**0.624**	0.216/0.258
CNN	*GMM*	*CE*	3.86	99.28	0.652/0.668	0.149/**0.157**
CNN(BDK)			**3.15**	**99.50**	**0.519**/**0.573**	**0.121**/0.169

Architecture			**Male**			
BB	**Pool.**	**Loss**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
CNN	*HMM*	*CE*	**6.76**	**97.98**	**0.683**/**0.693**	**0.247**/**0.384**
CNN(BDK)			7.31	97.87	0.689/0.726	0.261/0.476
CNN	*GMM*	*CE*	5.53	98.71	0.789/0.810	0.202/0.225
CNN(BDK)			**4.72**	**99.00**	**0.767**/**0.785**	**0.176**/**0.205**

Architecture			**Female+Male**			
BB	**Pool.**	**Loss**	**EER%**	**AUC%**	**min/actDCF10**	**minCLLR/CLLR**
CNN	*HMM*	*CE*	**6.59**	**98.19**	0.679/0.741	**0.237**/**0.312**
CNN(BDK)			6.94	98.04	**0.671**/**0.676**	0.251/0.374
CNN	*GMM*	*CE*	4.86	98.96	0.727/0.744	0.181/0.198
CNN(BDK)			**4.07**	**99.24**	**0.666**/**0.689**	**0.154**/**0.189**

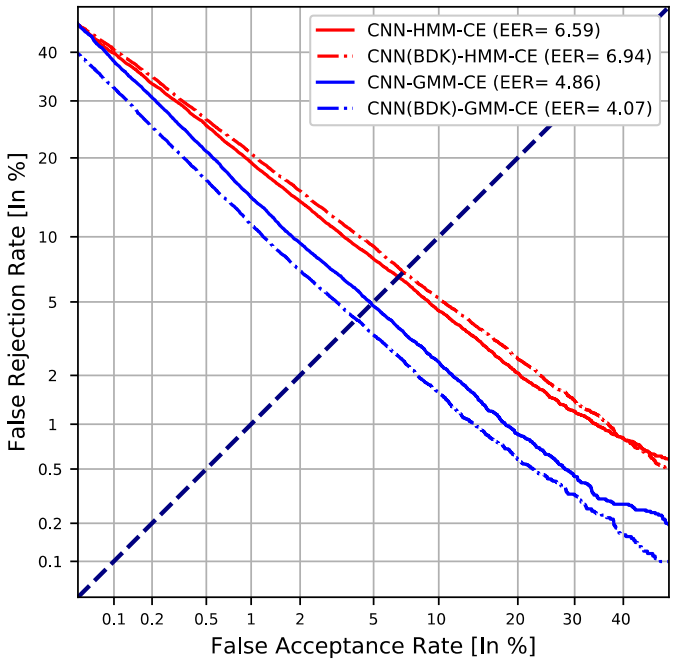


Figure 6.6: DET curve for female+male results on RSR2015-Part II of the different systems for each pooling configuration.

6.5.3 Analysis of Different Alternatives of Training Teacher-Student with RSR2015-Part I

Apart from observing the general effect of the introduction of the teacher-student philosophy as we have done in the previous section, we have also analyzed how these systems could be improved by applying some modifications in the perturbation strategy. Table 6.4 presents *EER%*, *minDCF10*, and *minCLLR* results for these experiments using GMM combined with MAP as alignment mechanism. We can observe that the proposed architectures based on KD: *CNN (BDK)* and *CNN (BDK2)*, achieve better performance than the simple *CNN* for every number of layers configuration. This improvement reflects the fact that the speaker verification system performs better if it can generalize the embedding representation to the unseen speakers that will be encountered in enrollment and evaluation. Especially relevant is the improvement achieved over the baseline with the best *CNN (BDK2)*, which uses different RE perturbations on the input of the student network as in [254] which allows to measure the uncertainty in those samples different from the training set. In terms of relative improvement, *EER%* and *minCLLR* have improved 27% and 21% respectively, and *minDCF10* is also improved.

Table 6.4: Experimental results on RSR2015-Part I [124] eval set, showing *EER%*, *CLLR* and *minDCF10*. These female results were obtained by training with bkg+dev subsets to compare the three systems with GMM+MAP as alignment mechanism and the different configuration layers for each system.

Architecture			**Results**		
BB	**T/S**	**ConvLayers**	**EER%**	**minDCF10**	**minCLLR**
CNN	–/–	2	0.59	0.138	0.024
		3	0.80	0.174	0.032
		4	1.37	0.296	0.053
CNN(BDK)	*RE1/RE1*	2	0.52	0.128	0.021
		3	0.60	0.155	0.024
		4	1.12	0.262	0.044
CNN(BDK2)	*RE1/RE2*	2	**0.43**	**0.123**	**0.019**
		3	0.59	0.126	0.024
		4	1.05	0.237	0.040

In addition to the above table, Figure 6.7 depicts DET curves representing the results of the female experiments with all system configurations. Note that both teacher-student architectures with three layers obtain similar behaviour than the single *CNN* with only two layers. Thus, these representations clearly demonstrate that systems with the teacher-student architecture have great system performance for all the configurations.

6.6 Conclusions

This chapter has presented the philosophy of Knowledge Distillation and two alternatives to introduce more variability in this approach. This kind of architecture allows

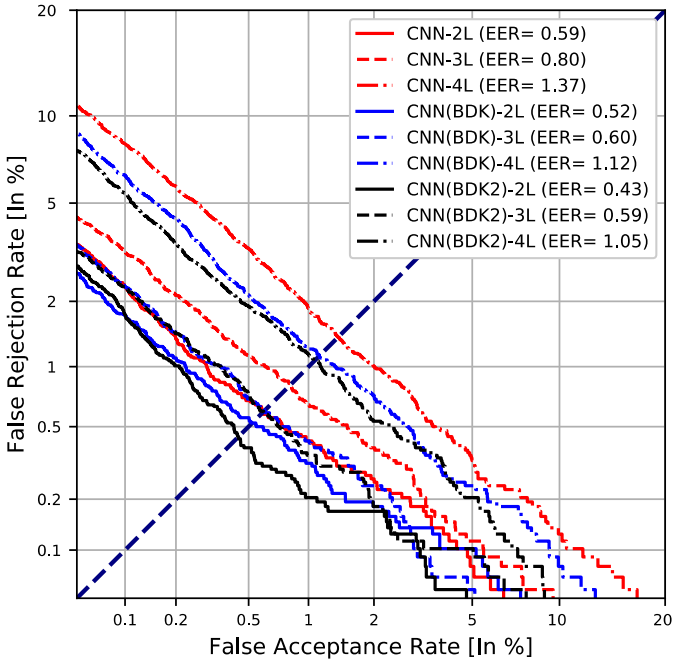


Figure 6.7: Detection Error Trade-off (DET) curve for female results on RSR2015-Part I of the three systems with GMM+MAP as alignment mechanism.

training two networks simultaneously, where the student network is trained to mimic the predictions of the teacher network, providing more robustness to the training of the system. Moreover, we have included the RE data augmentation method to introduce different variabilities to the input of the networks, which contributes to manage a potential overfitting problem in the models due to the lack of data. The proposal succeeds in generalizing and better modelling the variability introduced by the input signals. The evaluation was carried out in the RSR2015-Part I and II text-dependent speaker verification database. The results confirm that the architectures based on BDK combined with techniques that include variability to input signals are able to improve a single model in training *CNN* when a GMM with MAP is employed as alignment mechanism.

7

Multi-head Self-Attention Mechanisms with Memory Layers

7.1 Motivation

As we showed in previous chapters, keeping the order of phonetic information is important for text-dependent speaker verification tasks due to the lexical content, since this information is part of the identity. Deep Neural Network (DNN) models that use standard average pooling mechanisms to transform the processed utterance information into an embedding vector can have problems for this task. For this reason, in Chapter

5 [1–3], we presented an alternative pooling mechanism to replace the standard global average pooling. While in this chapter, we explore a different type of processing for temporal information in DNN that is provided by temporal attention mechanisms enhanced by a memory layer, which provides an efficient access to knowledge stored in the training phase. Architectures with attention mechanisms have become an effective approach in a wide variety of application areas [267, 268] to focus the processing of the DNNs on certain areas of feature maps or certain temporal slots, including the scenario of speaker verification [134, 167]. The success achieved with this approach in speaker verification may be due to the fact that this kind of mechanism allows models to learn frame-level representations, which are more precise in representing the speaker characteristics. Recently, the Multi-head Self-Attention (MSA) proposed in [268] for the Transformer architecture is a powerful attention mechanism which allows using several single attention mechanisms to extract diverse information about different parts of the network [140, 269]. These self-attention mechanisms are becoming a dominant approach in many fields beyond text-related tasks. For example, Transformers [268] are spreading to many tasks [270–273] where large scale databases are available. In speaker verification tasks, this kind of architecture has started to be successfully applied in text-independent speaker verification [140] where there are no constraints on the uttered phrase and large databases are available. However, in text-dependent speaker verification, there is still room for improvement since the amount of public data is not very large. Besides, phonetic information is relevant, so keeping the temporal structure is needed to obtain representations that correctly encode both phrase and speaker information. On the other hand, to improve the model capacity while keeping similar computational efficiency, memory layers [274] have been introduced in combination with the MSA mechanism with success for language modelling tasks. Nevertheless, this technique has not yet been applied to speaker verification tasks.

Apart from the issue of phonetic information, these techniques have had problems in applying them to text-dependent speaker verification tasks due to the lack of large public training databases to develop powerful deep learning approaches. However, two years ago, the multipurpose DeepMine database [233] was released and a challenge [234] related to this database has also been carried out. Thanks to them, a new large-scale database for text-dependent speaker verification can also be used to analyze these approaches.

In this chapter, we present an architecture based on Residual Networks (RN) [79] combined with MSA and memory layers [274] for text-dependent speaker verification tasks with small and large-scale databases. The details of the MSA layer are described in Section 7.2. On the other hand, as we will explain in Section 7.3, the memory layer is a product-key attention mechanism to store the knowledge learned by the DNN during the training process. Moreover, we have added more phonetic information to the different architectures with the use of phonetic embeddings extracted from a phonetic classification network [269, 275] which are introduced in Section 7.4. These phonetic embeddings are used as a complement to the feature extractor and can be used by the dot attention mechanism to locate phoneme similarities which can play the role of the posi-

tional embedding that is not used in our architecture. In Section 7.5, the specific blocks used to develop the architecture are presented. Finally, Section 7.6 collects and analyzes the experiments carried out and their results.

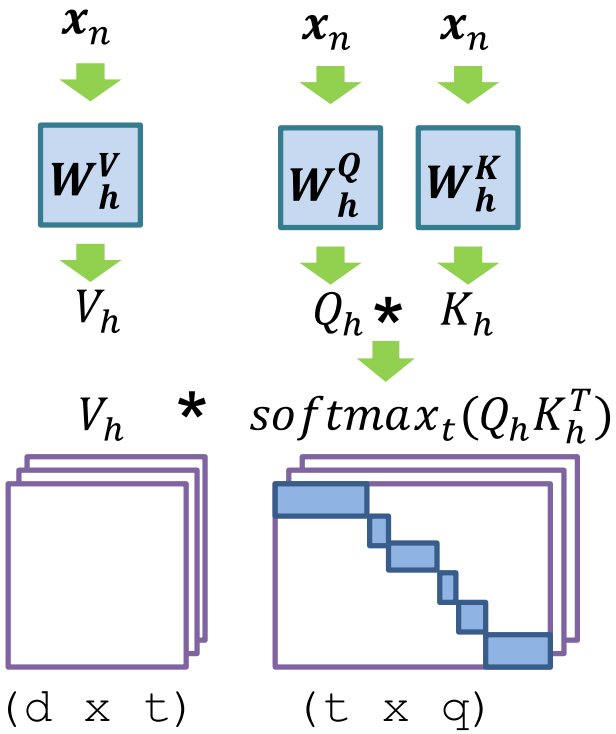


Figure 7.1: Multiple dot-product attention.

7.2 Multi-head Self-Attention Mechanism

The original transformer architecture [268] is composed of two main parts: the encoder and the decoder. However, in many representation and analysis tasks, the transformer encoder is the only part used to create deep learning systems. The core mechanism of each encoder block is the Multi-head Self-Attention (MSA) layer. Instead of performing a single attention function, this mechanism consists of a multiple dot-product attention that allows to do attention in parallel. In this thesis, we only employ the encoder part, so as we can see in Figure 7.1, the input to this attention mechanism is the same for query, key and value signals (Q_h, K_h, V_h):

$$Q_h = x \cdot W_h^Q, \; K_h = x \cdot W_h^K, \; V_h = x \cdot W_h^V, \tag{7.1}$$

where x is the input to this layer, and W_h^Q, W_h^K, W_h^V are learnable weight matrices to make the linear projections. After these projections, a softmax operation is performed on the temporal axis, which allows each head to focus on certain frames of the input sequence for each output. The result of this softmax operation is known as the self-attention matrix

for each head and can be defined as:

$$A_h = softmax_t\left(\frac{Q_h \cdot K_h^T}{\sqrt{d_k}}\right), \tag{7.2}$$

where d_k is the number of dimensions of the query/key vector. This self-attention matrix learns the most relevant information among the different data. Using this information, the value V_h feature vectors are aggregated to obtain the output of each head. The final output of each head can be calculated as,

$$H_h = V_h \cdot A_h. \tag{7.3}$$

Thus, MSA is defined as the concatenation of the outputs from each head H_h:

$$MSA(X) = [H_1, H_2 \dots H_{d^{head}}] \cdot W^{head}, \tag{7.4}$$

where X is the input to the attention layer, W^{head} is a learnable weight matrix to make a final linear projection, and d^{head} is the number of attention heads in the $h-th$ layer. When the architecture is trained with a speaker classification objective, the MSA mechanism learns to calculate a set of weights for each head that focus on different positions in the sequence and provide more relevance to the most important frames to discriminate better among speakers and utterances.

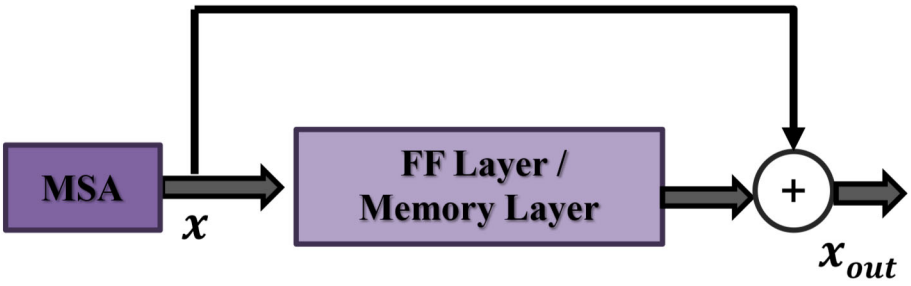


Figure 7.2: MSA layer alternates originally with a FF layer but in this thesis, we have proposed to replace with a Memory layer.

7.3 Memory Layer

The transformer encoder alternates the MSA layer with a second layer which is the feed-forward (FF) layer. However, in this thesis, we propose replacing the FF layers with memory layers as in [274]. The way to combine each of these layers with the MSA layer can be observed in Figure 7.2. The idea of using an external memory with a neural network was introduced in [276, 277]. A simplified version, which acts as a read-only memory at inference time, was used in [274] to improve the model capacity of a transformer architecture with a large external memory. This module is called memory layer and has a small computational overhead while providing significant performance improvement. In this chapter, we also use read-only memory layers that are able of storing the knowledge obtained for the network during the training process. Since the information has to be

stored while training, like any other network parameter, we need to use a differentiable mechanism to address the location. Thus, a product key-attention is used [274, 276, 277] where the closest keys to the signal enable the output of the learnable memory slots, and the output is composed of the weighted sum of the corresponding memory values of the k nearest keys.

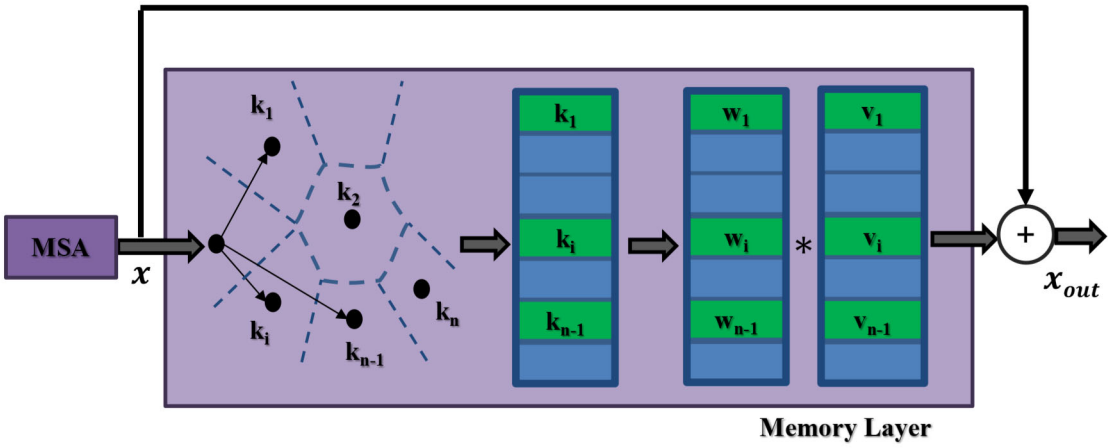


Figure 7.3: Memory layer which uses the output of the dot attention to select the closest stored values and produce a vector to add extra information to concatenate with the input.

With this layer, as shown in Figure 7.3, the input data are compared with all keys using a product key-attention, and the scores obtained are used to select the closest keys, which have the highest scores. After that, the associated weight vectors are computed with the following expression:

$$w = softmax_{n_m}(x \cdot U^K), \tag{7.5}$$

where x is the input to the memory layer, U^K is the key matrix, and the softmax is computed on the memory index axis of size n_m to focus on certain contents of the memory that will be used to provide the output. Once these vectors are obtained, these weights are combined with the memory values of the selected keys, and the output is concatenated with the previous attention output:

$$x_{out} = x + w \cdot U^V, \tag{7.6}$$

where w are the weights of the selected keys obtained with (7.5), and U^V are the memory values associated with the keys.

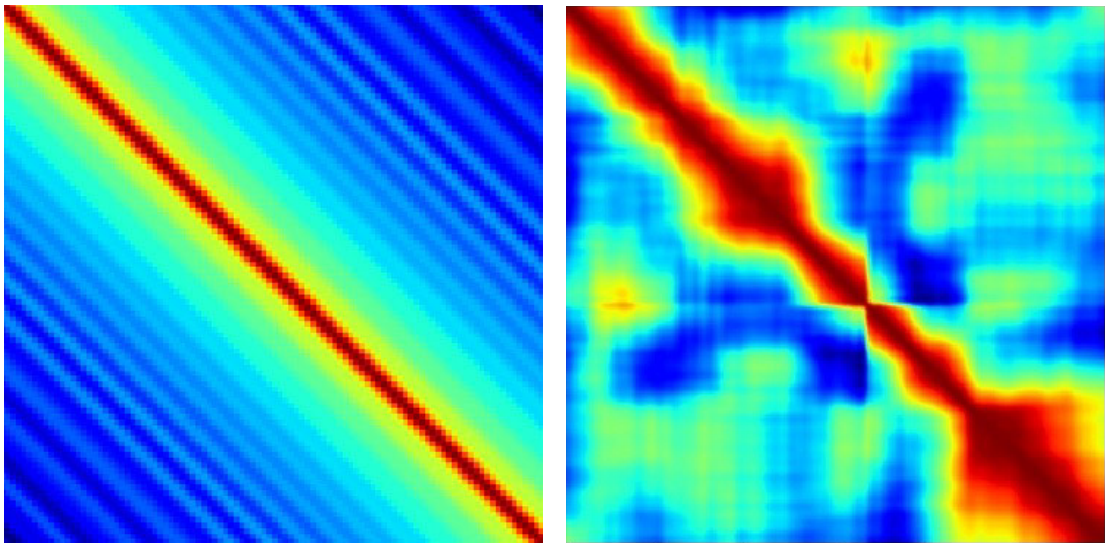
7.4 Phonetic Embeddings

Another relevant source of information employed in the transformer encoder is the positional information [268] for the MSA layers to provide good performance. This positional

information is obtained using the following expressions:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{7.7}$$

where pos is the position, and i is the dimension. Instead of using this temporal positional information like many language modelling applications, in this dissertation, we have employed the output of a phonetic classifier bottleneck [269, 275]. The architecture of the phonetic classifier is an evolution of [269]. In this system, we use a modification of the efficient net [213] to operate with 1D group convolutions as backbone. Efficient net processes the signal at several temporal scales. We combine them using a modification of [278], where we substitute the linear combinations by the operation concatenation of channels and 1D group convolution. With the integration of these phonetic embeddings into the transformer encoder architecture, the performance of the attention mechanism improves since the phonetic embeddings help to guide the attention mask to focus on the most relevant phonetic information. In Figure 7.4, examples of the cosine similarity obtained by comparing the original positional embedding [268] and the phonetic embedding with themselves are represented.



(a) Positional embeddings (b) Phonetic embeddings

Figure 7.4: (a) Left: Similarity of a positional embedding comparing with itself. (b) Right: Similarity of a phonetic embedding comparing with itself.

7.5 Residual Network Architecture combined with Multi-head Self-Attention and Memory Layers

In the following section, we describe the system architecture that combines Residual Network blocks (ResBlock), MSA and memory layers. Figure 7.5 depicts this architecture which is composed of two main parts: the backbone and pooling. Moreover, Table A.9

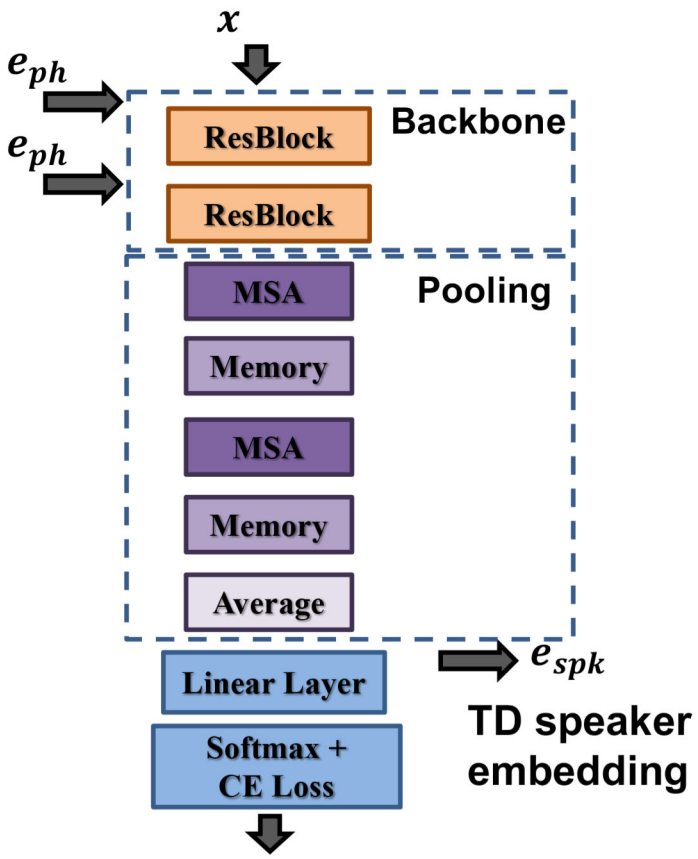


Figure 7.5: Architecture for ResBlock, MSA and Memory layers network, composed of a backbone, a pooling and a embedding extraction.

in Appendix A presents the detailed information related to the layers and dimensions of this architecture. The backbone uses two Residual blocks with three layers each block and Rectified Linear Units (ReLU) as non-linearities.

Following these Residual blocks, the pooling part alternates two MSA layers with two memory layers. The MSA layers can be seen analogous to an alignment method which allows assigning embeddings to several categories. This approach has proven useful for text-dependent speaker verification tasks [2, 3, 279]. In addition, we have introduced memory layers in our architecture that can store a significant amount of information for a relatively small inference computing cost.

After the previous blocks and layers have been applied, an average pooling mechanism is usually employed to reduce the temporal information and represent variable-length utterances with fixed-length vectors. However, this averaging may neglect the order of phonetic information, which is relevant for text-dependent speaker verification tasks.

Moreover, as we mentioned in the previous section, this type of architecture needs positional information to guide better the self-attention mechanisms of the MSA layers. As an alternative, in this thesis, we have introduced phonetic embeddings to help the

attention masks with phonetic information. To include this information, we have concatenated these phonetic embeddings before each Residual block.

7.6 Experiments and Results

7.6.1 Experimental setup

To check the power of the MSA layers used and the proposed modifications to the original components of the transformer encoder architecture, we have carried out three sets of experiments with a small (RSR2015 dataset) and a large databases (DeepMine dataset). For the experiments with DeepMine data, we have employed as input to the system a feature vector based on mel-scale filter banks. With this feature extractor, we obtain two log filter banks of sizes 24 and 32 that are concatenated with the log energy. While for the experiments with the RSR2015 database, 20 dimensional Mel-Frequency Cepstral Coefficients (MFCCs) stacked with their first and second derivates are employed as input to train the architecture. To train the systems with this database, we have only used the bkg subset. Furthermore, in both cases, we have used phonetic embeddings of 256 dimensions as positional information which are extracted from a phonetic classifier network [269]. This phonetic classification network has been trained using LibriSpeech [280] to extract phonetic embeddings. As the optimizer for the experiments in this chapter, the Adam optimizer is employed with a learning rate that increases from 10^{-3} to $5 * 10^{-3}$ during 5 epochs and then decays from $5 * 10^{-3}$ to 10^{-5}. In addition, training data is fed into the systems with a minibatch size of 32.

7.6.2 Analysis the Effect of using Positional Embeddings or Phonetic Embeddings using RSR2015-Part II and DeepMine-Part 1

In the first set of experiments, RSR2015-Part II and DeepMine-Part 1 have been employed. This initial experiment allows us to check the impact of replacing the fixed positional embeddings proposed in the original transformer work [268] by the phonetic embeddings introduced in [269]. Therefore, Tables 7.1 and 7.2 present Equal Error Rate (*EER%*), minimum Detection Cost Function 08 (*minDCF08*) and minimum Detection Cost Function 10 (*minDCF10*) results for the experiments with both dataset. We can see how a great performance improvement is achieved whether the phonetic embeddings are employed instead of the positional embeddings.

Table 7.1: Experimental results on RSR2015-Part II [124] evaluation set, showing *EER%*, *minDCF08*, and *minDCF10*. These results were obtained with bkg subset and comparing the use of positional or phonetic embeddings with the feed-forward layer in the architecture.

Architecture			Female+Male		
Embedding	**FF**	**MEM**	**EER%**	**minDCF08**	**minDCF10**
Baseline avg			11.59	0.556	0.981
Positional	yes	no	8.25	0.387	0.877
Phonetic			**5.29**	**0.255**	**0.743**

Table 7.2: Experimental results on DeepMine-Part 1 [233] evaluation set, showing *EER%*, and *minDCF08*. These results were obtained with bkg subset and comparing the use of positional or phonetic embeddings with the feed-forward layer in the architecture.

Architecture			Female+Male	
Embedding	**FF**	**MEM**	**EER%**	**minDCF08**
Baseline x-vectors* [281]			9.05	0.529
Positional	yes	no	6.30	0.245
Phonetic			**3.94**	**0.151**

7.6.3 Results with RSR2015-Part II

To carry out the second set of experiments with RSR2015-Part II, we have used, as baseline for comparison, the result obtained in [10] where a model for each phrase is trained with GMM as alignment method instead of attention mechanisms, using exactly the same data as in this work. Additionally, we compare the architecture using memory layers (MEM) with different sizes of the layer with the architecture using feed-forward layers (FF) as in the original transformer network [268].

Table 7.3 shows *EER%*, *minDCF08* and *minDCF10* results for the experiments focused on the RSR2015-Part II database. We can observe that the proposed architecture using memory layers with MSA mechanism achieves the best result. Note that regardless of the size of MEM layer, the result is better than using the original FF layer. In addition to the above metrics, in the last row of the table, we show the relative improvement achieved by comparing the architecture with the best result using the MEM layer and the architecture with the FF layer.

Furthermore, in Chapter 5, we demonstrated that approaches similar to x-vectors do not work correctly with the RSR2015 database due to small size of the training data, and the lack of special treatment of phonetic information. For this reason, we used an alignment mechanism and trained a model for each phrase in the baseline system [10]. However, in this chapter, we can check that a competitive result can be obtained with a single network for all phrases thanks to the architecture based on MSA with phonetic embeddings that allow a precise handling of phonetic information, which is one of the key points in text-dependent speaker verification. As we can see from the results, the

Table 7.3: Experimental results on RSR2015-Part II [124] evaluation set, showing *EER%*, *minDCF08*, and *minDCF10*. These results were obtained with bkg subset and varying the use of feed-forward layer or memory layer in the architecture, and the sizes of memory layer.

Architecture			**Female**		
FF	**MEM**	**Size**	**EER%**	**minDCF08**	**minDCF10**
Baseline* [10]			**4.19**	0.226	0.724
yes	no	-	4.72	0.233	0.697
no	yes	2^{11}	4.39	**0.218**	0.680
		2^{12}	4.64	0.228	**0.669**
		2^{13}	4.60	0.228	0.713
MEM vs FF Improv. %			**6.99**	**6.43**	**4.02**

Architecture			**Male**		
FF	**MEM**	**Size**	**EER%**	**minDCF08**	**minDCF10**
Baseline* [10]			5.80	0.316	0.911
yes	no	-	5.69	0.268	0.774
no	yes	2^{11}	5.32	0.255	0.716
		2^{12}	**4.92**	**0.244**	0.716
		2^{13}	5.29	0.255	**0.712**
MEM vs FF Improv. %			**13.53**	**8.95**	**8.01**

Architecture			**Female+Male**		
FF	**MEM**	**Size**	**EER%**	**minDCF08**	**minDCF10**
Baseline* [10]			5.10	0.276	0.845
yes	no	-	5.29	0.255	0.743
no	yes	2^{11}	4.88	0.238	**0.700**
		2^{12}	**4.80**	**0.237**	0.706
		2^{13}	4.99	0.245	0.721
MEM vs FF Improv. %			**9.26**	**7.05**	**5.78**

memory layer enhances the performance of the attention mechanism, which confirms that the information stored during training is useful at inference time.

7.6.4 Results with DeepMine-Part 1

In the third set of experiments with DeepMine-Part 1, the baseline to be compared is based on the x-vectors presented by the organizers of the SdSV Challenge [234]. Apart from

training with the mention DeepMine database, this baseline was trained with the popular VoxCeleb 1 and 2 databases [137, 282]. Nevertheless, we have not used VoxCeleb 1 and 2 datasets [137, 282] in the neural network training process. Motivated by the fact that in some situations and applications is required the implementation of custom systems with the few available in domain-data. For this reason, we have developed systems only with the in-domain data. Moreover, as in the first set, we compare the architecture using memory layers (MEM) with the architecture with feed-forward layers (FF).

The results obtained in the DeepMine-Part 1 database are shown in Table 7.4. Unlike previous text-dependent speaker verification works, in these experiments where the training data available in this database is larger, we observe that using a model trained with all phrases performs better. Moreover, we can see that the architecture with different sizes of MEM layers achieves better result than the architecture with FF layers, following the same trend as the other database. In Table 7.4, we can also see that both architectures outperform the baseline with x-vectors. Therefore, the importance of phonetic information combined with temporal attention using MEM layers and MSA to train DNN architectures on text-dependent speaker verification is again demonstrated.

Table 7.4: Experimental results on DeepMine-Part 1 [233] evaluation set, showing *EER%*, and *minDCF08*. These results were obtained with train set and varying the use of feed-forward layer or memory layer in the architecture, and the sizes of memory layer.

Architecture			**Female+Male**	
FF	**MEM**	**Size**	**EER%**	**minDCF08**
Baseline x-vectors* [281]			9.05	0.529
yes	no	-	3.94	0.151
no	yes	2^{11}	3.70	0.143
		2^{12}	**3.58**	**0.136**
		2^{13}	3.62	0.137
MEM vs FF Improv. %			**9.14**	**9.93**

7.7 Conclusions

Along this chapter, we have introduced a new architecture for the text-dependent speaker verification task. This kind of architecture allows us to take advantage of the knowledge acquired by temporal attention mechanisms to keep the phonetic information. Moreover, the use of memory layers improves the model capacity while keeping the efficiency of the architecture based on the original transformer network. The evaluation was carried out on two text-dependent speaker verification databases to confirm the improvement achieved. The first one is RSR2015-Part II, which due to the lack of data had suffered problems with deep architectures when an alignment mechanism was not incorporated into the network, but in this work, we have shown that with this architecture is possible to achieve competitive results on this database. Using the other database, DeepMine-

Part 1, the proposed architecture outperforms the baseline with x-vectors, so we have demonstrated the relevance of keeping temporal information during training even with larger databases.

8

Class and Distillation Tokens for Multi-head Self-Attention Systems

8.1 Motivation

The advantages of replacing the traditional pooling mechanism based on averaging of temporal information by other pooling mechanisms such as the alignment mechanism or Multi-head Self-Attention (MSA) layers combined with memory layers have been shown to be effective in the previous chapters. Nevertheless, both approaches still have some problems. The neural network supervector approach presented in Chapter 5 [1–3] allows keeping the temporal structure and representing both phrase and speaker information, but the temporal alignment has to be done by an external method as a phone decoder, a Gaussian Mixture Model (GMM) or an Hidden Markov Model (HMM). Whereas the

use of MSA of Chapter 7 [5] allows the model to focus on the most relevant frames of the sequence by means of the attention combined with the phonetic embeddings to discriminate better among utterances and speakers. However, the proposed architecture based on MSA employed an average pooling mechanism to obtain the final embedding representation.

In this chapter, to address these issues, we have introduced a learnable vector known as Class Token, which is inherited from Natural Language Processing (NLP) [270], and recently, from many image recognition systems [271]. However, this approach has not yet been applied to speaker verification tasks. To introduce this vector into the system based on DNN with MSA and memory layers, the class token is concatenated to the input before the first MSA layer, and the state at the output is employed to perform class prediction. During training, temporal information is encoded into the token, and this token interacts with the whole input sequence through self-attention and learns a global description similar to a supervector approach [2,279] since the multiple heads act as slots of the supervector. A similar mechanism has also been used recently in [273]. Therefore, the average pooling mechanism is not necessary to obtain a representation. Multiple heads can encode more details about the sequence order than the average, playing the role of states and improving the results as shown in Chapters 5 and 6 [1–4] with the use of external alignment mechanisms based on HMM and GMM. In addition, the information encoded in these multiple heads can be represented and analyzed, which improves the interpretability of the results of this kind of approach. To improve the performance obtained with the class token approach, we also introduce a novel Bayesian multiple initialization sampling mechanism to reduce potential initialization problems and give more robustness against the lack of data to model predictions. Since it is a case of use in the industry to develop specific custom systems with the small in-domain datasets and this kind of approach could be a possible solution.

Moreover, this chapter contributes with another approach based on the Transformer architecture and KD [272, 283]. We propose a teacher-student approach combined with the RE data augmentation [4,206] that allows modelling the uncertainty in the parameters of a teacher model with a compact student model and getting more reliable predictions. Following the idea proposed in [272], we have also introduced the Distillation Token in the student network to replicate the predictions of the teacher network, while the class token is trained to reproduce the true label. Unlike the objective in [272], in this chapter as we also presented in Chapter 6 [3,4], the distillation process does not aim to compress the teacher model, but rather both models are trained together and the student model learns to better capture the intrinsic variability of the teacher predictions.

The rest of the chapter is organized as follows. In Section 8.2, we explain the strategy of introducing a learnable class token to obtain a global utterance descriptor associated to the concept of supervector in speaker verification and the new approach based on a Bayesian approximation to estimate the class token. Section 8.3 introduces the approach based on a teacher-student architecture with an additional token known as distillation token that is combined with the class token to provide robustness to the learned student

model. In Section 8.4, we describe the system used. Section 8.5 presents the experimental setup and explains the results achieved. Conclusions are presented in Section 8.6.

8.2 Representation using Class Token

In many NLP and computer vision tasks, the transformer architecture uses a learnable vector called Class Token (x_{CLS}), as in the original BERT model [270] or Vision Transformer (ViT) [271], instead of a global average pooling. To employ this token in the transformer encoder, the vector is concatenated to the input of the first MSA layer to perform the classification task. With this token, the self-attention is forced to capture the most relevant information with the class token to obtain a representation as a global utterance descriptor similar to the supervector approach. Instead of mixing all the information with an average pooling mechanism, the temporal structure can be kept since the attention mechanism acts as a weighted sum of the temporal tokens for each layer. The output vector is the concatenation of different head subvectors and each of them is the result of a different attention outcome. Thus, the mechanism can be seen similar to those used in Chapter 5, where the heads play the role of the states and the supervector in [279]. The supervector mechanism is also similar to [269] but in that case, the task was text-independent speaker verification and MSA layers were not used. Besides, this type of mechanism allows to enhance the interpretability of what the neural network learns through the self-attention layers as we will see in the experimental section.

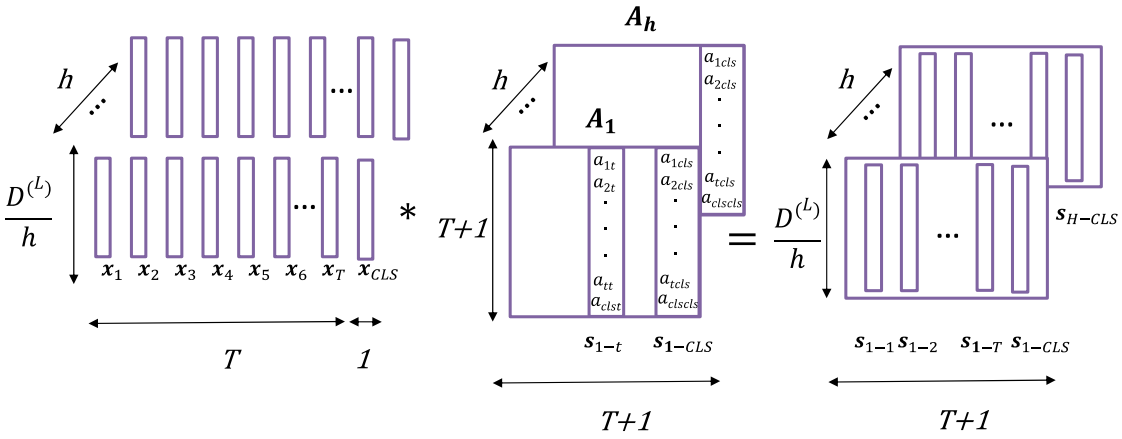


Figure 8.1: Process of alignment, the input signal x is multiplied by H alignment matrices A_H to produce H matrices with vectors s_{H-T+1} which are then concatenated to obtain the supervector.

In [279], this mechanism to obtain the supervector is defined similar to a conventional GMM supervector with the following expression:

$$s_c = \frac{\sum_t x_t \cdot a_{tc}}{\sum_t a_{tc}} = \sum_t x_t \cdot \bar{a}_{tc}, \tag{8.1}$$

where a_{tc} are the weights obtained by a softmax function on the output of a learnable layer, s_c are vectors per state/component C of dimension D that summarise the informa-

tion associated along the sequence of feature vectors x_t of dimension D, and $\bar{a}_{tc}$ are the normalized weights defined as $a_{tc}/\sum_t a_{tc}$. The final supervector is a representation concatenated of these vectors $S = (s_1, ..., s_C)$. In this chapter, the application of the MSA layer can be seen equivalent to using (8.1) for each head H in the layer, where $\bar{a}_{tc}$ corresponds to the matrix of self-attention weights of each head A_h, in particular for the class token which is in the last row of A_h would play the same role in the weighted sum. Therefore, the final class token obtained with this mechanism is the concatenation of the different head subvectors corresponding to the class token position, which can be expressed as the supervector presented previously $S_{CLS} = (s_{1-CLS}, ..., s_{H-CLS})$. This mechanism is depicted in Figure 8.1.

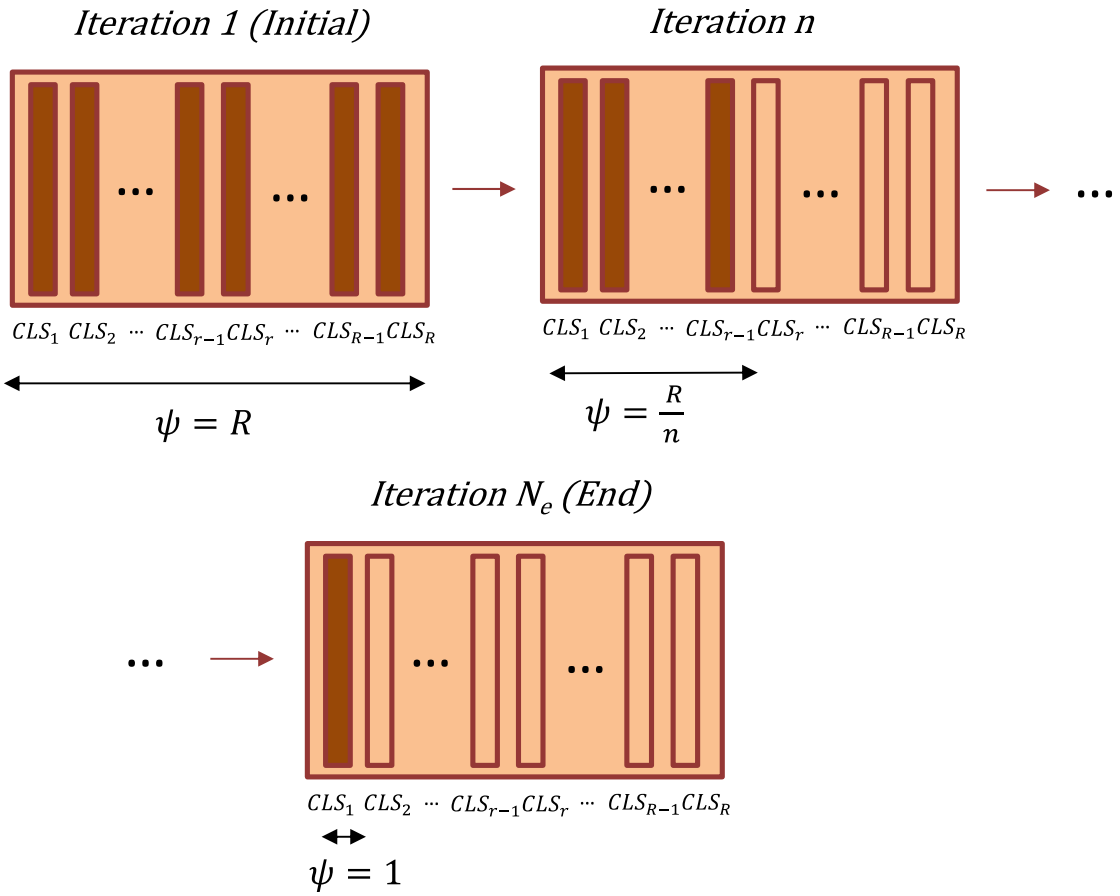


Figure 8.2: Evolution of the number of vectors in the token matrix that are available for sampling from the beginning of the training process (iteration 1) to the final iteration (iteration N_e). In each iteration, the dark vectors represent the enabled class tokens, while the light vectors are the disabled tokens.

To introduce the class token into the system, one trainable vector parameter is defined with the dimension of the feature vectors when the network is initialized. For each batch, it is replicated and concatenated at the end of each input feature sequence in the training batch as an additional token. Hence, a single shared vector is trained to learn the final embedding representation.

In this chapter, we propose the use of a Bayesian approach [284], and instead of having a single shared class token for the whole batch, we assume that this sensitive parameter is

the result of sampling from a multimodal prior distribution, a mixture of Gaussians which allows several vector modes to be selected during training by sampling them. In order to do that, we define a matrix of R vectors (*Token Matrix*) and sample it to take one of them for each example in the batch, therefore introducing uncertainty in the class token (*CLS Token*). However, the use of this approach would lead to a complex and slower evaluation process, since a Bayesian sampling inference would have to be carried out to obtain the representations. For this reason, to avoid making sampling at inference time, we have scheduled a forced reduction of the available prior components in the mixture throughout the training process. Thus, at the end of this process, only one weight is different from zero, and the class token vector parameter is fixed. This strategy allows us to start the training (*Iteration* 1) with a matrix of several initial class tokens to sample from and, gradually, we reduce the number of vectors as the training progresses to finish (*Iteration* N_e) with only one as the original class token, as Figure 8.2 depicts. Therefore, training leads the system to progressively focus the relevant information on the first vector in the matrix.

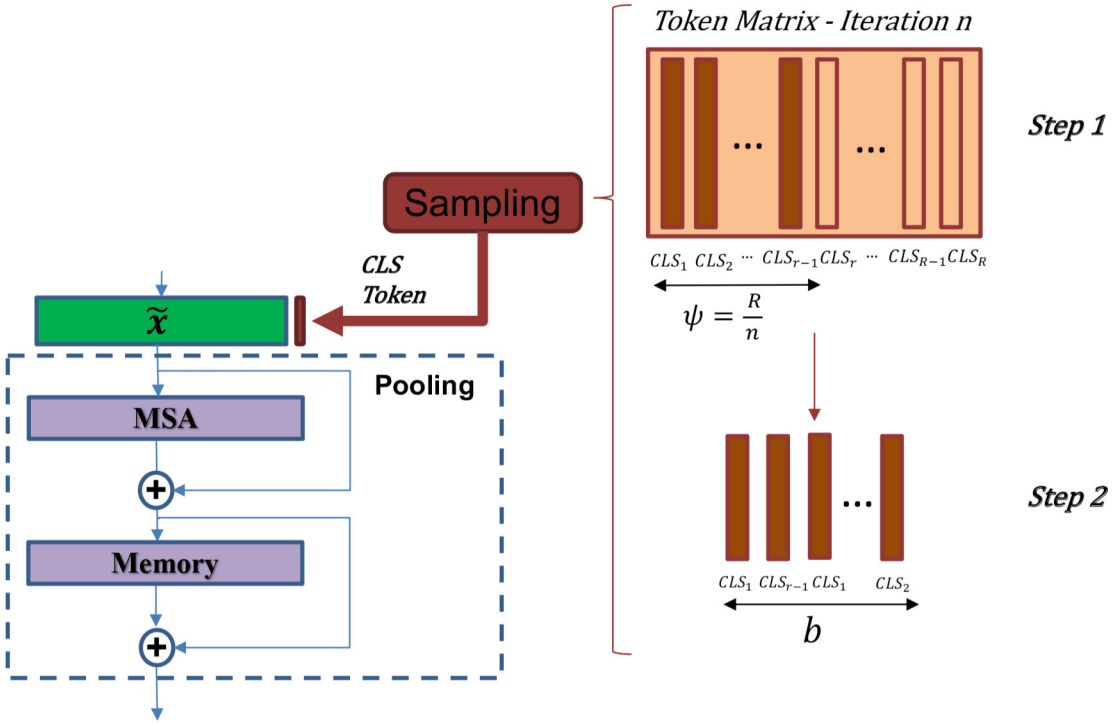


Figure 8.3: Example of the sampling steps in iteration n of the training process. In Step 1, the available vectors of the token matrix in that iteration are defined and the random indices of batch size (b) are calculated. In Step 2, the class tokens are selected and added to the input of MSA layer.

To carry out this process, we define a linear decrease scheduling using the following vector, which indicates to the neural network the number of tokens available for sampling at each iteration of the training process:

$$\psi = (\psi_1, ..., \psi_n, ..., \psi_{N_e}), \psi_n = R, ..., 1, with\ R \in \mathbb{R} \tag{8.2}$$

Algorithm 1: Algorithm for sampling class token as substitute of global average pooling and introducing it before the pooling part.

Input: Input examples $\mathcal{X}$, batch size b, samples of batch x, the number of layers L, the total number of tokens to sample R, and the number of epochs N_e

1. Define the vector with the number of sample vectors ψ available to select each epoch:

$\psi = (\psi_1, ..., \psi_n, ..., \psi_{N_e})$, $\psi_n = R, ..., 1$, with $R \in \mathbb{R}$

2. Define the random matrix of class token vectors:

$TokenMatrix = random\_matrix(R)$

for $n = 1$ **to** N_e **do**

  for $x \in \mathcal{X}$ **do**

    3. Sampling Process

    3.1 Step 1, every update b integer indexes are randomly generated from the available ψ_n vectors:

    $inds = random\_integer(\psi_n, b)$

    3.2 Step 2, the correspondent tokens are selected:

    $CLS_{Token} = TokenMatrix[inds]$

    4. Network Training

    4.1 Step 1, the class token is concatenated with the input to the MSA layer:

    $x_l = [x \| CLS_{Token}]$

    4.2 Step 2, the concatenation is the input of the first MSA layer in the pooling part and the L layers are applied:

    for $l = 1$ **to** L **do**

      $x_l^{'} = x_l + MSA(x_l)$

      $x_l = x_l^{'} + Memory(x_l^{'})$

    end

    4.3 Step 3, the state at the output of the last layer in the pooling block of the class token is used as final representation:

    $x_{CLS} = x_l^{end}$

    4.4 Step 4, the final representation is passed through a linear layer and the loss function is calculated:

    $out = Linear(x_{CLS})$

    $loss = CE(softmax(out, y)) + RL(out)$

  end

end

where R is the number of tokens defined in the matrix, and N_e is the total number of iterations for the training process. Among the number of tokens available at each iteration, a random selection of the batch size is made to select the index of the vectors.

These vectors are selected from the distribution (*Token Matrix*) and used as class tokens (*CLS Token*) in the batch to concatenate to the input of the first MSA layer. The overall process is described in Algorithm 1. Besides, Figure 8.3 shows a graphical example of how this sampling process is made in an intermediate iteration (*Iteration n*).

8.3 Knowledge Distillation with Tokens

Motivated by the benefits obtained when the training databases are not very large with the Teacher-Student architecture based on Convolutional Neural Networks (CNNs) [4] as we proved in Chapter 6, we have implemented this architecture using two transformer networks as Figure 8.4 depicts. Using a Bayesian approach similar to [254], the teacher-student architecture provides robustness to the system. In this approach, the teacher and student networks are trained at the same time, unlike previous works [259, 260] where the teacher network is usually a pre-trained model to reduce complexity and remains frozen during the training. Besides, different sources of noise are added to the input signals of both networks, so we have also employed the RE data augmentation method to provide more variability as in Chapter 6. With this kind of architecture, the student network attempts to mimic the label predictions produced by the teacher network using the class token output. However, inspired by [272], we have also included an extra learnable token in the student network which is known as Distillation Token (*Distill Token*). The introduction of this extra token allows us to implement a multi-objective optimization using the class token to reproduce the true label while the distillation token is trained to mimic the predictions of the teacher network. To achieve this, Kullback-Leibler Divergence (KLD) loss between the student and teacher distributions is minimized. As it was shown in Chapter 6, KLD loss can be formulated as,

$$KLD = -\sum_{i=1}^{I}\sum_{j=1}^{J} p_T(y_i^{cls}|x_j) \cdot \log\left(p_S(y_i^{dist}|x_j)\right) + const, \tag{8.3}$$

where i and j are the speaker and utterance indices, x_j is the input signal, $p_T(y_i^{cls}|x_j)$ is the output posterior probability of the label y_i^{cls} from the class token of the teacher model, $p_S(y_i^{dist}|x_j)$ is the output posterior probability of the label y_i^{dist} from the distillation token of the student network for the same example, and $const$ is defined in Chapter 6. Hence, to train the teacher-student architecture shown in Figure 8.4, we employ the following two loss expressions for the teacher and student networks:

$$Loss_T = CE(y_T^{cls}, y), \tag{8.4}$$

$$Loss_S = KLD(y_S^{dist}, y_T^{cls}) + CE(y_S^{cls}, y), \tag{8.5}$$

where CE is Cross-Entropy loss, y_T^{cls} is the posterior probability of the class token from the teacher network, y_S^{cls} is the posterior probability of the class token from the student

network, y_S^{dist} is the posterior probability of the distillation token from the student network, and y are the ground truth labels.

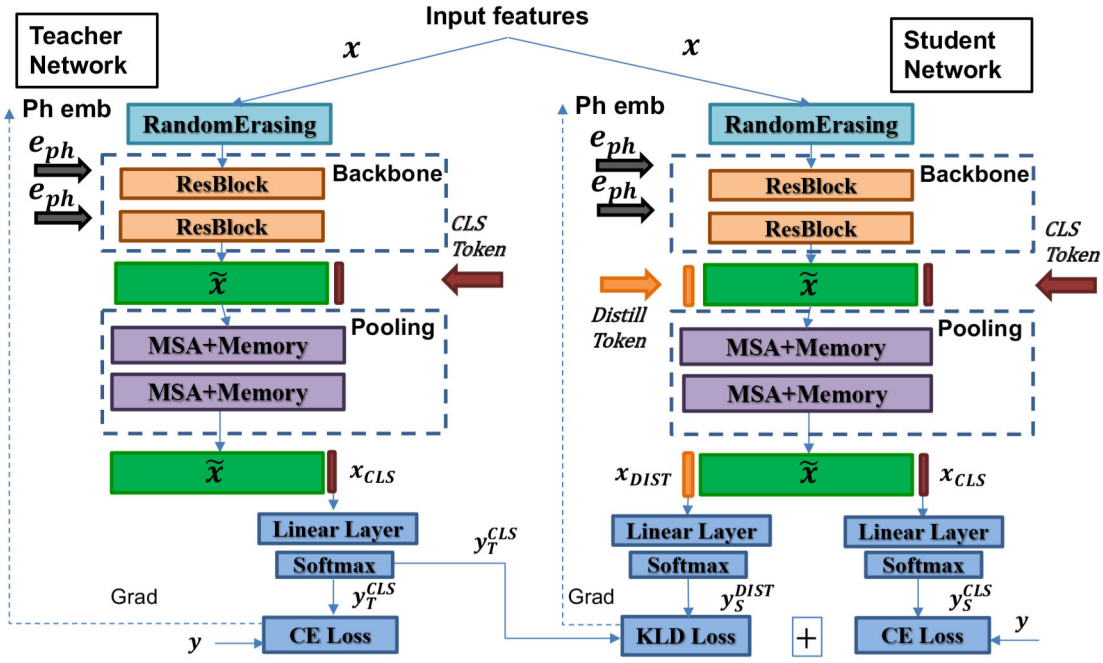


Figure 8.4: Teacher-student architecture used to create the system, where the dashed line indicates the process of backpropagation of the gradients of both loss functions. Both networks are employed to train while for testing, the student network is the only one used.

8.4 Class and Distill Tokens for Teacher-Student Architecture

In this section, we describe the new approach to teacher-student architecture developed in this chapter, which is depicted in Figure 8.4. Both architectures follow the structure described in Chapter 7 with the same backbone and pooling parts which are detailed in Tables A.10, A.11, A.12 and A.13 in Appendix A. Moreover, before the first MSA layer, the class token is concatenated with the input. In the case of the student network, the distillation token is also included. Thanks to the self-attention mechanism, these tokens learn to obtain a global representation for each utterance without applying global average pooling. These representations, similar to the neural network supervector in Chapter 5, are more convenient for the text-dependent speaker verification task. Besides, the use of memory layers increases the amount of knowledge obtained by the network that can be stored. After training the system, the class and distillation tokens are extracted as representations and a cosine similarity is applied on them to perform the verification process.

8.5 Experiments and Results

8.5.1 Experimental Setup

For the experiments, the two text-dependent speaker verification datasets presented in Chapter 3 have been employed. To carry out the experiments with the RSR2015 dataset, a set of features composed of 20 dimensional Mel-Frequency Cepstral Coefficients (MFCCs) with their derivates are used as input. While for the experiments with the DeepMine dataset, we have employed a feature vector based on mel-scale filter banks. With this feature extractor, we obtain two log filter banks of sizes 24 and 32, which are concatenated with the log energy to obtain a final input dimension of 57. Moreover, phonetic embeddings of 256 dimensions have been used as positional information as in Chapter 7. As the optimizer for the experiments in this chapter, the Adam optimizer is employed with a learning rate that increases from 10^{-3} to $5 * 10^{-3}$ during 60 epochs and then decays from $5 * 10^{-3}$ to 10^{-4}. In addition, training data is fed into the systems with a minibatch size of 32.

In this chapter, two sets of experiments have been carried out to evaluate the proposals with both databases. Different approaches to obtain the representations with a single neural network using the same architecture as the teacher network are compared: the use of the traditional global average pooling (AVG) and the introduction of the learnable class token (CLS). For the class token approach, we evaluate our proposal of sampling a mixture distribution implemented as a matrix of R vectors and reducing it until having a single vector ($Sampling$). This parameter is also swept for different values of R, including $R = 1$ which corresponds to the original idea in the bibliography [271] of having a single token and repeating it. Moreover, we analyze the effect produced by the fact of using a teacher-student architecture with an extra distillation token ($CLS - DIST$).

8.5.2 Class Token Study

A first set of experiments was performed to compare the use of a class token to obtain global utterance descriptors with the use of a global average pooling method. Thus, we study the two approaches to introduce this vector explained during this chapter and the effect of the number of vectors chosen for the sampling approach.

RSR-Part II

Table 8.1 presents Equal Error Rate (EER%), minimum Detection Cost Function 08 (*minDCF08*) and minimum Detection Cost Function 10 (*minDCF10*) results for the experiments with the RSR2015-Part II dataset. Regardless of the number of vectors in the sampling for class tokens, if we apply our proposed strategy to introduce the tokens with a sampling alternative, the obtained performance is better. In addition, the results show how employing a learnable token outperforms the use of an average embedding. Note

that the token is trained through self-attention and keeping the temporal structure to obtain a global utterance representation, while the average embedding neglects this information that is relevant to the speaker verification task. As we can also observe with the sweep of the value of R, the use of several vectors to create the token matrix is better than using a single vector ($R = 1$) and repeating it for the whole batch, which corresponds to the original way of applying this approach [271]. However, when the number of available tokens is too large, the performance begins to degrade.

Table 8.1: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF08* and *minDCF10*. These results were obtained by training with bkg subsets to compare the different approaches to obtain the representations: average or sampling strategies. The case of having a single vector and repeat it corresponds with the experiments with R=1.

Architecture			**Female**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	4.64	0.228	0.669
CLS	no	R=1	3.71	0.174	0.580
		R=50	3.37	0.169	0.580
		R=100	**3.33**	**0.158**	**0.552**
		R=200	3.55	0.171	0.562

Architecture			**Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	4.92	0.244	0.716
CLS	no	R=1	4.27	0.215	0.679
		R=50	4.04	0.199	0.601
		R=100	**3.68**	**0.182**	**0.552**
		R=200	4.09	0.199	0.607

Architecture			**Female + Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	4.79	0.237	0.706
CLS	no	R=1	4.12	0.201	0.634
		R=50	3.75	0.187	0.606
		R=100	**3.57**	**0.173**	**0.565**
		R=200	3.86	0.189	0.587

DeepMine-Part 1

In Table 8.2, the results obtained in the DeepMine-Part 1 database are shown. Unlike the other dataset, the training data in DeepMine is larger, which indicates that the lack of data is not so critical to train a powerful and robust system. Therefore, the replacement of the average embedding by a class token improves the performance only slightly. Besides, the

sweep of the value R shows that an improvement is also achieved although smaller than in the RSR-Part II results.

Table 8.2: Experimental results on DeepMine-Part 1 [233] eval subset, showing *EER%*, *minDCF08* and *minDCF10*. These results were obtained by training with train set to compare the different approaches to obtain the representations: average or sampling strategies. The case of having a single vector and repeat it corresponds with the experiments with R=1.

Architecture			**Female**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	3.92	0.135	0.411
CLS	no	R=1	3.81	0.128	0.389
		R=50	3.92	0.131	0.393
		R=100	**3.69**	**0.124**	**0.379**
		R=200	3.89	0.133	0.417

Architecture			**Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	3.02	0.137	0.676
CLS	no	R=1	3.32	0.143	0.697
		R=50	3.19	0.140	0.668
		R=100	3.09	0.137	0.658
		R=200	**2.92**	**0.133**	**0.655**

Architecture			**Female + Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	3.58	0.136	0.521
CLS	no	R=1	3.60	0.134	0.520
		R=50	3.62	0.134	0.519
		R=100	**3.43**	**0.129**	**0.505**
		R=200	3.50	0.133	0.521

8.5.3 Effect of Knowledge Distillation using Tokens

In this section, we analyze the effect of introducing an approach based on KD philosophy which consists of a teacher-student architecture. Furthermore, in this approach, an extra distillation token ($CLS - DIST$) is incorporated [272]. This approach has been employed to compare the performance obtained in the case of the global average pooling as well as in the proposed sampling approach to use the class token. In this second case, we have developed the teacher-student architecture using the R value of the best configuration obtained in the previous section, and also, the case of $R = 1$ as it is the usual way to apply this class token approach in the literature.

RSR-Part II

Results of these experiments in RSR-Part II are shown in Table 8.3. Regardless of the kind of approach to obtain the representations used, we can observe that the use of an architecture based on a teacher-student approach improves the robustness and achieves better performance in all the alternatives to extract the representations. Moreover, the best performance is obtained by applying our proposed strategy to introduce the tokens with a sampling alternative with more than a single vector, especially if it is combined with the distillation token representation.

Table 8.3: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF08* and *minDCF10*. These results were obtained by training with bkg subset to compare the use of a teacher-student architecture for the different approaches to obtain the representations: average or sampling strategies. The case of having a single vector and repeat it corresponds with the experiments with R=1.

Architecture			**Female**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	4.64	0.228	0.669
	yes	–	3.52	0.170	0.587
CLS	no	R=1	3.71	0.174	0.580
CLS-DIST	yes	R=1	3.01	0.148	0.548
CLS	no	R=100	3.33	0.158	0.552
CLS-DIST	yes	R=100	**2.47**	**0.122**	**0.414**

Architecture			**Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	4.92	0.244	0.716
	yes	–	3.78	0.186	0.579
CLS	no	R=1	4.27	0.215	0.679
CLS-DIST	yes	R=1	3.40	0.173	0.557
CLS	no	R=100	3.68	0.182	0.552
CLS-DIST	yes	R=100	**2.83**	**0.138**	**0.463**

Architecture			**Female + Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	4.79	0.237	0.706
	yes	–	3.74	0.185	0.602
CLS	no	R=1	4.12	0.201	0.634
CLS-DIST	yes	R=1	3.31	0.167	0.558
CLS	no	R=100	3.57	0.173	0.565
CLS-DIST	yes	R=100	**2.68**	**0.133**	**0.443**

DeepMine-Part 1

On the other hand, Table 8.4 presents the performance of systems with DeepMine-Part 1. In this case, the results show that the application of only the teacher-student architecture does not improve the systems. However, the use of the teacher-student architecture and the extra distillation token ($CLS-DIST$), combined with the sampling strategy with several token vectors also allows achieving a more robust system and a significant improvement in the results. The best results are again obtained when the class and distillation token representations are used.

Table 8.4: Experimental results on DeepMine-Part 1 [233] eval subset, showing *EER%*, *minDCF08* and *minDCF10*. These results were obtained by training with train set to compare the use of a teacher-student architecture for the different approaches to obtain the representations: average or sampling strategies. The case of having a single vector and repeat it corresponds with the experiments with R=1.

Architecture			**Female**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	3.92	0.135	0.411
	yes	–	4.07	0.135	0.401
CLS	no	R=1	3.81	0.128	0.389
CLS-DIST	yes	R=1	3.80	0.131	0.395
CLS	no	R=100	3.69	0.124	**0.379**
CLS-DIST	*yes*	R=100	**3.51**	**0.122**	0.385

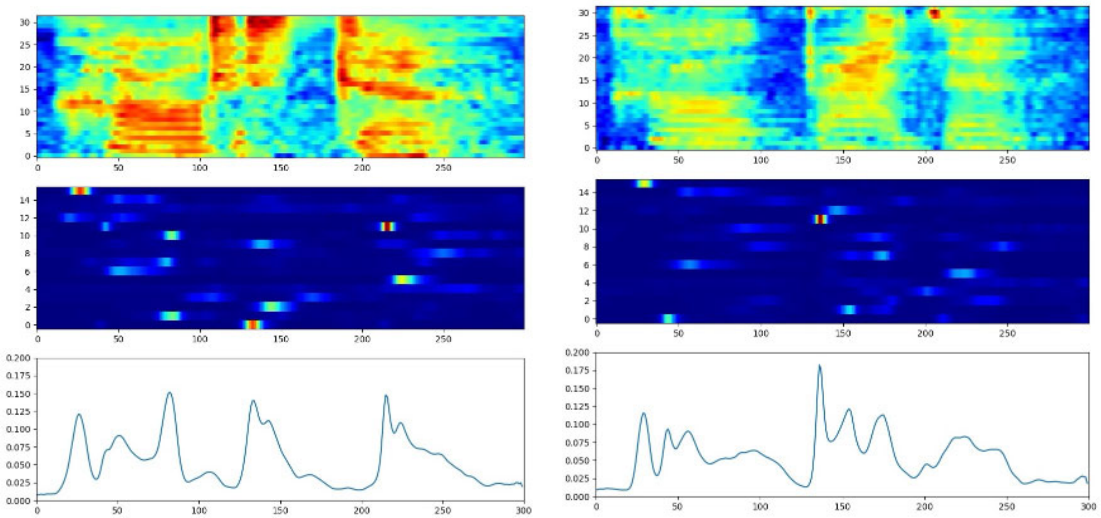
Architecture			**Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	3.02	0.137	0.676
	yes	–	3.04	0.141	0.646
CLS	no	R=1	3.32	0.143	0.697
CLS-DIST	yes	R=1	3.25	0.144	**0.621**
CLS	no	R=100	3.09	0.137	0.658
CLS-DIST	*yes*	R=100	**2.68**	**0.122**	0.652

Architecture			**Female + Male**		
Type	**T/S**	**Sampling**	**EER%**	**minDCF08**	**minDCF10**
AVG	no	–	3.58	0.136	0.521
	yes	–	3.65	0.138	0.501
CLS	no	R=1	3.60	0.134	0.520
CLS-DIST	yes	R=1	3.57	0.135	0.494
CLS	no	R=100	3.43	0.129	0.505
CLS-DIST	*yes*	R=100	**3.19**	**0.123**	**0.492**

8.5.4 Analysis of Class Token Self-Attention Representations

In view of the relevant results obtained, we have also conducted an analysis to interpret where the self-attention matrix A is focusing on each system. To perform this analysis, we have employed the system with the best performance from each database, and within these systems, the last MSA layer of the student model has been selected to make the representations. In addition, we have chosen different utterances to analyze in Figures 8.5 and 8.6. For each utterance, three figures are plotted: the spectrogram of the utterance, the matrix of attention weights corresponding to the class token for each of the 16 heads of the MSA layer, and the sum of the weights of these class token attentions.

In Figure 8.5, two examples of utterances of different phrases (*"Call sister"*, *"Call brother"*) pronounced by the same speaker are shown. These examples are obtained from the evaluation set of the RSR-Part II database. Whether we look in the middle and bottom figures, we can observe the relevant information learned by the self-attention weights to correctly determine the phrase and speaker of each utterance using the class token. Note that these two phrases of example begin exactly the same with the word *Call*, so focusing on the beginning of the figures, we observe how the self-attention gives a similar relevance pattern in both cases to the areas of same phonemes. Moreover, we can also see that the weights do not pay attention to the area at the beginning and the end of the utterances that correspond to moments of silence.

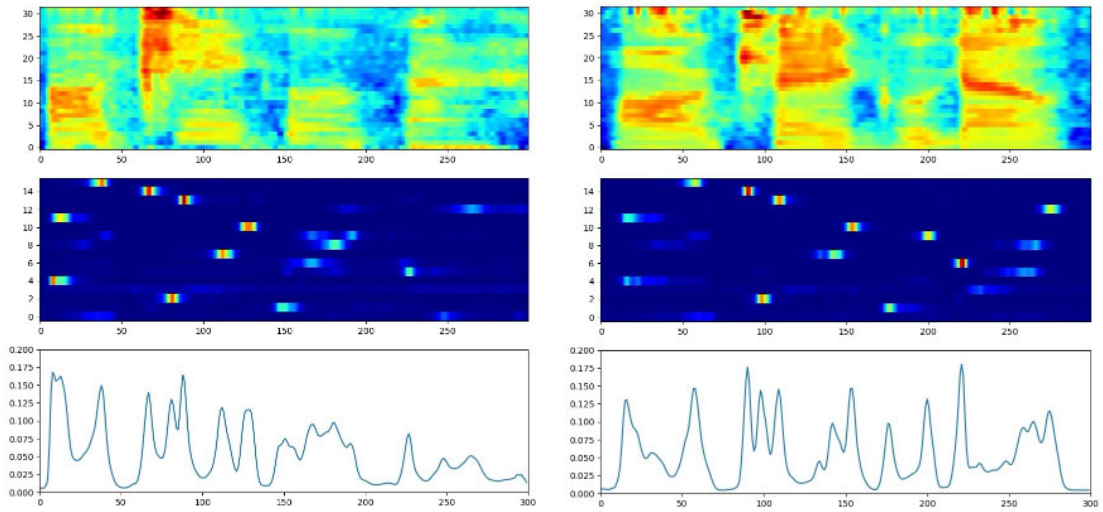


(a) Call sister, speaker 1

(b) Call brother, speaker 1

Figure 8.5: Visualizing two examples of different phrases of RSR2015-Part II which are pronounced by the same speaker. In both cases, three representations are presented. The figure on top shows the spectrogram of each phrase. In the middle, the attention weights learnt by the class token for each of the 16 heads in the last MSA layer are depicted. Finally, the plot on bottom is the sum of the rows of the previous weight attention matrix.

Figure 8.6 represents two examples of utterances of the same phrase (*"OK Google"*) pronounced by different speakers. In this case, the examples are obtained from the evaluation set of the DeepMine-Part 1 database. Note that since these figures are of the same phrase, self-attention is focused on the same areas, but different relevance is given to



(a) OK Google, speaker 1

(b) OK Google, speaker 2

Figure 8.6: Visualizing two examples of the same phrase of DeepMine-Part 1 which are pronounced by different speakers. In both cases, three representations are presented. The figure on top shows the spectrogram of each phrase. In the middle, the attention weights learnt by the class token for each of the 16 heads in the last MSA layer are depicted. Finally, the plot on bottom is the sum of the rows of the previous weight attention matrix.

some of them. Besides, the effect of not focusing on the beginning and end of the utterance also occurs in these examples.

8.6 Conclusions

In this chapter, we have presented a novel approach for the speaker verification task. This approach is based on the use of a learnable class token to obtain a global utterance descriptor instead of employing the average pooling. Moreover, we have developed an alternative to create the class token with a sampling strategy that introduces uncertainty that helps to generalize better. Apart from the previous approach, we have also employed a teacher-student architecture combined with an extra distillation token to develop a more robust system. Using this architecture, the distillation token in the student network learns to replicate the predictions from the teacher network. Both proposals were evaluated in two text-dependent speaker verification databases. Results achieved show in RSR2015-Part II that each of the approaches introduced to obtain a robust system and reduce potential underperformance due to the lack of data improves the overall performance. However, in DeepMine-Part 1, the results obtained replacing only the average embedding by the class token present a small but consistent improvement, while the use of a teacher-student architecture achieves a great improvement and confirms the power of this kind of approach to train the systems.

Part III

Metric Learning

9

Analysis of Different State-of-the-art Training Loss Functions for Verification Systems based on Deep Neural Networks

9.1 Motivation

During Chapters 4, 5, 6, 7 and 8, different approaches to obtain robust representations using Deep Neural Networks (DNNs) have been analyzed. To develop these approaches, we have followed the typical framework to project the utterances into a low dimensional space employed in many state-of-the-art verification systems based on DNNs [132, 135, 166]. This framework has three key components: a data processing step followed by a backbone network, a pooling mechanism which produces representations to characterize the whole file, and a loss function to train the entire system. In all previous chapters, several alternatives have been studied for the backbone and pooling parts, while the loss

function has always been the same which is based on a widespread approach using Cross Entropy (CE) loss [75, 135] combined with a complementary loss such as Ring loss [84]. Then, the verification process is performed through a separated back-end, in this case, a similarity metric [74]. Although these verification systems have already provided reasonably good results, the CE loss employed is a general approach that was not designed to optimize the verification task itself. This loss function does not encourage the discriminative learning of features also valid for out of domain data and the straightforward classification models trained with it may not be always able to generalize properly on unseen data. Therefore, when these systems are trained for classification using CE loss, the performance of a verification task may not be optimal as it will be shown in the following chapters, since it is a problem where generalization of unseen classes is important. Hence, for the successful training of DNNs, the chosen loss function plays an important role, since a suitable loss function can improve the ability to discriminate of the verification systems. For this reason, recently, the design of new loss functions has been widely investigated to find the most suitable loss function for training deep learning systems. These efforts have focused on two lines of study, on one side, the verification or metric learning loss functions, and on the other side, the redesign of identification or classification loss functions. Metric learning loss functions are increasing as a relevant research focus since these approaches allow making the training process more appropriate to the evaluation procedure [75, 142]. The main purpose of metric learning algorithms consists of bringing similar samples closer, while different samples are pushed apart from each other using a specific loss function. Thus, these approaches aim to learn a more discriminative embedding space. However, this kind of metric learning loss functions requires careful sample preparation, which usually involves a high computational cost. To address this problem, recent research efforts have focused on redesigning traditional classification loss functions to improve discrimination ability. Nevertheless, the existent loss functions in both lines of study have not been created focused on the goal task.

Motivated by the above ideas, the aim of this chapter is to describe some of the most extended state-of-the-art loss functions, as well as their main disadvantages, before moving on to the explanation of the loss functions developed in this thesis. These loss functions have been created oriented to the verification task as will be explained in detail in the following chapters.

9.2 State-of-the-art Loss Functions for Training DNNs

Training loss functions play an important role in verification systems based on DNNs, since an effective loss function can improve the discriminative power of the learned features. The selection of which loss function should be used depending on the task is also relevant. Loss functions can be grouped into two categories:

- Identification or classification loss functions: which are used in classification tasks where all test identities are predefined in the training set, and the features are ex-

pected to be separable. In this category, the current extended function is CE loss with softmax output units [71, 285, 286] combined with Ring loss [84] or Center loss [83], and its variants, such as Angular Softmax loss (A-Softmax) [81, 145], Additive Angular Margin loss (ArcFace) [86] or Additive Angular Margin Softmax loss (AAMSoftmax) [143].

- Verification or metric learning loss functions: which are designed to improve the discrimination power with a pairwise- or triplet-based training and a similarity metric, which leads to a supervised embedding learning in Triplet loss [75, 142], Tuple-based End-to-End (TE2E) loss [164], Generalized End-to-End (GE2E) loss [168], Contrastive loss [87], Partial AUC loss (pAUC) [146–148] or NeuralPLDA [149].

Both groups of loss functions have several variants. In this section, we focus on a detailed explanation of the most widespread functions of each group.

9.2.1 Cross-Entropy Loss

The traditional and most common identification loss function is Cross-Entropy (CE) loss [71,285,286]. Due to its simplicity, excellent performance and probabilistic interpretation, this function has been widely applied for multi-class classification. CE loss can be defined as,

$$L_{CE} = -\sum_{i}^{m}\sum_{j}^{N} y_{ij} \cdot \log(\hat{y_{ij}}), \tag{9.1}$$

where y_{ij} is the ground truth class label with $i \in \{1, ..., m\}$ where m is the number of samples and $j \in \{1, ..., N\}$ where N is the number of classes, and $\hat{y_{ij}}$ is the predicted probability extracted from the output of the last fully connected layer. When the labels are hard, this loss is also known as Negative Log-likelihoood (NLL) and can be expressed as,

$$L_{CE} = -\sum_{i}^{m} \log(\hat{y_i}), \tag{9.2}$$

Moreover, this loss function is usually combined with a softmax function. Thus, CE loss can also be written as,

$$L_{CE} = -\frac{1}{m}\sum_{i}^{m} log\frac{\exp\left(W_{y_i}^{T} \cdot x_i + b_{y_i}\right)}{\sum_{j}^{N}\exp\left(W_{j}^{T} \cdot x_i + b_j\right)}, \tag{9.3}$$

where x_i is the input sample with $i \in \{1, ..., m\}$ and m is the number of samples, y_i is the class label, W is the weight matrix, b_{y_i} and b_j indicate the bias values, W_{y_i} and W_j are the y_i and j column of W with $j \in \{1, ..., N\}$ and N is the total number of classes.

When a DNN is trained using this loss function, it learns how to separate features as far away as possible from the decision boundary, which is the goal for a classification

task. Nevertheless, deeply learned embeddings are not explicitly encouraged to enlarge the inter-class distance and reduce the intra-class variations, which is not suitable for verification systems, as they require separable and also discriminative embeddings for the verification task. Therefore, CE loss improves the posterior probability of training samples, which is not the best approach to achieve generalization in learned representations in verification tasks which compare if unseen examples in the training set belong to the same class.

9.2.2 CE Loss combined with Ring Loss

To solve the above drawbacks with CE loss and simultaneously keep the same efficiency during training, other approaches have been proposed with encouraging results. CE loss learns to separate embeddings of different classes, but this loss function does not address intra-class compactness. In order to mitigate the effects of the lack of generalization in the representations, an alternative is to combine CE loss with a complementary loss such as Ring loss [84]. Ring loss was proposed to apply a convex norm constraint on the primary loss to normalize the embeddings and bring compactness to them. With this complementary function, the system learns to force the embedding norms to be close to the unit norm, which increases the generalization in the representation among different classes since all vectors have similar module and different angles are used to represent the data while the intra-class feature variability is reduced. Ring loss is formulated as,

$$L_R = \frac{\lambda_w}{2m} \sum_{i=1}^{m} (\|x_i\|_2 - R_n), \tag{9.4}$$

where R_n is the target norm value, usually 1, λ_w is the loss weight, x_i is the input sample of the penultimate layer with $i \in \{1, ..., m\}$ and m is the number of samples. Thus, the joint loss to minimize with this approach is defined as,

$$L = L_{CE} + L_R. \tag{9.5}$$

9.2.3 Angular Softmax Loss

Many recent studies have pointed out the need to define new CE variants to address the lack of generalization for classes out of the training set. Thus, another interesting approach to solve the generalization problems was to introduce a redesign of CE loss by incorporating an angular margin to encourage greater variance among classes. This loss function is known as Angular Softmax, or A-Softmax loss [81, 145] which introduces an angular margin to learn angular discriminative embeddings. The loss function is defined as follows:

$$L_{ANG} = -\frac{1}{m} \sum_{i}^{m} log \frac{\exp(\|x_i\| \zeta(\theta_{y_i,i}))}{\exp(\|x_i\| \zeta(\theta_{y_i,i})) + \sum_{j \neq y_i}^{N} \exp(\|x_i\| cos(\theta_{y_i,j}))}, \tag{9.6}$$

where $\zeta(\theta_{y_i,i})$ is the angle function which is a monotonic function defined as,

$$\zeta(\theta_{y_i,i}) = (-1)^k cos(m_a \theta_{y_i,i}) - 2k, \tag{9.7}$$

with $\theta_{y_i,i} \in [\frac{k\pi}{m_a}, \frac{(k+1)\pi}{m_a}]$, $k \in [0, m_a - 1]$, and m_a is an integer to control the angular margin.

A-Softmax loss has been proved to be an effective method for improving some recognition systems. However, it is difficult to train with this function since it is sensitive to the values of the parameters.

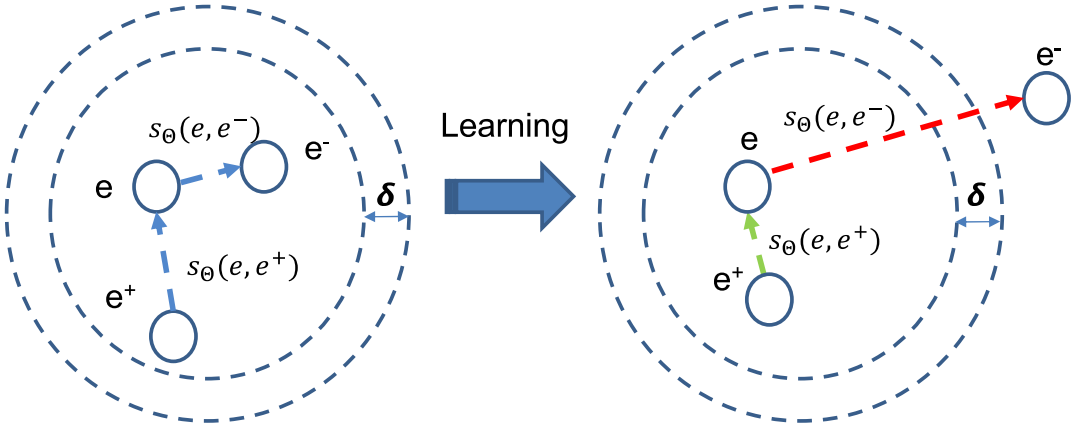


Figure 9.1: Triplet loss learning process.

9.2.4 Triplet Loss

Motivated to improve the discriminative power of the features extracted from the network, the metric learning approaches or verification loss functions were introduced in the training process of DNNs. One of the most widespread verification loss functions for metric learning is Triplet loss [75]. To use this type of loss function, a triplet neural network structure is applied in which three examples are selected to create a negative pair and a positive pair of samples. They are defined as an example of a specific class called anchor (e), a positive sample of the same class of the anchor (e^+), and a negative sample of a different class (e^-). Once the triplet selection process is made, three instances of the same neural network with shared parameters are trained to enforce a larger similarity metric on the anchor-positive pair than on the anchor-negative pair with a margin, as Figure 9.1 depicts. Triplet loss is defined as,

$$L_{TR} = \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} \left[\|(s_\theta(p_i^+)\|_2 - \|s_\theta(p_j^-)\|_2 + \delta \right]_+, \tag{9.8}$$

where $s_\theta(p_i^+)$ is the similarity metric of each pair of anchor-positive embeddings where $p_i^+ = (e, e_i^+)$ with $i \in \{1, ..., m^+\}$ and m^+ is the total number of positive examples, $s_\theta(p_j^-)$ indicates the metric of each pair of anchor-negative embeddings where $p_j^- = (e, e_j^-)$ with $i \in \{1, ..., m^-\}$ and m^- is the total number of negative examples, and δ is the minimum margin between those similarities.

10

Optimization of the Area Under the ROC Curve for Training Deep Neural Networks

10.1 Motivation

Ideally, verification systems based on Deep Neural Networks (DNNs) should be trained to directly carry out the verification process and, also, all parameters should be trained at the same time. For example, training an end-to-end system as a binary classification, so the system is able to determine between two examples as a target or a non-target trial. Some attempts with this philosophy have been successfully trained thanks to the availability of a large amount of training data or using strong pre-trained models as starting point. However, these situations are not the most typical for many specific tasks. Therefore, as Chapter 9 has introduced, the most widespread approach to develop verification systems consists of training DNNs for multi-class classification using the traditional Cross-Entropy (CE) loss. After that, a back-end is applied on the embeddings extracted from the previous DNN to perform the verification process, either through a similarity metric [74], a Probabilistic Linear Discriminant Analysis (PLDA) [287, 288], or another NN. Nowadays, one of the most common back-end approaches based on DNNs is a triplet neural network with Triplet loss [75, 289]. Nevertheless, this loss function is not designed to optimize the verification task. For this reason, an alternative to Triplet loss optimization has been proposed in this thesis. The use of a novel method combined with the triplet philosophy is more appropriate for a verification task. In this method, the loss function approximates the Area Under the Receiver Operating Characteristic (ROC) Curve [290–293] in a differentiable way. Area Under the Curve (AUC) is a metric used to evaluate performance in verification systems. This metric measures the area between the ROC and the axes, and AUC is obtained by sweeping across all thresholds (under the assumption of uniform sampling). Therefore, by optimizing the proposed loss function, we can optimize the AUC performance.

In summary, in this chapter, we propose a new optimization procedure as a back-end that combines the triplet neural network approach with an approximation of AUC (*aAUC*) as a loss function. The optimization of AUC as a cost function is appropriate for a detection task as it directly correlates with performance metric of the system. Moreover, this kind of approach needs careful selection of the triplet examples for correct convergence during training. For this reason, a smart triplet selection method is included in this chapter. In addition, to prove that this loss function is suitable for any verification task, we have employed it to train systems for text-dependent speaker, face and language verification tasks.

10.2 Triplet Neural Network

The triplet neural network defines a cost function to evaluate the embeddings provided by three instances of the same neural network with shared parameters. As input to this network, three examples are used, one example from a specific class x (an anchor), an-

other example from the same class x^+ (a positive example), and an example from another class x^- (a negative example). In most of the existing recognition systems using this approach [75, 132, 141, 169, 289, 294], the goal of the neural network learning process is to maximize the distance between the anchor and the negative example while the distance between the anchor and the positive example is minimized, if it is greater than a margin defined in Triplet loss. Unlike previous systems that train the neural network using Triplet loss, in this thesis, we propose to use the AUC metric as the training objective, which allows us to directly optimize a well-suited loss function that measures the performance of the whole system independently of the operating point. The choice of the AUC as the objective function for the training process is due to its simplicity, since it can be approximated with differentiable functions [290] allowing us to build an end-to-end system. Furthermore, another important point to correctly train this type of systems is the triplet data selection that is applied to choose which are the examples that compose the triplets. Therefore, in this chapter, we have decided to introduce an approach similar to the triplet sampling strategy proposed in [75] which is usually called Hard Negative Mining.

For the training of the triplet neural network as a back-end approach, the pipeline of the proposed scheme is depicted in Figure 10.1. In addition, we detail below the proposed objective function and the training strategy employed to carry out this back-end.

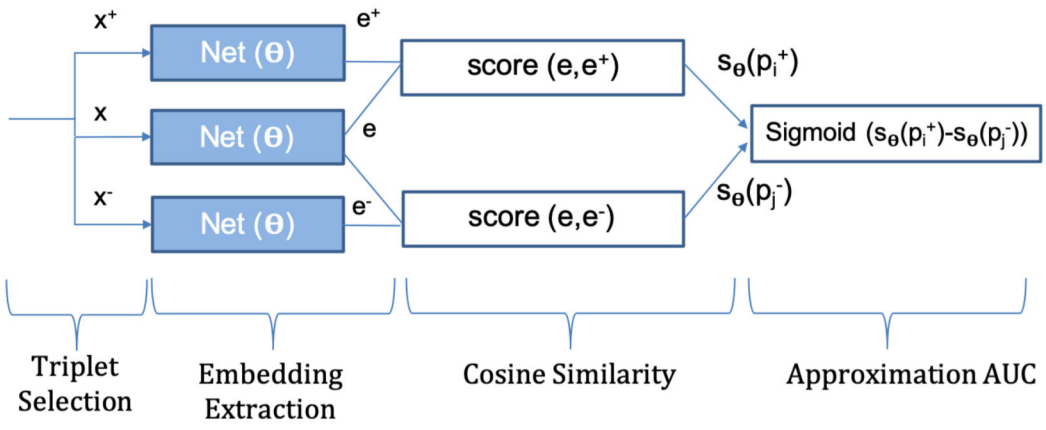


Figure 10.1: Triplet neural network, the examples are grouped in triplets by the triplet selection to train the network and evaluated the two pairs of embeddings to optimize the objective function, where x is the anchor and e is its embedding from back-end network, x^+ is the positive example and e^+ is its embedding, and x^- is the negative example and e^- is its embedding. Besides, $s_\theta(p_i^+)$ is the cosine similarity from anchor-positive pair, and $s_\theta(p_j^-)$ is the cosine similarity from anchor-negative pair.

10.2.1 Optimization of the Area Under the ROC Curve

Metric learning loss functions aim to improve the generalization ability of systems as they learn to better discriminate in verification problems. However, these functions are designed without considering the measure of the performance used in the verification process. Verification systems are generally trained with discriminative paradigms to op-

timize classification performance. Nevertheless, in that way, the training process does not consider the verification process and relative measures such as the trade-off between false alarms and misses. For that reason, to make the training process more consistent with the evaluation procedure, we propose to directly optimize the Area Under the Curve (*AUC*) as loss function using the triplet philosophy. AUC is an operating point independent metric and measures the probability that all pairs of examples are correctly ranked. Therefore, AUC is a proper metric to make the training procedure consistent with the evaluation process. Since this metric is not differentiable, we propose an effective approximation of the AUC function (aAUC) to enable backpropagation of the gradients during the training process.

For m training examples, AUC is defined as the function that maximizes the average number of times the score of anchor-positive pairs is greater than the score of anchor-negative pairs. The anchor-positive score is given by the cosine similarity value $s_\theta(p_i^+)$ where $p_i^+ = (e, e^+)$ indicates each pair of anchor-positive embeddings with $i \in \{1, ..., m^+\}$ and m^+ is the total number of positive examples, and the anchor-negative score is provided by the cosine similarity value $s_\theta(p_j^-)$ where $p_j^- = (e, e^-)$ represents the anchor-negative pairs with $j \in \{1, ..., m^-\}$ and m^- is the total number of negative examples. Both values are also expressed as a function of the net learning parameters θ. Hence, given a set of network parameters θ, the AUC function can be written as

$$AUC(\theta) = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} \mathbb{1}(s_\theta(p_i^+) > s_\theta(p_j^-)), \tag{10.1}$$

where $\mathbb{1}()$ has a value equal to '1' whenever $s_\theta(p_i^+) > s_\theta(p_j^-)$, and '0' otherwise. With this expression, the optimization process leads the score of the anchor-positive pair to be greater than the score of the negative pair. This function can be rewritten using the unit step function $u()$ as,

$$AUC(\theta) = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} u(s_\theta(p_i^+) - s_\theta(p_j^-)). \tag{10.2}$$

To enable the backpropagation of the gradients, this expression must be approximated in order to be differentiable. For that reason, we substitute the step function by a sigmoid function, as we can see in Figure 10.2 is a close approximation to the step function which is one if the condition is met and zero otherwise. In our case, the condition is that the score of e and e^+ must be greater than the score of e and e^-. Thus, our approximation of the AUC (*aAUC*) loss function can be formulated as,

$$aAUC(\theta) = \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} \sigma_\alpha(s_\theta(p_i^+) - s_\theta(p_j^-)), \tag{10.3}$$

where $\sigma_\alpha()$ is the sigmoid function which is defined as,

$$\sigma_\alpha(s) = \frac{1}{1 + e^{-\alpha s}}, \tag{10.4}$$

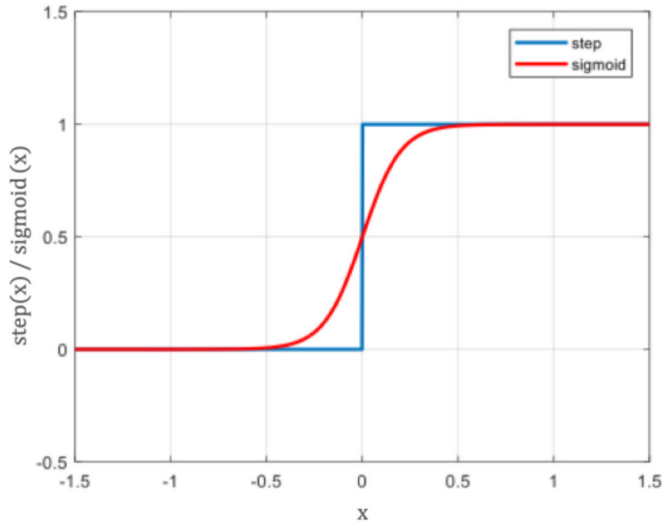


Figure 10.2: Representation of unit step and sigmoid functions, where the sigmoid function can be seen as a good approximation of the unit step function.

where α is an adjustable parameter which modifies the slope of the sigmoid to make it closer to the unit step [293]. The use of a differentiable function as the sigmoid function allows us to include this optimization function in combination with the alignment mechanism presented in Chapter 5 in our end-to-end system. Thus, we seek the network parameters θ^* which maximize this expression:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, aAUC(\theta) = \underset{\theta}{\operatorname{argmax}}\, \frac{1}{m^+ m^-} \sum_{i=1}^{m^+} \sum_{j=1}^{m^-} \sigma_\alpha(s_\theta(p_i^+) - s_\theta(p_j^-)). \tag{10.5}$$

10.2.2 Triplet Training Method

Aside from the loss function, the method used to select triplets is an important factor in training a good triplet neural network. If the anchor-negative example consistently comes from easily-separable classes, the network will not provide the discrimination needed to separate closely-related classes. In this section, we describe the different triplet selection strategies to perform this selection. Nevertheless, depending on the verification task and the data, we have decided which one to apply.

Random Selection

The first strategy is a straightforward approach consisting of a random choice of classes and examples of each class. To perform this selection, at each iteration, we randomly choose G groups of P classes where $G \cdot P$ is equal to the total number of classes N. Then, all examples from each class selected from each group are used to make random triplets to train the neural network.

Hard Negative Mining Selection

Random selection is the simplest way to perform this process, but it is not the smartest option. Therefore, in this thesis, this process has also been developed following the triplet sampling strategy proposed in [75] termed Hard Negative Mining strategy, which is an intelligent selection process. This technique involves selecting anchor-negative pairs with the maximum similarity value (hard negative) for which the system triggers a false alarm, and anchor-positive pairs with the minimum similarity value (hard positive) that the system can not detect and produces a miss. To manage this selection process, we propose two methods using both the same random selection for classes:

- *Hard Negative Mining Type 1 (HM1)*: This method has been developed to create the triplets with only a random subset of K examples from each P class.
- *Hard Negative Mining Type 2 (HM2)*: As an evolution of the first approximation, we employ the whole set of examples from each P class to find the most complicated cases to discriminate. We handle this process in Algorithm 1 using a random selection of the identities, since using all of them at the same time would involve a significant cost. We use all the examples from each identity to find the cases which are more complicated to discriminate. Once the pairs of challenging examples have been calculated, we select only a small percentage of them p_r to train the triplet network. This value is chosen according to the size of the database, so the number of examples used for training is similar to the original size.

After the selection process, the training of the neural network is made using subsets of these pairs selected in the triplet selection process, the number of pairs selected is defined by the batch size. The *aAUC* value and the gradients to update the network are calculated for these pairs of embeddings in each subset or minibatch. Thus, this computation is not the exact AUC, because to obtain the whole AUC, we should calculate the value and gradients for all pairs at the same time, but the convergence would be extremely slow. Hence, the benefit of this process in terms of performance comes with a higher computational cost.

10.3 Experiments and Results in Speaker Verification

In this section, we present the experiments and results obtained for the speaker verification task by applying the proposed novel approach with the following verification system.

Algorithm 2: Training algorithm for triplet neural network using *aAUC* loss function.

Input: Input examples to triplet selection process $\mathcal{X}$, class labels **Y**, initial network parameters θ, number of classes N_y, selection percentage of examples p_r, adjustable parameter α, batch size b, learning rate λ, number of class for block N_s, and the number of epochs N_e

Output: Updated network parameters θ.

$\mathcal{X} \leftarrow \emptyset$

for n=1 **to** N_e **do**

1. Triplet Selection Process

for n_y = 1 **to** N_y **do**

1.1 Step 1, select randomly N_s classes from Y:

ids=random_selection(Y,N_s)

1.2 Step 2, evaluate all versus all selected ids.

1.2.1 Step 2.1, choose the percentage p_r of reference training examples x (anchors).

1.2.2 Step 2.2, select the percentage p_r of the cases with lowest score and same class as reference x^+ (hard positives).

1.2.3 Step 2.3, choose the percentage p_r of the cases with highest score and different class to the reference x^- (hard negatives).

1.2.4 Step 2.4, add the triplet to the new training set $\mathcal{X}$:

$\mathcal{X} \leftarrow \mathcal{X} \cup \{(x, x^+, x^-)\}$

end

2. Network Training

2.1 Step 1, every update a minibatch of size b is extracted from the train set $\mathcal{X}$ composed by triplets (x, x^+, x^-).

for $(x, x^+, x^-) \in \mathcal{X}$ **do**

2.2 Step 2, extract the neural network embeddings :

e=Net(x,θ)

e^+=Net(x^+,θ)

e^-=Net(x^-,θ)

2.3 Step 3, obtain the scores from each pair of embeddings (e, e^+),(e, e^-):

$s_\theta(e, e^+)$=score(e,e^+)

$s_\theta(e, e^-)$=score(e,e^-)

2.4 Step 4, calculate the *aAUC* loss function using (10.3)**:**

$aAUC$=$\sigma(\alpha(s_\theta(e, e^+) - s_\theta(e, e^+))$

2.5 Step 5, update network parameters θ with gradient of *aAUC*:

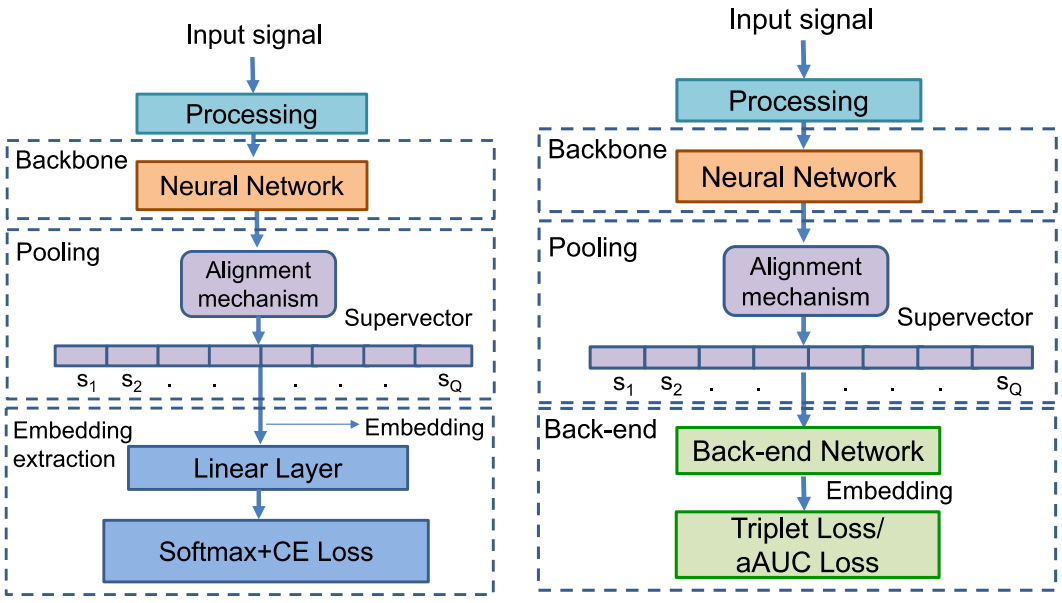
$\theta \leftarrow \theta + \lambda \bigtriangledown_\theta(aAUC)$

end

end

10.3.1 System Description

The text-dependent speaker verification system employed to carry out these experiments is based on the systems developed in Chapter 5 and 6. As reference backbone network for each alignment mechanism, the best configuration from those chapters has been used. The output of the pooling part of the system will be the neural network supervector or embedding. These neural network supervectors with fixed dimension are processed by the back-end to train a classification of identities, therefore, enforcing the separability of the speaker in the supervector space or to provide directly the verification scores.



(a) Architecture type C (b) Architecture type D

Figure 10.3: The architectures developed to check the effectiveness of our proposed back-end, 10.3(a) the architecture type C is trained for multiclass classification using the alignment mechanisms. In 10.3(b), the architecture type D is trained to optimize the back-end network.

In Figure 10.3, we show the architectures employed to develop this approach. These architectures are designed as follow:

- *Architecture C.* Chapter 5 have introduced the architecture type C depicted in Figure 10.3(a) which has been created by combining the architecture type B with a backbone network, and a flatten layer is used to link with the last layer to train the system for multi-class classification with CE loss. The alignment layer explained in that chapter is used as a link between the backbone and loss parts, allowing us to train the whole system to optimize any cost function we decide. To implement the backbone network, the best configuration of Chapter 5 and 6 for each alignment technique has been employed. For the verification process, once our system is trained, a neural network supervector is extracted from the flatten layer for each enrollment and test utterance, and then a cosine similarity is applied on them to produce the verification scores.

- *Architecture D.* In this chapter, we propose another architecture with a novel cost function in this context. This function is better integrated with the back-end and, thus, comes closer to the task objective. In the architecture type D which we show in Figure 10.3(b), an embedding is obtained for each utterance, and then the back-end is applied to provide the verification scores directly with the metric allowing us to have an end-to-end system.

10.3.2 Experimental Setup

The following section describes the experimental setup employed for the text-dependent speaker verification task with RSR2015-Part I and II. To develop our experiments, a set of features composed of 20 Mel-Frequency Cepstral Coefficients (MFCCs) with their first and second derivates are employed as input to train the alignment mechanisms and also as input to the DNN. The bkg partition has been employed to train two different alignment mechanisms, both of which were trained to obtain one model per phrase without the need to know the phrase transcription. On the one hand, Hidden Markov Models (HMMs) have been trained using a left-to-right model of 40 states for each phrase. On the other hand, a Gaussian Mixture Model (GMM) of 64 componenents has been trained per phrase. With these models, we can extract alignment information to use inside our DNN architecture. Furthermore, we have trained the DNNs using an Adam optimizer [248]. In all experiments with the *aAUC* function, the α parameter in the sigmoid function has a fixed value of 10 to have a shape close to the unit step since this parameter acts as a control of steepness. For this task, we fixed it using the dev partition.

In this section, a set of experiments has been developed for each part of the text-dependent speaker verification database to compare the diverse back-end loss functions. As back-end, the network used is composed of two dense layers and the specific configuration is in Table A.14 in Appendix A. The triplet selection strategy employed to train is described in Section 10.2.2 with the name of Hard Negative Mining Type 2. Moreover, these approaches have been evaluated with the two alignment mechanisms proposed in Chapter 5. Each set of experiments has been developed using the bkg, dev and eval data corresponding to each part and the parameters of the system and training algorithm have been selected using the dev partition.

10.3.3 Results with RSR2015-Part I

In the first experiments with RSR2015-Part I, we have contrasted the best architecture *C* for each alignment mechanism with architecture *D* using the proposed *aAUC* optimization, and also, compared with Triplet loss (TrLoss). To initialize the second architecture, we employ a pre-trained model with architecture *C*.

Table 10.1 shows that if we apply the back-end network to the neural network supervectors instead of just using cosine similarity, we improve the ability to discriminate

between different identities and phrases. This improvement is due to the joint estimation of the backbone and back-end network parameters to optimize the *aAUC* objective. Therefore, it is an end-to-end system as all parameters are learned to jointly optimize the detection task. The best system performance is obtained when we use the end-to-end *aAUC* optimization strategy to combine the backbone and back-end networks with the alignment mechanism, which plays the important role of encoding the features in the supervector communicating both networks. In addition, we have achieved 11% and 15% relative improvement of *EER%* using HMM and GMM with MAP respectively, and the *DCF10* values are also improved in both cases. *CLLR* also improves when we employ the proposed methods. In Figure 10.4, we represent the DET curves for these experiments. These representations clearly demonstrate that the systems using GMM with MAP as the alignment mechanism have great system performance with all approaches, and especially when we apply the proposed *aAUC* function for the back-end, the best system performance is achieved.

Table 10.1: Experimental results on RSR2015-Part I [124] eval set, showing *AUC%*, *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the best backbone neural network with both alignment techniques using the different losses.

Architecture				Female		
Type	**BB**	**Pool.**	**Loss**	**EER%/AUC%**	**min/actDCF10**	**minCLLR/CLLR**
C	*CNN*	*HMM*	*CE*	0.59/99.95	0.10/**0.11**	0.027/0.030
D			*TrLoss*	0.89/99.94	0.23/0.24	0.038/0.041
			aAUC	**0.52**/**99.96**	**0.10**/0.12	**0.023**/ **0.027**
C	*CNN(BDK)*	*GMM*	*CE*	0.51/**99.98**	0.12/0.17	0.021/0.025
D			*TrLoss*	0.63/99.96	0.14/0.30	0.028/0.044
			aAUC	**0.40**/**99.98**	**0.09**/**0.16**	**0.018**/**0.024**

Architecture				Male		
Type	**BB**	**Pool.**	**Loss**	**EER%/AUC%**	**min/actDCF10**	**minCLLR/CLLR**
C	*CNN*	*HMM*	*CE*	0.71/99.95	0.16/0.18	0.030/0.033
D			*TrLoss*	1.09/99.93	0.24/0.25	0.043/0.046
			aAUC	**0.67**/**99.97**	**0.14**/**0.16**	**0.027**/**0.028**
C	*CNN(BDK)*	*GMM*	*CE*	0.78/99.96	0.15/0.24	0.031/0.039
D			*TrLoss*	0.70/**99.97**	0.16/0.41	0.028/0.045
			aAUC	**0.69**/**99.97**	**0.12**/**0.19**	**0.027**/**0.030**

Architecture				Female+Male		
Type	**BB**	**Pool.**	**Loss**	**EER%/AUC%**	**min/actDCF10**	**minCLLR/CLLR**
C	*CNN*	*HMM*	*CE*	0.73/99.95	0.14/0.17	0.030/0.032
D			*TrLoss*	1.07/99.93	0.26/0.26	0.043/0.045
			aAUC	**0.65**/**99.96**	**0.13**/**0.16**	**0.027**/**0.029**
C	*CNN(BDK)*	*GMM*	*CE*	0.66/99.97	0.13/0.21	0.027/0.033
D			*TrLoss*	0.72/99.96	0.17/0.34	0.030/0.038
			aAUC	**0.56**/**99.98**	**0.11**/**0.17**	**0.023**/**0.025**

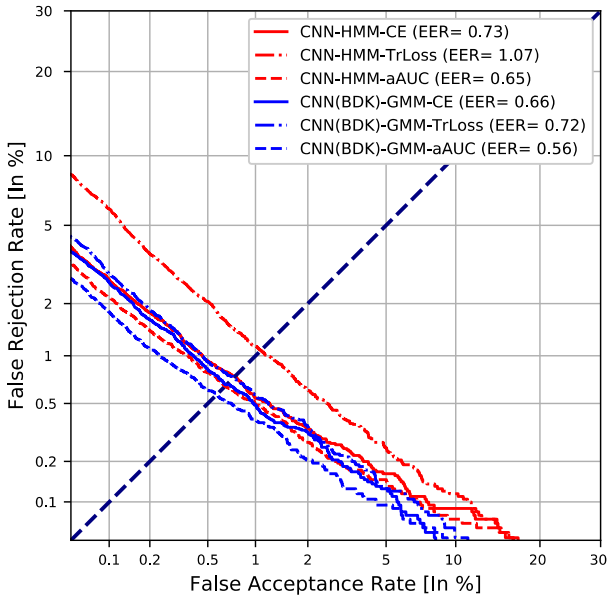


Figure 10.4: DET curves for female+male results on RSR2015-Part I of the best backbone networks combined with the different losses.

10.3.4 Results with RSR2015-Part II

As in the first set of experiments, when we apply the same back-end techniques using architecture *D* on Part II of the database, the discriminative ability also improves. This improvement is shown in Table 10.2 and Figure 10.5. Besides, in terms of relative improvement, *EER%* has improved 11% with both alignment mechanisms, and the *DCF10* values also improve in both cases. We have also improved the *CLLR* values with the proposed approaches.

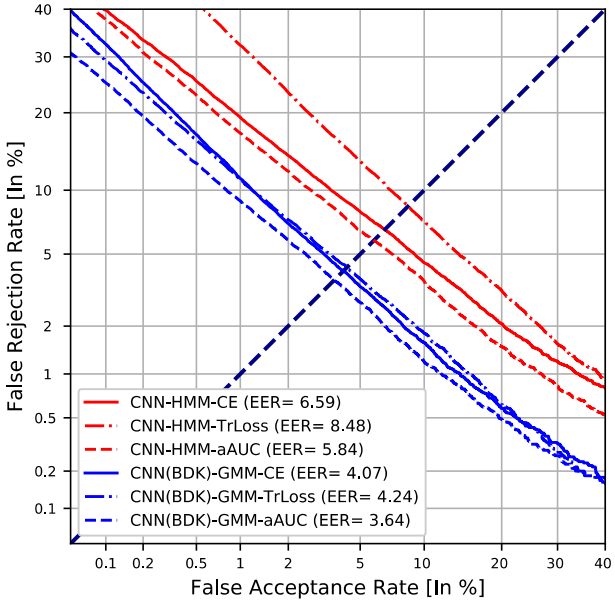


Figure 10.5: DET curves for female+male results on the RSR2015-Part II of the best backbone networks combined with the different losses.

Table 10.2: Experimental results on RSR2015-Part II [124] eval set, showing *AUC%*, *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg+dev subsets to compare the best backbone neural network with both alignment techniques using the different losses.

Architecture				Female		
Type	**BB**	**Pool.**	**Loss**	**EER%/AUC%**	**min/actDCF10**	**minCLLR/CLLR**
C	*CNN*	*HMM*	*CE*	5.55/98.67	0.65/0.82	0.205/0.234
D			*TrLoss*	8.09/97.57	0.89/0.98	0.285/0.417
			aAUC	**5.15/98.92**	**0.61/0.71**	**0.185/0.201**
C	*CNN(BDK)*	*GMM*	*CE*	3.15/99.50	0.52/0.57	0.121/0.169
D			*TrLoss*	3.28/99.46	0.51/0.60	0.127/0.176
			aAUC	**2.74/99.59**	**0.47/0.56**	**0.108/0.147**

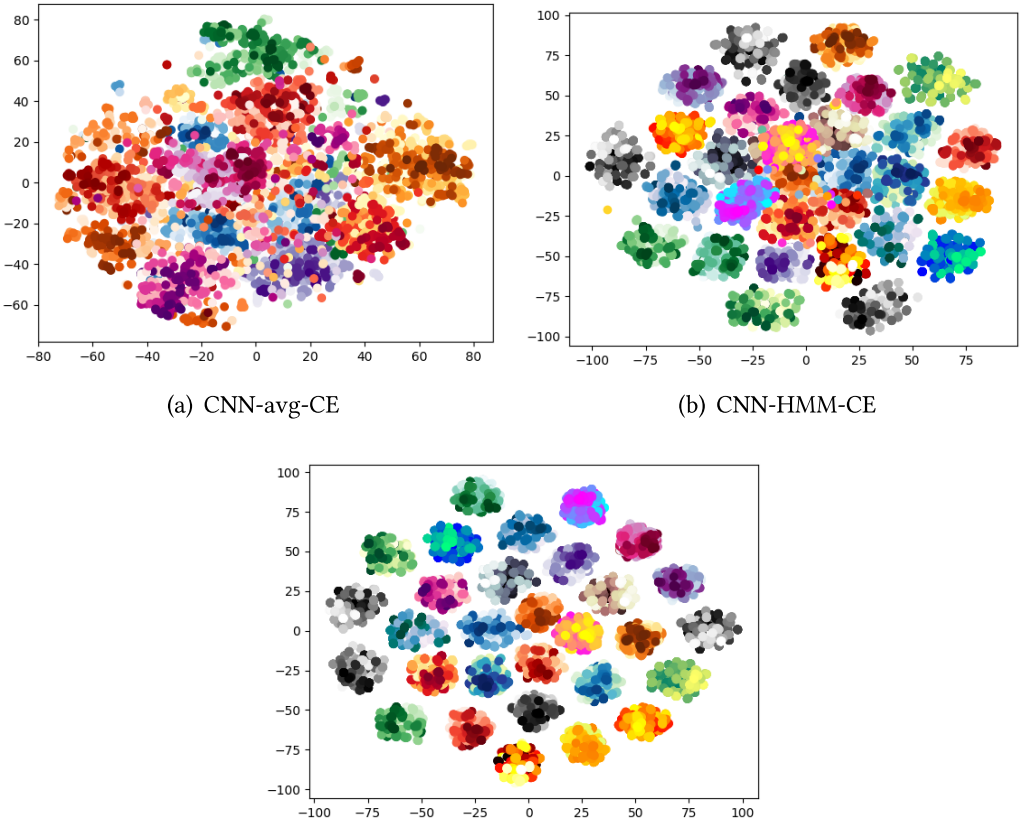
Architecture				Male		
Type	**BB**	**Pool.**	**Loss**	**EER%/AUC%**	**min/actDCF10**	**minCLLR/CLLR**
C	*CNN*	*HMM*	*CE*	6.76/97.98	0.68/**0.69**	0.247/0.384
D			*TrLoss*	8.32/97.35	0.86/0.90	0.296/0.376
			aAUC	**6.02/98.48**	**0.67**/0.76	**0.219/0.243**
C	*CNN(BDK)*	*GMM*	*CE*	4.72/99.00	0.77/0.79	0.176/**0.205**
D			*TrLoss*	4.58/99.10	**0.65**/0.77	0.169/0.372
			aAUC	**3.99/99.25**	**0.65/0.75**	**0.151**/0.259

Architecture				Female+Male		
Type	**BB**	**Pool.**	**Loss**	**EER%/AUC%**	**min/actDCF10**	**minCLLR/CLLR**
C	*CNN*	*HMM*	*CE*	6.59/98.19	0.68/**0.74**	0.237/0.312
D			*TrLoss*	8.48/97.30	0.85/0.97	0.301/0.419
			aAUC	**5.84/98.60**	**0.66**/0.82	**0.212/0.233**
C	*CNN(BDK)*	*GMM*	*CE*	4.07/99.24	0.67/0.69	0.154/**0.189**
D			*TrLoss*	4.24/99.23	0.59/0.69	0.157/0.281
			aAUC	**3.64/99.38**	**0.57/0.67**	**0.137**/0.198

10.3.5 Analysis of Applying a Triplet Neural Network with aAUC Loss as Back-end

In addition to the two sets of experiments described above, Figure 10.6 depicts the embeddings of the thirty phrases in a two-dimensional space using T-Distributed Stochastic Neighbor Embedding (t-SNE) [252]. Figure 10.6(a) represents the embeddings that are extracted from architecture A presented in Chapter 4 using the global average pooling mechanism and CE as loss function, while Figure 10.6(b) shows the neural network supervectors of architecture *C* using the alignment mechanism and CE loss. The third representation in Figure 10.6(c) depicts the embeddings extracted from architecture *D*. As we also shown in Chapter 4, the embeddings obtained from the global average pooling are not able to separate between the same identity with the same phrase and the same identity with different phrase. Besides, we verify that the use of the alignment mechanism improves the discrimination ability for unseen classes, and this ability is even better if we use architecture *D* with our proposed *aAUC* loss function.

For illustrative purposes, Figure 10.7 represents the evolution of the real AUC function (10.2) against the *aAUC* proposed (10.3) during the training process. In this representation, we can see that the proposed differentiable estimation of the AUC function is



(a) CNN-avg-CE

(b) CNN-HMM-CE

(c) CNN-HMM-aAUC

Figure 10.6: Visualizing Average embeddings vs Neural Network Supervectors vs Embeddings from aAUC architecture for 30 phrases from female using t-SNE. Each phrase is marked by one different color scale. The examples used for this representation are from the test set, and they have not been seen during the training process.

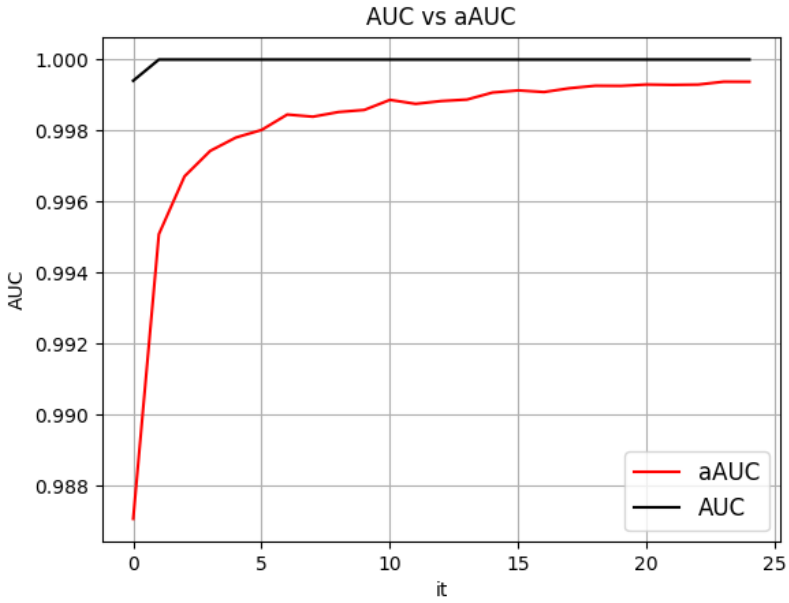


Figure 10.7: Training evolution of real AUC vs aAUC.

getting closer to the real function as the training progresses, which supports the assertion that *aAUC* is an effective approximation of the real AUC function.

The differences between the real AUC function and *aAUC* are due to the approximation used, which is based on a sigmoid function to substitute the original step function. This step function can be seen as a binary counter, only 0 or 1 values, so there are no intermediate values. However, the sigmoid function is a smooth approximation, so during the training process, there are intermediate values of this function, not just 1 or 0 values, like the unit step function.

10.4 Experiments and Results in Language Verification

After proving the success of the proposed back-end approach in the speaker verification task, we have applied this new approach in the language recognition task to improve the discrimination ability between similar languages. We have focused our attention on improving only the back-end part of the system for this task, due to the assumption that the language embedding backbone can generalize to a variety of domains. Besides, working only in the back-end allows us to adapt faster the system for specific needs. To demonstrate the advantages of the triplet neural network back-end, we benchmark against traditional back-ends including Gaussian Back-end (GB), Probabilistic Linear Discriminant Analysis (PLDA), Support Vector Machines (SVM), and NN.

10.4.1 System Description

The following section describes the system used which is shown in Figure 10.8. We first detail the front-end and backbone employed to extract a language embedding from an audio sample. This is followed by a description of various back-ends explored in this task, with each being based on the same embeddings extracted from the backbone. A final calibration step is applied to the scores from each back-end in order to transform the scores into calibrated likelihoods.

Front-end

Front-end is composed of 3 subsystems. The Speech Activity Detection (SAD) system selects frames containing speech, and then a Bottleneck Senone DNN (BN-DNN) system extracts features rich in phonetic information [295]. BN features are used as input to the posterior backbone DNN which is trained to discriminate languages as output classes.

- **Speech Activity Detection (SAD)**

  We use a DNN-SAD model composed of two hidden layers with 500 and 100 nodes, respectively, and trained to discriminate speech vs non-speech frames. It is based

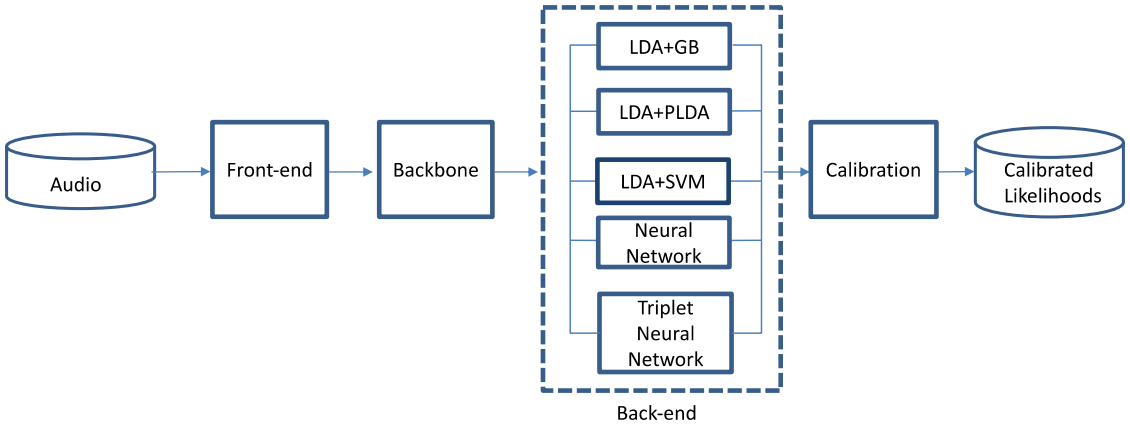


Figure 10.8: The language recognition system, composed of a front-end, backbone, selection of one back-end option, and calibration.

on 20-dimensional MFCC features, stacked with 31 frames to provide an input of 620 dimensions to the DNN, which is first mean and variance normalized over a 201 frame window. A threshold was applied to speech posteriors to determine whether a frame of audio consisted of speech or not.

- **Bottleneck Senone DNN Extractor**

  Senone DNN extracts 80-dimensional bottleneck (BN) features from a bottleneck layer. The network targets 3450 senones (tri-phone states) learned using Fisher and Switchboard data. Log Mel Spectra features of 40 dimensions were used as acoustic input features. The network has 5 hidden layers of 600 nodes with the last hidden layer being the bottleneck of 80 nodes.

Backbone Network used as Language Embedding Extractor

The embedding extractor network is trained with BN features extracted from the BN-DNN and is trained to discriminate 49 different languages from the LRE09 development data described in Chapter 3. The data was augmented with 4 types of degradation: reverberation, compression, non-vocal music, and noise. The structure of the network starts with four frame-level hidden layers followed by a statistics pooling layer and two final segment layers. Features for back-end are extracted from the first segment-level hidden layer of 512 nodes. More information about the DNN structure and the augmentation process can be found in [295].

Traditional Back-ends

To establish a baseline framework for the language verification task, we have applied a range of traditional back-ends to the language embeddings such as SVM, GB and PLDA. All of these techniques have been detailed in Chapter 3, but below we describe the specific

aspects of how they have been employed for this particular task. Before applying each of these techniques, the embeddings were transformed to 48 dimensions with Linear Discriminant Analysis (LDA) into a new feature space that maximizes class separability, followed also by mean and length normalization.

- **Weighted Gaussian Back-end (WGB)**

  The Generative Gaussian Back-end (GB) is one of the most common back-ends applied to language recognition. In this back-end, each language is modeled by a Gaussian distribution, defined by a language-dependent mean and a full covariance matrix shared across all languages. These Gaussian models are used to compute the likelihood of an embedding being based on each of the modeled languages. Since the back-end training data is language-imbalanced, we have employed a modification of GB [191] to normalize for the imbalance by appropriately weighting the examples during the computation of the means m_l and covariance S of the model.

- **Probabilistic Linear Discriminant Analysis (PLDA)**

  Traditionally, PLDA approaches combined with i-vectors were not as successful for language recognition as in speaker recognition tasks, since the number of languages N is much smaller than the speaker identities, so there is a significant loss of information for the identification process with the projection into a *(N-1)* dimensional space. However, in view of the impressive results achieved with the PLDA in speaker verification tasks with embeddings, we apply it to have a reference result of PLDA in embeddings for language recognition. Thus, we have used as a binary detector to determine whether pairs of examples are from the same language or different.

- **Support Vector Machines (SVM)**

  The aforementioned generative back-end techniques may struggle to separate similar languages because they are not trained to separate them explicitly. For this reason, we have decided to substitute the classical generative back-end with discriminative techniques. Before starting with the neural network approaches, we employ a discriminative SVM kernel based classifier [184]. The SVM classifier focuses on maximizing the margin between language classes.

  The SVM is by nature a binary classifier, so in order to use SVM with multiclass data we employ a "one vs. all" strategy. This strategy consists of training language-dependent SVMs that target one language against the pool of all the other languages. In this manner, the margin between the target language and other languages will be largely defined by the closely-related languages to the target language.

Neural Network Back-end

As another reference back-end for comparisson, we also apply a feed-forward neural network (NN) approach, which is trained to discriminate languages directly. In contrast

to the embeddings network trained on short audio cuts of 2-4 seconds, this back-end is trained with longer audio samples using a fixed embedding extraction process. The employed NN is trained using CE loss for multi-class language classification. After the training process, a secondary embedding is extracted from an intermediate layer and a similarity metric such as cosine similarity is applied to compare the test embedding to those obtained from the embeddings of each language in the training dataset.

Triplet Neural Network Back-end

Finally, the last back-end employed is the one proposed in this chapter. Previous work found that language-imbalanced databases are not well suited to NN back-end techniques, which is a hinderance when a large amount of training data for each language is required [191, 203]. Thus, to help with this imbalance, we propose to apply a novel approach for the language recognition task based on a triplet neural network that is trained to discriminate between pairs of embeddings. For this case, the input of this network is three embeddings, one embedding from a specific language e (an anchor), another embedding from the same language e^+ (a positive example), and an embedding from another language e^- (a negative example).

Calibration

The scores generated by all back-ends are calibrated by transforming the scores into proper likelihoods using multiclass logistic regression as described in [217] and Chapter 3. These are then transformed into calibrated log-likelihood ratios (LLRs).

10.4.2 Baseline Results with NIST LRE 2009

In this section, we compare the performance of the traditional back-end techniques described previously, each of them with previous LDA reduction, mean and length normalization, to establish a baseline result using the LRE09 database for language recognition described in Chapter 3. Table 10.3 details *EER%* and *minCLLR* for the initial systems based on three different back-ends. We have found that although traditionally the i-vector and PLDA pipeline has not been the most suitable approach for the language recognition task, embeddings followed by PLDA as a binary detector is found to outperform the back-ends based on classifiers such as WGB and SVM. Thus, in the following section, we use the PLDA back-end as the baseline system.

Table 10.3: Comparison of different traditional back-end techniques on the LRE09 [235] eval data in terms of showing *EER%* and *minCLLR*. Audio files contained 8 seconds of speech.

Back-end	minCLLR	EER%
WGB	0.229	3.48
PLDA	**0.135**	**3.07**
SVM	0.317	4.33

10.4.3 Results using Neural Network Approaches with Different LRE datasets

Once the best result with the traditional back-ends has been established, we benchmark both the existing and proposed neural network approaches against the PLDA baseline. To develop these experiments, we have implemented a straightforward architecture with a single dense layer as described in Table A.15 in Appendix A for the two different NN approaches, and we have used cosine similarity to produce system scores for a test sample.

In Table 10.4, we can see the results achieved on the LRE09 dataset in terms of *EER%* and *minCLLR*, where we compare the NN performance and the triplet neural network perfomance with the different strategies proposed in Section 10.2.2: random selection strategy (TripletNet-Rand), Hard Negative Mining Type 1 (TripletNet-HM1), and Hard Negative Mining Type 2 (TripletNet-HM2). While the traditional NN back-end is outperformed by the PLDA back-end, the TripletNet-Rand is comparable to PLDA, and the use of an intelligent selection strategy via Hard Negative Mining brings reasonable improvements in *minCLLR*. In the best case based on the *minCLLR* result, TripletNet-HM2, we achieve a relative improvement of 17% in terms of *minCLLR* with respect to the PLDA baseline. While, in terms of *EER%*, the result of TripletNet-HM1 is better than TripletNet-HM2 with a relative improvement of 25% respect to PLDA.

Table 10.4: Comparison of PLDA and the triplet neural network approaches on the LRE09 [235], the LRE15 [237] and the LRE17 [238] eval data in terms of showing *EER%* and *minCLLR*.

	LRE09		LRE15		LRE15-nofre		LRE17	
Back-end	minCLLR	EER%	minCLLR	EER%	minCLLR	EER%	minCLLR	EER%
PLDA	0.135	3.07	**0.231**	**5.91**	0.188	**4.60**	0.285	7.35
DNN	0.149	3.38	0.345	8.83	0.258	7.36	0.359	8.11
TripleNet – Rand	0.129	2.72	0.443	8.11	0.351	7.18	0.438	8.02
TripleNet – HM1	0.119	**2.29**	0.372	6.33	0.285	5.49	0.351	6.73
TripleNet – HM2	**0.112**	2.61	0.274	6.36	**0.183**	4.95	**0.283**	**6.72**

Furthermore, we have also evaluated the PLDA baseline and the NN approaches on the LRE15 and LRE17 datasets to check the ability to generalize from our back-end approach. The results in Table 10.4 show that a relative improvement of 9% in terms of *EER%* is achieved on LRE17 using the triplet neural network. Nevertheless, in LRE15, the PLDA performance outperforms the triplet neural network results. The performance of the system with respect to the proposed triplet selection strategies shows a promis-

ing trend, as the more elaborate selections are providing better results. In the case of LRE15, the french cluster is known to suffer from a significant mismatch between the development and evaluation sets [296, 297]. Therefore, we additionally present LRE15-nofre results in which the removal of the 'fre' cluster provided a 33% and 18% relative gain for the Triplet NN and PLDA backends, respectively. This effect demonstrates that the triplet NN backend is particularly impacted by the condition mismatch in this cluster, and warrants further work to cope with this issue.

10.4.4 Limitations of the Triplet Neural Network Back-end

Despite the clearly achieved improvement in system performance on the LRE09 dataset, we expect to obtain a relevant reduction of the confusion between closely related languages. Therefore, we conducted an analysis of the most confused pairs of languages based on the NIST LRE 2009 evaluation: Bosnian-Croatian, Farsi-Dari, and Russian-Ukrainian. Supporting our assumption, it has been found that the confusion between Russian and Ukrainian languages has been reduced using the system with the best performance. Similarly, in the case of Farsi and Dari languages, Farsi test samples have been more accurately allocated to Farsi instead of confused with Dari, however, the detection of Dari samples has not improved. Discrimination between Bosnian and Croatian did not tend to improve nor decline. A possible cause of the failure in the reduction of discrimination problem between these languages could be derived from the lack of examples of some of the languages mentioned. A possible solution to solve this problem would be to bias the triplet selection to provide relatively more triplets from the pairs of languages with greater confusion, with the intent to focus the discriminative power of the DNN on these languages.

10.5 Experiments and Results in Face Verification

Apart from the text-dependent speaker and language verification tasks, the approach proposed in this chapter has also been applied for the face verification task.

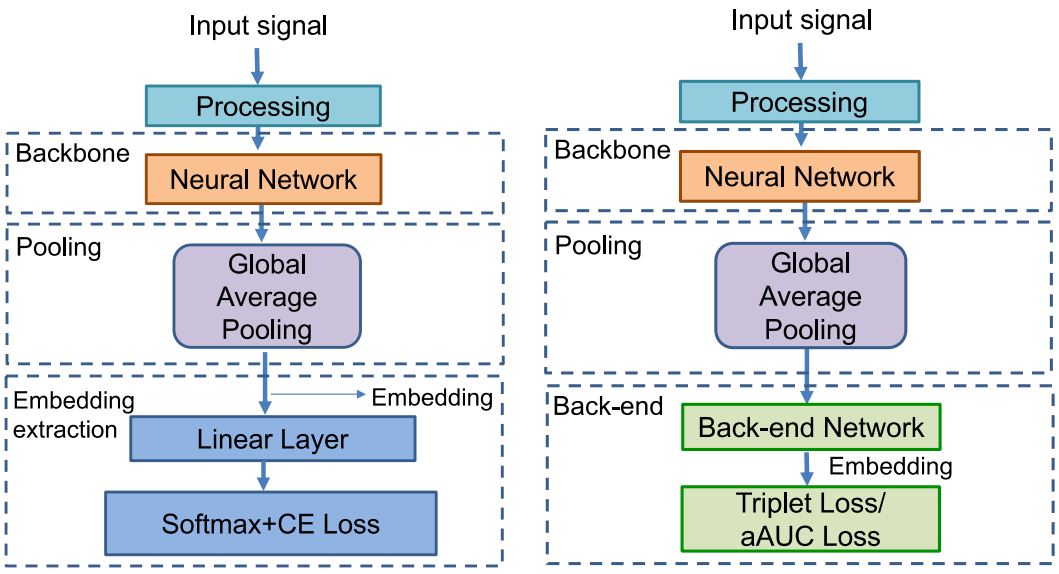
10.5.1 System Description

To validate the effectiveness of this new approach, we have developed the face verification system for this approach using both baseline architectures introduced in Chapter 4. As in the other tasks, the system employed for face verification task has two different architectures which can be seen in Figure 10.9. These architectures are designed as follow:

- *Architecture C.* Unlike architecture C developed previously for the text-dependent speaker verification, we have used two architectures like the employed for the

state-of-the-art systems. In these architectures, we can use the usual global average pooling layer as a link between the backbone and loss parts, allowing us to train the whole system to optimize any cost function we decide. Thus, we have used the architecture depicted in Figure 10.9(a) which was developed in Chapter 4 and was trained for multi-class classification with CE loss. To implement the backbone network, the WideResnet and the pre-trained Facenet model based on Inception Resnet introduced in Chapter 4 have been used. For the verification process, once our system is trained, each enrollment and test image are passed throughout the architecture to extract the embeddings from the flatten layer. After that, a cosine similarity is applied on them to produce the verification scores.

- *Architecture D.* The novel cost function proposed in this chapter is incorporated to the systems thanks to the use of a new architecture. In concrete, to integrate this function in the back-end part, the architecture type D depicted in Figure 10.9(b) is developed, an embedding is obtained for each image, and then the back-end is applied to provide the verification scores directly with the metric allowing us to have an end-to-end system.



(a) Architecture type C (b) Architecture type D

Figure 10.9: The architectures developed to check the effectiveness of our proposed back-end in the case of face verification, 10.9(a) the architecture type C is trained for multiclass classification. In 10.9(b), the architecture type D is trained to optimize the back-end network.

10.5.2 Results using WideResnet with MOBIO

For the face verification task with the MOBIO database, the first set of experiments has been performed using the WideResnet architecture developed in Chapter 4 as architecture C. As baseline to compare the new back-end approach, we have employed the results

obtained in that chapter where architecture C was trained with the traditional CE loss. Moreover, we have also compared the results with architecture D trained using Triplet loss. The triplet selection strategy employed to train is described in Section 10.2.2 with the name of Hard Negative Mining Type 2.

Table 10.5 presents *EER%*, *AUC%*, *minDCF08*, *minDCF10* and *minCLLR* results. In these experiments, we can observe both in the female experiments and in the male case, the best performance is obtained by applying architecture D with our proposed aAUC loss. Note that the improvement achieved in both experiments with verification metrics that measure the whole performance is more consistent and greater than metrics that are evaluated at a specific operating point. Furthermore, the results of the experiments carried out with Triplet loss show that this approach is not a good choice in this task with this dataset.

Table 10.5: Experimental results on MOBIO [244] eval set, showing *AUC%*, *EER%*, *minCLLR*, *minDCF08*, and *minDCF10*. These results were obtained by training with train set to compare the different losses.

Architecture			**Female**				
Type	**BB**	**Loss**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
C	WideResnet	*CE*	4.72	99.12	0.210	0.505	0.166
D		*TrLoss*	4.67	98.98	0.208	0.490	0.175
		aAUC	**3.63**	**99.48**	**0.179**	**0.371**	**0.129**
Architecture			**Male**				
Type	**BB**	**Loss**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
C	WideResnet	*CE*	1.22	99.91	0.060	**0.148**	0.049
D		*TrLoss*	2.66	99.64	0.129	0.369	0.103
		aAUC	**0.92**	**99.93**	**0.055**	0.229	**0.040**

10.5.3 Results using Pre-trained FaceNet with MOBIO

In the second set of experiments, the embeddings extracted from the pre-trained Facenet model are directly evaluated with cosine similarity to obtain the reference result for this set of experiments. As in the previous case, this reference result is compared to architecture D using Triplet loss and our proposed aAUC loss, and the triplet selection strategy is also the same.

In Table 10.6, we can observe *EER%*, *AUC%*, *minDCF08*, *minDCF10* and *minCLLR* results for the female and male experiments. These results show the same trend as the experiments using the WideResnet implemented in Chapter 4. Notice that the back-end with aAUC loss achieves a large improvement in almost all metrics.

Table 10.6: Experimental results on MOBIO [244] eval set, showing *AUC%*, *EER%*, *minCLLR*, *minDCF08*, and *minDCF10*. These results were obtained by training with train set to compare the different losses.

Architecture			Female				
Type	**BB**	**Loss**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
C	FaceNet	*CE*	2.14	99.76	0.086	0.201	0.079
D		*TrLoss*	2.65	99.74	0.102	0.236	0.088
		aAUC	**1.52**	**99.90**	**0.054**	**0.131**	**0.052**
Architecture			**Male**				
Type	**BB**	**Loss**	**EER%**	**AUC%**	**minDCF08**	**minDCF10**	**minCLLR**
C	FaceNet	*CE*	0.59	99.94	0.036	0.154	0.028
D		*TrLoss*	0.62	99.96	0.031	**0.141**	0.026
		aAUC	**0.54**	**99.96**	**0.030**	0.148	**0.025**

10.6 Conclusions

In this chapter, we present a novel optimization procedure to optimize aAUC loss as an alternative to the triplet loss cost. To prove the versatility of this new loss function, we have carried out experiments in different verification tasks.

First, this proposal has been evaluated in the RSR2015-Part I text-dependent speaker verification database. As we show, training the system end-to-end to maximize the AUC performance measure provides better performance in the detection task. Besides, we check the generalization ability of our proposals using RSR2015-Part II, which is a more complicated task. In this second part, we can see how the results achieved follow the same trend as the results of the first part, and they are also competitive compared to the state-of-the-art systems for this Part II.

In addition, we have implemented this new triplet network approach for the language verification task to discriminate similar languages. This approach outperformed the traditional back-ends by up to 17% and 25% relative, in the context of a closed-set evaluation of the LRE 2009 dataset. Besides, this approach has provided relevant results in LRE 2015 and 2017, although the generalization power can be still improved. Results confirm that a back-end technique based on triplet neural networks is an interesting line of research for this task since we have obtained competitive results even though some discrimination problems still exists. Moreover, note that the experimental results have shown that the triplet selection has a big influence in the performance. Especially, the smart triplet selection presented in Algorithm 2 has shown a great improvemente respect to the use of random triplet selection. Thus, we can achieve even better results with an improvement of the language selection combined with the smart triplet selection.

Finally, we have also proven the ability to use this approach for each verification task, including the face verification task. To evaluate the effect of applying the approach proposed in this chapter, we have employed the MOBIO database to develop the face verification system. The experiments for this task have been carried out using exactly

the same configuration for the back-end part as that employed in the text-dependent speaker verification system, including the strategy for the triplet data selection. Therefore, we have demonstrated with these experiments that the proposed aAUC loss can be successfully applied for each verification task.

11

Approximated Detection Cost Function as Training Objective Loss

11.1 Motivation

The use of the traditional loss function for training verification systems based on Deep Neural Network (DNN) architectures does not encourage the discriminative learning of features. Hence, recently, metric learning functions have been alternatively used in training to handle this issue. These metric learning approaches have shown to be very effective techniques to improve the discriminative ability of verification systems. For this reason,

in the previous chapter [3, 7], we proposed an alternative back-end that combines the triplet loss philosophy with the optimization of Area Under the ROC Curve (*AUC*) as a loss function [290–292]. *AUC* provides a performance measure independent of the operating point. The proposed aAUC loss function is optimized in the learning framework to improve the overall system performance in terms of Equal Error Rate (*EER*). However, this kind of metric learning loss functions with pairs or triplets have some drawbacks as slow convergence or instability. Traditionally, to address these problems, sample mining strategies have been applied to select the most informative pairs to create the triplets [75]. This process improves performance, but also involves a high computational cost which slows down excessively the training process. To alleviate this problem, recent research efforts have focused on improving the ability to discriminate by redesigning the traditional classification loss functions while maintaining the low computational cost of these traditional loss functions. Nevertheless, this stream of research does not take into account metrics to measure performance in the verification process.

As a consequence of the aforementioned drawbacks, we have investigated alternative approaches to train the systems. The motivation for this search is to find a new loss function which keeps the low computational cost of the traditional classification loss functions, while the system is trained with a metric focused on the goal task to increase the generalization capability. The goal of the verification task consists of training systems to make a binary decision based on a decision threshold (Ω): acceptance or rejection [218]. Thus, this type of systems produces two types of decision errors which are described in Chapter 3 and depicted in Figure 3.8. False Acceptance (FA) refers to when an impostor identity is incorrectly accepted, and the False Rejection (FR) is related to the incorrect rejection of a true identity [219, 220]. For this reason, apart from using *EER* and *AUC* as metrics, performance in a verification system is obtained by combining the ratio of these two errors, which are defined by the number of times of each one occurs in relation to the number of legitimate or impostor identities [226]. The selection of the threshold, Ω, is what relates the system to the operating point of interest depending on the application, and there are three different kind of cases in function of the total number of errors of each type occurred as we explained in Chapter 3.

Therefore, the aim of this chapter is to replace the classical Cross-Entropy (CE) loss function in an architecture similar to those used for multi-class classification by a new loss function. Due to the relevance of decision errors in the verification process, we propose a new loss function called approximated Detection Cost Function (aDCF) loss which is inspired by Detection Cost Function (DCF) [298, 299] used by the National Institute of Standards and Technology (NIST) during Speaker Recognition Evaluations (SRE). This function measures the cost of decision errors of verification systems in terms of FA and FR and allows the systems based on DNNs to be trained directly to optimize a metric focused on the goal task. We have implemented this approximation of DCF by a differentiable function, which allows the training algorithm to adapt the network parameters to minimize the cost and learn the optimal decision threshold. Unlike recent classification or metric learning loss functions [75, 145, 300], aDCF has the ability to adapt the parameters to modify the optimal threshold and the tradeoff between FA and FR errors to meet

the requirements of a concrete system application. The ability to manage the score distributions in function of the operating point is a useful skill to provide for such end-to-end systems with. Moreover, we have studied the effects of using aDCF loss combined with a cosine layer [8, 9] as the last layer in the DNN architecture instead of a linear layer to train the system. On the other hand, we have also explored the effects of training the verification systems by giving different relevance to both types of errors depending on the application. Finally, we have analyzed the behaviour of using a complementary loss function in combination with CE loss or aDCF loss to improve the discriminative power.

11.2 Approximated Detection Cost Function Loss

Motivated by the idea of taking advantage of the efficiency and speed of multi-class training and, at the same time, the improvement achieved with verification loss functions such as the previous aAUC loss, we have proposed approximated Detection Cost Function (aDCF). This function is inspired by DCF [298, 299], which is one of the main performance metrics in the evaluation process of speaker verification tasks. Using aDCF loss to train the systems, the network learns to minimize this evaluation metric and find the optimal threshold for the specific application. In order to carry out this function, we have to develop an effective and differentiable expression for the original DCF metric.

aDCF loss is composed of a weighted sum of the batch level estimate of the probability of misses or FRR (P_{miss}) and the probability of false alarm or FAR (P_{fa}). P_{miss} is defined by the average number of times the scores of target identities N_{tar} are smaller than the decision threshold (Ω), so that the system cannot effectively detect, and a miss is produced. While P_{fa} is determined by the average number of times the scores of non-target identities N_{non} are greater than Ω, so a false alarm occurs. These two kinds of errors are depicted graphically in Figure 3.8 for each application case. Therefore, as a function of the network parameters θ, P_{fa} and P_{miss} can be written as,

$$P_{fa}(\theta, \Omega) = \frac{\sum_{y_i \in y_{non}} \mathbb{1}(s_\theta(x_i, y_i) > \Omega)}{N_{non}}, \tag{11.1}$$

$$P_{miss}(\theta, \Omega) = \frac{\sum_{y_i \in y_{tar}} \mathbb{1}(s_\theta(x_i, y_i) < \Omega)}{N_{tar}}, \tag{11.2}$$

where N_{tar} is the number of target identities, N_{non} is the number of non-target identities, $\mathbb{1}()$ is equal to *'1'* whenever the score $s_\theta(x_i, y_i)$ meets the condition with respect to Ω, and *'0'* otherwise. The score $s_\theta(x_i, y_i)$ is obtained from the last layer of the neural network where x_i is the input sample with $i \in \{1, ..., m\}$ and m is the number of samples, y_i is the class label. Equations (11.1) and (11.2) can be rewritten using the unit step function $u()$ as,

$$P_{fa}(\theta, \Omega) = \frac{\sum_{y_i \in y_{non}} u(s_\theta(x_i, y_i) - \Omega)}{N_{non}}, \tag{11.3}$$

$$P_{miss}(\theta, \Omega) = \frac{\sum_{y_i \in y_{tar}} u(\Omega - s_\theta(x_i, y_i))}{N_{tar}}. \tag{11.4}$$

However, the expressions (11.3) and (11.4) for the probabilities are not differentiable, so we replace the unit step function $u()$ by a sigmoid of the difference to make an approximation of the binary counter which enables the backpropagation of gradients:

$$\hat{P}_{fa}(\theta, \Omega) = \frac{\sum_{y_i \in y_{non}} \sigma_\alpha(s_\theta(x_i, y_i) - \Omega)}{N_{non}}, \tag{11.5}$$

$$\hat{P}_{miss}(\theta, \Omega) = \frac{\sum_{y_i \in y_{tar}} \sigma_\alpha(\Omega - s_\theta(x_i, y_i))}{N_{tar}}. \tag{11.6}$$

where $\sigma_\alpha()$ is the sigmoid function and is defined as,

$$\sigma_\alpha(s) = \frac{1}{1 + exp(-\alpha \cdot s)}, \tag{11.7}$$

where α is an adjustable parameter. Thus, using these expressions, we can now propose to minimize the following approximated loss function defined as,

$$aDCF(\theta, \Omega) = \gamma \cdot \hat{P}_{fa}(\theta, \Omega) + \beta \cdot \hat{P}_{miss}(\theta, \Omega), \tag{11.8}$$

where γ and β are adjustable parameters to provide more cost relevance to one of the terms over the other. The effect of the values of these parameters with respect to the system application will be studied in the experimental section. For instance, some system applications provide more relevance when a target identity is not detected, while other applications need to decrease the number of non-target identities that the system accepts. Note that Ω will be optimized as part of the system parameters. Moreover, to employ this function efficiently, we have adopted an implementation that follows the same training philosophy of multi-class architectures with DNNs as we detail below.

11.2.1 Relationship between aDCF and CE Loss

To perform the multi-class classification task using DNNs, a widely adopted and efficient approach consists of training the system with CE loss combined with a softmax function. Thus, for one hot labels, CE loss can be written as,

$$L_{CE} = -\frac{1}{m} \sum_{i}^{m} log \frac{\exp\left(s_\theta(x_i, y_i)\right)}{\sum_{j}^{N} \exp\left(s_\theta(x_i, j)\right)}, \tag{11.9}$$

where x_i is the input sample with $i \in \{1, ..., m\}$ and m is the number of samples, y_i is the class label, N is the total number of classes, and $s_\theta(x_i, j)$ is obtained from the last layer of

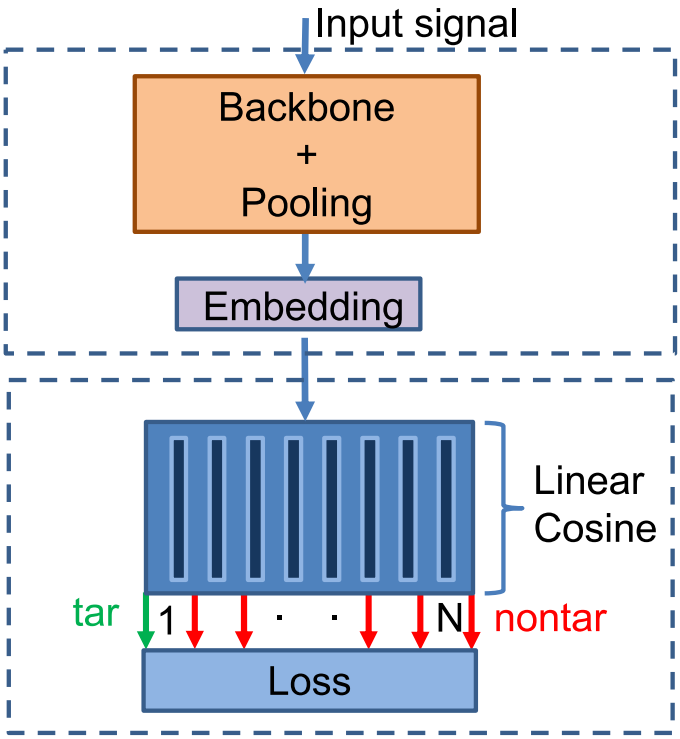


Figure 11.1: Interpretation of the last layer of the neural network as a matrix of weights which models each identity.

the DNN. Typically, the last layer is defined as a linear layer and the score for each class $s_\theta(x_i, y_i)$ is obtained as,

$$s_\theta(x_i, y_i) = W_{y_i}^T \cdot x_i + b_{y_i}, \tag{11.10}$$

where x_i is the input of the last linear layer, $W_{y_i}^T$ is the row of the matrix of weights containing the layer parameters of the class y_i, and b_{y_i} is the bias term. Although in this chapter, we will remove the bias term to simulate the score evaluations as the output of the last layer. The interpretation of the output of this layer as scores is possible since each vector of the matrix of weights used in the last layer can be interpreted as a model for each trained identity and the output as the dot product with the input vector x_i in a similar way as is done to evaluate similarity in verification tasks, as Figure 11.1 depicts.

Despite its accuracy in the training process, CE loss is designed to maximize the posterior probability of the correct class in a multi-class classifier. Thus, this function is appropriate for classification tasks where the goal is to determine the identity of each sample in a known set of identities. However, in verification tasks, the main goal consists of detecting if two samples belong to the same identity or not, so we need to measure the degree of separation and similarity. Therefore, the use of aDCF loss is more consistent with the verification task since it allows to minimize the false alarms for the output dimensions of the neural network that are non-target for which (11.5) provides positive values, so the gradient will tend to reduce this term. And also to reduce the misses when (11.6) term is positive for the output dimension of the neural network that is the same as

the label. As in architecture with CE loss, the key to keep the efficiency for training the system with aDCF loss is the chance to interpret the matrix of weights as a representation of each trained identity and obtain scores to optimize the system during the training progresses with it. Using this approach, DNN is trained with positive samples to rise its score values which minimizes the probability of misses and negative samples to reduce its score values which minimizes the probability of false alarms. Therefore, an example of the learning process can be seen in Figure 11.2 where (11.5) and (11.6) terms characterized by the sigmoid function are minimized during the learning process by gradient backpropagation.

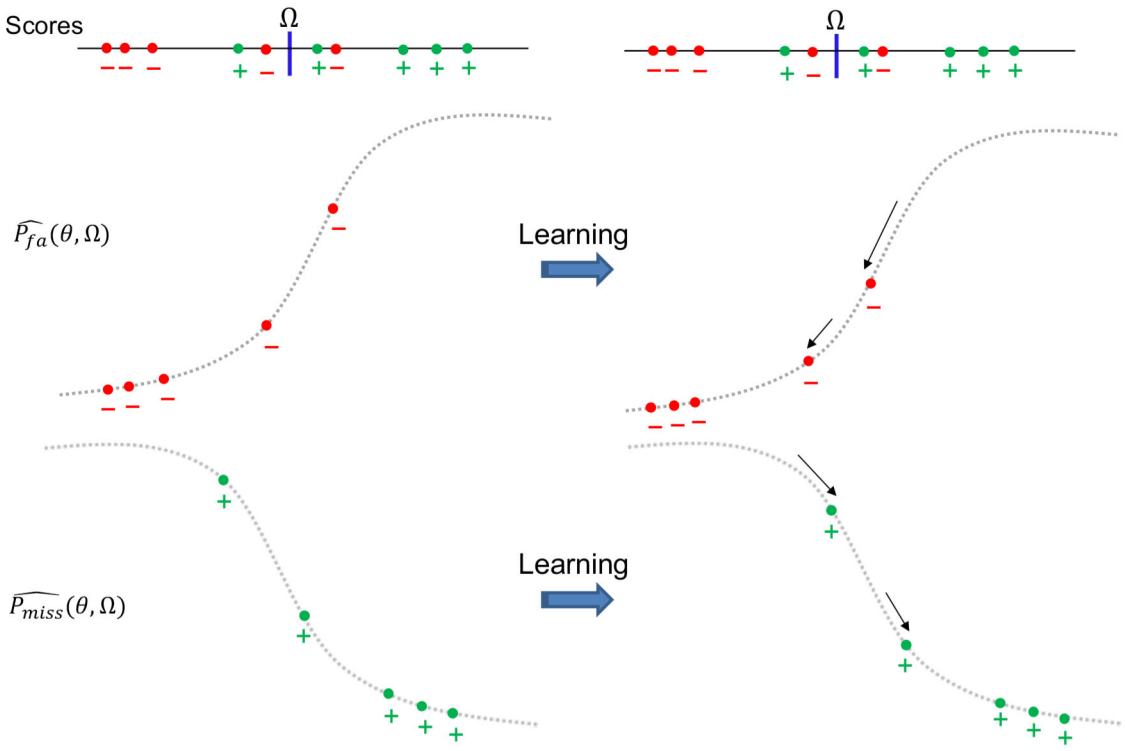


Figure 11.2: aDCF learning process using the sigmoid functions trained with target and non-target examples.

11.2.2 Efficient Implementation

To develop this process efficiently, training is made using subsets of samples, since standard neural network optimization operates with small batches of samples. When the network is trained with a batch size b, the number of N_{tar} and N_{non} that will be taken into account in the expression (11.8) to calculate the gradients will be b, and $b(N-1)$, where N is the total number of speakers. Thus, as we can see in Figure 11.1, the target and non-target scores used in (11.5) and (11.6) are obtained by using the ground truth labels and comparing with the model for each trained identity which is stored in the matrix of weights. Therefore, these assumptions allow us to obtain similar efficiency and convergence speed as the most common approaches to multi-class classification as CE loss during the training process.

11.2.3 Cosine Distance Layer

Previous works [85,301,302] have remarked the fact that there is a gap between the metric used during training (11.10) and the cosine metric employed in the evaluation. Also, these works have shown the relevance of the normalization of features and weights to provide significant performance improvements. Therefore, in this chapter, we have also analyzed the use of a cosine layer instead of a linear layer as the last layer of the neural network to obtain this score as,

$$s_\theta(x_i, y_i) = \frac{W_{y_i}^T \cdot x_i}{\|W_{y_i}^T\| \cdot \|x_i\|}, \tag{11.11}$$

where $\|x_i\|$ is the normalized input signal to the last linear layer, and $\|W_{y_i}^T\|$ is the normalized layer parameters of the speaker class y_i. This cosine layer can be also seen as a composition of a feature normalization combined with a linear layer with weight normalization without the bias term [303] as Figure 11.3 shows. As we will show, the use of the cosine layer combined with aDCF loss achieves better results.

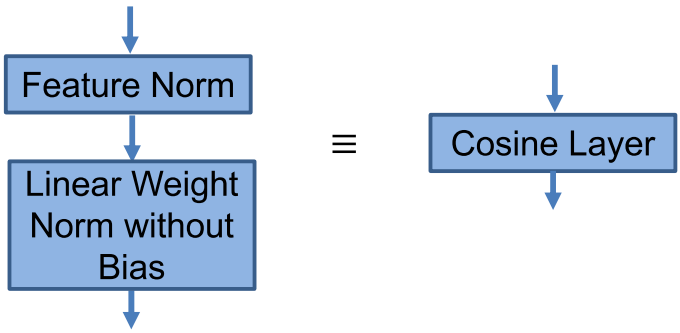
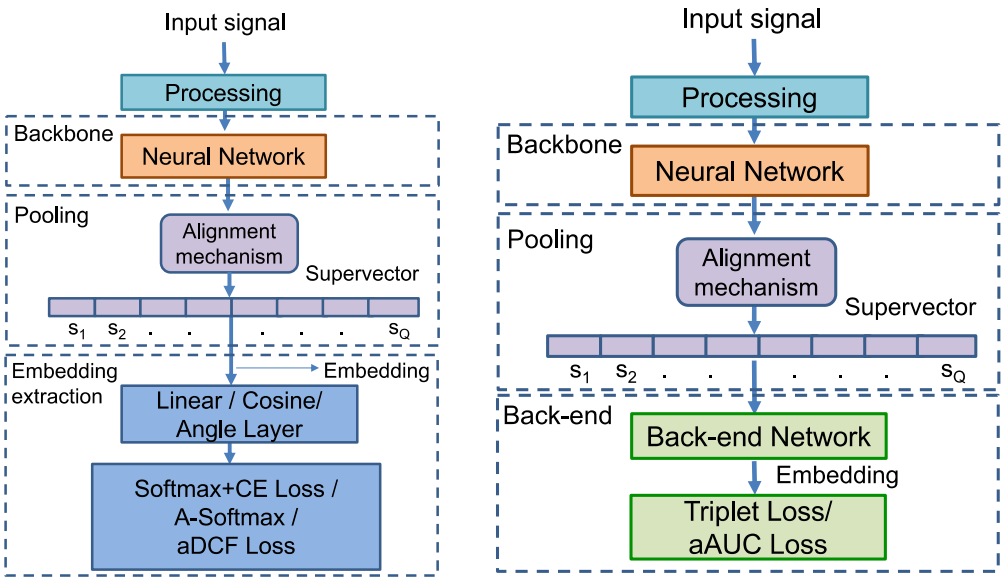


Figure 11.3: Cosine layer is equivalent to a feature norm followed by a linear layer with weight normalization without bias.

11.3 System Employed for Training with aDCF Loss

Following with the architectures used in Chapter 10, the speaker verification system employed to validate this new loss function is practically the same. Figure 11.4 depicts the two architectures used in the experiments. The backbone and pooling parts are shared in both architectures and are described below.

As backbone network, the best configuration found in Chapter 6 using the Bayesian Dark Knowledge (BDK) approach to model the uncertainty in the parameters during the training process is employed. This approach combined with the alignment mechanism based on Gaussian Mixture Model (GMM) with Maximum A Posteriori (MAP) adaptation improves system performance. We introduce this pooling mechanism based on a frame-to-components alignment process instead of using global average pooling, since it allows us to keep the order of the phonetic information of each phrase.



(a) Architecture type C (b) Architecture type D

Figure 11.4: The architectures developed to check the effectiveness of our proposed loss, 11.4(a) the architecture type C is trained for multiclass classification using the alignment mechanisms. In 11.4(b), the architecture type D is trained to optimize the back-end network.

Nevertheless, the last part is different to make the verification process. Architecture C shown in Figure 11.4(a) combines the backbone and pooling with a cosine or linear layer and the loss function to train the system. Moreover, architecture C has been modified with respect to architecture C of the previous chapters to incorporate the possibility of using A-Softmax loss and our aDCF loss instead of CE loss. The configuration of these architectures is detailed in Tables A.16, A.17, A.18 and A.19 in Appendix A. After the training process, verification scores are obtained by applying a cosine similarity on the embeddings. On the other side, architecture *D* depicted in Figure 11.4(b) represents a trainable back-end with Triplet loss or aAUC loss as the objective function. As initialization of this architecture to avoid the convergence issues, a pre-trained model of architecture *C* with CE loss is employed as we did in the experiments of Chapter 10.

11.4 Experiments and Results

11.4.1 Experimental Description

To carry out the experiments in this chapter, RSR2015-Part I and II have been employed. As an alignment mechanism, we have trained a 64 component GMM per phrase without phrase transcription using the bkg partition of each part. From these models, we extract the alignment information for use in the frame-to-component alignment mechanism of each of the DNN architectures, since one model is trained for each different phrase. As input to the DNN, a feature set composed of 20 dimensional Mel-Frequency Cepstral

Coefficients (MFCCs) with their first and second derivates are employed. To train the DNN architecture for each different phrase, the bkg partition has been used. While in this chapter, the dev partition has been employed for normalization and calibration.

In this chapter, several sets of experiments have been developed. First, a set of experiments has been carried out to study the behaviour of the system while different parameter values in aDCF loss are swept. After that, we have analyzed the use of a complementary loss to improve the discrimination ability in combination with CE loss and aDCF loss with the second set of experiments. In the last set of experiments, we have evaluated the system employing some of the most widespread state-of-the-art loss functions to compare the performance with our aDCF loss. These experiments have been performed with score normalization (snorm) [215].

11.4.2 aDCF Parameters α, γ, β

In this section, we analyze the performance of the system when we sweep the aDCF loss parameters which are the terms of the cost relevance (γ, β) and the adjustable parameter in the sigmoid function (α).

Tables 11.1, 11.2 and 11.3 shows Equal Error Rate (*EER%*), minimum Detection Cost Function 10 (*minDCF10*) and actual Detection Cost Function 10 (*actDCF10*) results with Part I for different configurations of parameter values. As we can see, in most of the experiments, the use of a greater value for α parameter improves the results since the value of α modifies the slope of the sigmoid function. Thus, a greater value involves the sigmoid is closer to the unit step, which is used to define the exact *DCF*. Furthermore, we observe that the results are even better when we give more relevance to the probability of false alarms (P_{fa}) during training with the cost term γ. However, we have noted that if this value is too extreme (upper 0.90-0.10), there are some convergence problems during the training process, and the results are more sensitive to the variation of the value of α.

In addition to the above table, Figure 11.5 depicts the Detection Error Trade-off (DET) curves representing the relationship between P_{fa} (FAR) and P_{miss} (FRR). These curves show the best result in terms of *minDCF10* for each configuration of the cost parameters. Note that these representations demonstrate the best results for each configuration of γ and β, and although the results are too similar, we decide to use as a reference system the configuration with better behaviour in all the operating points corresponding to $\gamma = 0.75$, $\beta = 0.25$ and $\alpha = 40$.

The results obtained for RSR2015-Part II are shown in Tables 11.4, 11.5 and 11.6 and Figure 11.6. In this set of experiments, the phrases are shorter and have overlapped lexical content, so it is more challenging than Part I and the general performance is worse. In this case, we observe that when the cost terms are balanced, with an intermediate value of α is enough to achieve the best performance.

Table 11.1: Experimental results on RSR2015-Part I [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These female results were obtained by training with bkg subset and sweeping of the parameter values in aDCF loss with normalization (snorm).

aDCF Parameters			**Female**	
γ	β	α	**EER%**	**min/actDCF10**
0.15	0.85	1	1.22	0.328/0.646
		10	0.57	0.164/0.199
		20	0.45	0.086/0.163
		30	0.34	0.096/0.293
		40	0.43	0.095/0.117
0.25	0.75	1	1.18	0.300/0.772
		10	0.51	0.153/0.163
		20	0.44	0.085/0.118
		30	0.38	0.085/0.088
		40	0.43	0.094/0.210
0.50	0.50	1	1.15	0.311/0.587
		10	0.51	0.115/0.186
		20	0.35	0.093/0.093
		30	0.39	0.088/0.111
		40	0.36	0.072/0.104
0.75	0.25	1	3.81	0.723/0.984
		10	0.42	0.093/0.119
		20	0.44	0.094/0.097
		30	0.36	0.078/0.158
		40	0.33	**0.068**/0.084
0.85	0.15	1	4.69	0.709/0.986
		10	2.32	0.393/0.420
		20	0.34	0.071/**0.073**
		30	**0.32**	0.078/0.096
		40	0.38	0.077/0.080

For illustrative purposes, we have included Figure 11.7 to depict the evolution of aDCF loss with different values of α against the exact DCF during training. This figure shows that the differentiable approximation of DCF is getting close to the real function while the training progresses. Note that this evolution supports the fact that aDCF loss is an effective approximation of the real DCF.

Table 11.2: Experimental results on RSR2015-Part I [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These male results were obtained by training with bkg subset and sweeping of the parameter values in aDCF loss with normalization (snorm).

aDCF Parameters				**Male**
γ	β	α	**EER%**	**min/actDCF10**
0.15	0.85	1	1.19	0.279/0.611
		10	0.76	0.179/0.180
		20	**0.52**	0.132/0.134
		30	0.57	**0.110**/0.119
		40	0.57	0.119/0.121
0.25	0.75	1	0.90	0.245/0.248
		10	0.70	0.170/0.174
		20	0.67	0.140/0.141
		30	0.61	0.122/0.124
		40	0.62	0.116/0.133
0.50	0.50	1	0.91	0.267/0.270
		10	0.64	0.132/0.135
		20	0.54	0.129/0.143
		30	0.64	0.118/0.121
		40	0.63	0.120/0.138
0.75	0.25	1	6.83	0.797/1.000
		10	0.56	0.154/0.214
		20	0.56	0.125/0.158
		30	0.59	0.115/**0.118**
		40	0.55	0.116/0.150
0.85	0.15	1	7.14	0.794/0.874
		10	6.65	0.687/0.699
		20	0.88	0.187/0.445
		30	0.61	0.128/0.226
		40	0.59	0.125/0.225

11.4.3 Last Layer and Ring Loss Study

A second set of experiments has been carried out to observe the system performance when a complementary loss as Ring loss is added to CE loss or aDCF loss. The results of these experiments in Part I (Table 11.7) demonstrate that our aDCF loss does not need a complementary loss function to improve the discrimination ability, while CE loss needs

Table 11.3: Experimental results on RSR2015-Part I [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These female+male results were obtained by training with bkg subset and sweeping of the parameter values in aDCF loss with normalization (snorm).

aDCF Parameters			**Female+Male**	
γ	β	α	**EER%**	**min/actDCF10**
0.15	0.85	1	1.21	0.312/0.322
		10	0.70	0.204/0.210
		20	0.52	0.137/0.148
		30	0.51	0.139/0.141
		40	0.55	0.124/0.132
0.25	0.75	1	1.04	0.272/0.516
		10	0.62	0.180/0.189
		20	0.60	0.133/0.133
		30	0.55	0.128/0.146
		40	0.59	0.126/0.206
0.50	0.50	1	0.63	0.146/0.163
		20	0.51	0.119/0.128
		30	0.58	0.128/0.130
		40	0.56	**0.115**/0.123
0.75	0.25	1	6.59	0.797/0.999
		10	0.58	0.145/0.154
		20	0.55	0.131/0.134
		30	0.55	0.123/0.140
		40	**0.50**	0.117/**0.119**
0.85	0.15	1	6.79	0.781/0.999
		10	4.25	0.525/0.684
		20	0.68	0.157/0.159
		30	0.54	0.132/0.135
		40	0.57	0.118/0.120

it. Furthermore, when we employ aDCF loss to train the system, the use of a cosine layer as last layer is the most suitable option, since this cosine metric employed during the training process is the same metric employed by the evaluation process to obtain the final verification scores. Thus, we can see better results when the training process used is a pipeline more similar to the final verification process.

When these experiments are carried out for Part II, we observe in Table 11.8 that the effect of using a complementary loss and a cosine layer instead of a linear layer follows a

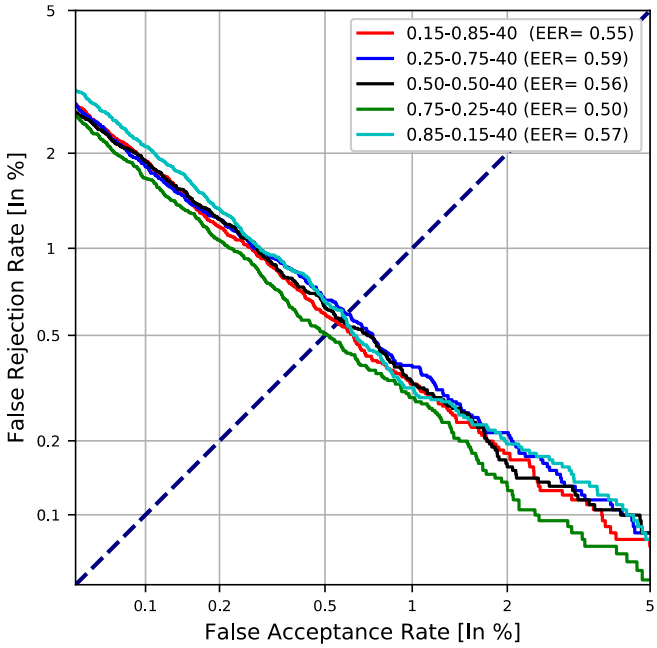


Figure 11.5: DET curves for female+male results on RSR2015-Part I varying the parameter values of aDCF loss γ and β, and using the best α value in each case.

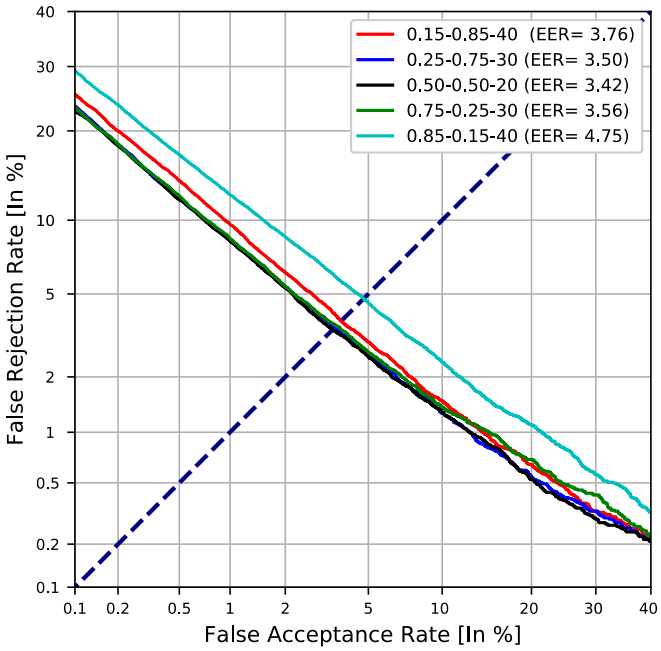


Figure 11.6: DET curves for female+male results on RSR2015-Part II varying the parameter values of aDCF loss γ and β, and using the best α value in each case.

similar trend as the results of Part I. However, we have checked that CE loss without Ring loss with this set of data provides similar results to those obtained with Ring loss. The difficult in this dataset may cause that the system can not bring together correctly some features from the same identity even though Ring loss is applied and for this reason, the results do not improve so clearly when applying this complementary function.

Table 11.4: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These female results were obtained by training with bkg subset and sweeping of the parameter values in aDCF loss with normalization (snorm).

aDCF Parameters			**Female**	
γ	β	α	**EER%**	**min/actDCF10**
0.15	0.85	1	5.22	0.746/0.839
		10	3.24	0.588/0.602
		20	2.66	0.478/0.491
		30	2.59	0.434/0.446
		40	2.83	0.456/0.463
0.25	0.75	1	4.18	0.719/0.755
		10	2.84	0.502/0.541
		20	2.82	0.443/0.481
		30	2.67	0.428/0.446
		40	2.70	0.460/0.463
0.50	0.50	1	4.13	0.648/0.658
		10	2.76	0.489/0.518
		20	**2.47**	0.424/0.467
		30	2.54	0.425/**0.432**
		40	3.83	0.446/0.452
0.75	0.25	1	19.55	0.983/1.000
		10	2.84	0.486/0.529
		20	2.61	0.443/0.467
		30	2.57	0.437/0.448
		40	2.65	**0.422**/**0.432**
0.85	0.15	1	23.47	0.964/1.000
		10	18.08	0.940/0.995
		20	4.36	0.604/0.955
		30	3.29	0.475/0.680
		40	2.95	0.458/0.466

11.4.4 Comparison with State-of-the-Art Loss Functions

In the last set of experiments, we have performed a comparison among the best configurations of aDCF loss for Part I and Part II and some of the most extended loss functions in the state-of-the-art, which are CE loss combined with Ring loss (CE+RL), Triplet loss (Trloss), aAUC loss, and A-Softmax loss. In Table 11.9, we show the results of these exper-

Table 11.5: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These male results were obtained by training with bkg subset and sweeping of the parameter values in aDCF loss with normalization (snorm).

aDCF Parameters				**Male**
γ	β	α	**EER%**	**min/actDCF10**
0.15	0.85	1	12.05	0.817/1.000
		10	7.17	0.721/0.771
		20	6.93	0.671/0.770
		30	4.54	0.615/0.626
		40	4.39	0.588/0.688
0.25	0.75	1	6.56	0.741/0.921
		10	4.96	0.624/0.626
		20	4.15	0.576/0.609
		30	**4.08**	0.583/0.587
		40	4.20	0.611/0.705
0.50	0.50	1	5.24	0.725/0.727
		10	**4.08**	0.606/0.628
		20	4.10	**0.568/0.570**
		30	4.21	0.596/0.598
		40	4.21	0.576/0.593
0.75	0.25	1	22.32	0.988/1.000
		10	4.95	0.633/0.645
		20	4.18	0.579/0.598
		30	4.17	**0.568/0.570**
		40	4.15	0.609/0.619
0.85	0.15	1	26.11	0.988/1.000
		10	21.69	0.974/1.000
		20	7.93	0.755/0.759
		30	5.92	0.697/0.705
		40	5.77	0.658/0.674

iments with Part I. We can observe that architecture *C* trained using aDCF loss achieves the best results. Especially relevant is the improvement at comparing with CE loss and A-Softmax loss, since both have been trained using the most comparable strategy with architecture *C*, which allows us to keep the same efficiency for the training process. In terms of relative improvement, *EER%* and *minDCF10* values have improved by 30.6% and 26.4% with respect to CE loss, and by 28.6% and 34.3% with respect to A-Softmax. This

Table 11.6: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These female+male results were obtained by training with bkg subset and sweeping of the parameter values in aDCF loss with normalization (snorm).

aDCF Parameters			**Female+Male**	
γ	β	α	**EER%**	**min/actDCF10**
0.15	0.85	1	8.93	0.786/1.000
		10	5.42	0.666/0.744
		20	5.41	0.588/0.685
		30	3.75	0.537/0.543
		40	3.76	0.530/0.539
0.25	0.75	1	5.51	0.733/0.771
		10	4.10	0.566/0.570
		20	3.60	0.521/0.523
		30	3.50	0.514/0.516
		40	3.62	0.543/0.545
0.50	0.50	1	4.75	0.693/0.701
		10	3.51	0.551/0.575
		20	**3.42**	**0.506/0.515**
		30	3.54	0.518/0.528
		40	4.03	0.519/0.522
0.75	0.25	1	21.91	0.971/1.000
		10	4.05	0.566/0.613
		20	3.64	0.527/0.530
		30	3.56	0.514/0.520
		40	3.58	0.526/0.531
0.85	0.15	1	25.16	0.977/1.000
		10	20.14	0.968/1.000
		20	6.88	0.693/0.756
		30	4.99	0.595/0.653
		40	4.75	0.567/0.572

improvement is remarkable given that the models involved have the same number of parameters and the inference time does not increase.

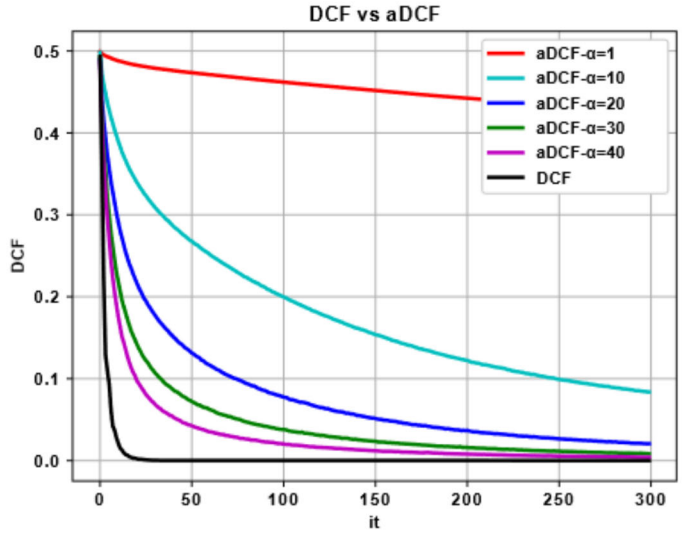


Figure 11.7: Evolution training aDCF with different α values and exact DCF.

Table 11.7: Experimental results on RSR2015-Part I [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These results were obtained by training with bkg subset to analyze the behaviour using a complementary loss with normalization (snorm).

Architecture			**Female**	
Layer	**Loss**	**Ring**	**EER%**	**min/actDCF10**
Linear	CE	yes	**0.47**	**0.114/0.149**
		no	0.86	0.240/0.255
Cosine		yes	0.73	0.217/0.243
		no	0.92	0.282/0.375
Linear	aDCF	yes	1.39	0.320/0.764
		no	1.77	0.430/0.684
Cosine		yes	0.34	0.085/0.140
		no	**0.33**	**0.068/0.084**

Architecture			**Male**	
Layer	**Loss**	**Ring**	**EER%**	**min/actDCF10**
Linear	CE	yes	**0.83**	**0.177/0.189**
		no	1.00	0.243/0.258
Cosine		yes	1.51	0.337/0.379
		no	0.91	0.245/0.288
Linear	aDCF	yes	2.23	0.487/0.939
		no	2.32	0.455/0.998
Cosine		yes	0.68	0.129/0.137
		no	**0.55**	**0.116/0.150**

Architecture			Female+Male	
Layer	**Loss**	**Ring**	**EER%**	**min/actDCF10**
Linear	CE	yes	**0.72**	**0.159/0.165**
		no	0.96	0.259/0.269
Cosine		yes	1.21	0.289/0.296
		no	0.94	0.264/0.314
Linear	aDCF	yes	2.09	0.420/0.768
		no	2.08	0.447/0.953
Cosine		yes	0.60	0.125/0.145
		no	**0.50**	**0.117/0.119**

Table 11.8: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These results were obtained by training with bkg subset to analyze the behaviour using a complementary loss with normalization (snorm).

Architecture			Female	
Layer	**Loss**	**Ring**	**EER%**	**min/actDCF10**
Linear	CE	yes	3.31	**0.550/0.563**
		no	**3.20**	0.569/0.608
Cosine		yes	4.14	0.663/0.773
		no	3.68	0.641/0.647
Linear	aDCF	yes	8.69	0.823/0.832
		no	10.92	0.901/0.976
Cosine		yes	3.47	0.512/0.536
		no	**2.47**	**0.424/0.467**

Architecture			Male	
Layer	**Loss**	**Ring**	**EER%**	**min/actDCF10**
Linear	CE	yes	5.45	0.690/0.833
		no	**5.03**	**0.689**/0.969
Cosine		yes	8.76	0.804/0.872
		no	5.24	0.719/**0.732**
Linear	aDCF	yes	12.57	0.883/0.890
		no	9.51	0.841/1.000
Cosine		yes	5.15	0.638/0.788
		no	**4.10**	**0.568/0.570**

Architecture			Female+Male	
Layer	**Loss**	**Ring**	**EER%**	**min/actDCF10**
Linear	CE	yes	4.58	**0.627/0.645**
		no	**4.25**	0.639/0.717
Cosine		yes	6.71	0.745/0.771
		no	4.55	0.689/0.718
Linear	aDCF	yes	10.97	0.857/0.874
		no	10.94	0.882/1.000
Cosine		yes	4.43	0.580/0.633
		no	**3.42**	**0.506/0.515**

Table 11.9: Experimental results on RSR2015-Part I [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These results were obtained by training with bkg subset to compare the different loss functions with normalization (snorm).

Architecture		Female	
Loss	**Type**	**EER%**	**min/actDCF10**
CE+RL	C	0.47	0.114/0.149
Trloss	D	0.78	0.161/0.559
aAUC	D	0.47	0.103/0.142
A-Softmax	C	0.68	0.156/0.263
aDCF	C	**0.33**	**0.068/0.084**

Architecture		Male	
Loss	**Type**	**EER%**	**min/actDCF10**
CE+RL	C	0.83	0.177/0.189
Trloss	D	0.96	0.174/0.319
aAUC	D	0.74	0.157/0.170
A-Softmax	C	0.70	0.163/0.187
aDCF	C	**0.55**	**0.116/0.150**

Architecture		Female+Male	
Loss	**Type**	**EER%**	**min/actDCF10**
CE+RL	C	0.72	0.159/0.165
Trloss	D	0.88	0.173/0.261
aAUC	D	0.64	0.137/0.157
A-Softmax	C	0.70	0.178/0.187
aDCF	C	**0.50**	**0.117/0.119**

Moreover, we have also added Figure 11.8 with the corresponding DET curves. These curves show results of the female+male experiments. These representations demonstrate that the best system performance for all operating points is obtained for architecture *C* trained with aDCF loss. On the other hand, note that A-Softmax loss was proposed in the literature as a better approach to replace CE loss, but in this case, the system trained using CE loss combined with Ring loss achieves better overall performance. In addition, we can observe that the whole performance in the system using aAUC loss is better than the performance in the system with A-Softmax loss.

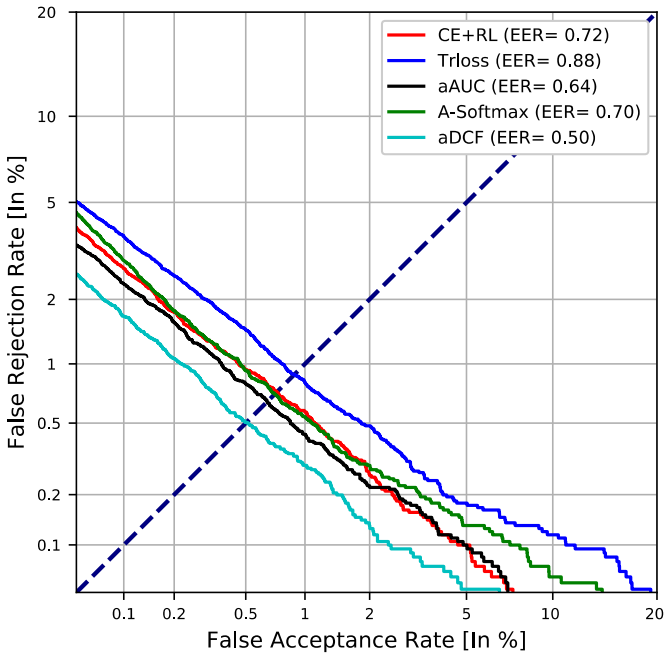


Figure 11.8: DET curves for female+male results on RSR2015-Part I using different loss functions.

The results of these experiments using Part II are shown in Table 11.10 and Figure 11.9. As in the experiments with Part I, the system trained with aDCF loss has the best results with a relative improvement in *EER%* and *minDCF10* of 25.3% and 19.3% with respect to CE loss, and 2.6% and 14.1% compared to A-Softmax. In addition, it is worth noting that the DET curves show that the system trained with aDCF loss obtains the best behaviour for low FAR operating points, while when FRR is low, the behaviour is similar to the system trained with A-Softmax.

Table 11.10: Experimental results on RSR2015-Part II [124] eval subset, showing *EER%*, *minDCF10* and *actDCF10*. These results were obtained by training with bkg subset to compare the different loss functions with normalization (snorm).

Architecture		**Female**	
Loss	**Type**	**EER%**	**min/actDCF10**
CE+RL	C	3.31	0.550/0.563
Trloss	D	3.75	0.542/0.603
aAUC	D	2.76	0.503/0.527
A-Softmax	C	2.79	0.511/0.575
aDCF	C	**2.47**	**0.424/0.467**

Architecture		**Male**	
Loss	**Type**	**EER%**	**min/actDCF10**
CE+RL	C	5.45	0.690/0.833
Trloss	D	5.53	0.621/0.655
aAUC	D	4.62	0.601/0.760
A-Softmax	C	**4.01**	0.655/0.666
aDCF	C	4.10	**0.568/0.570**

Architecture		**Female+Male**	
Loss	**Type**	**EER%**	**min/actDCF10**
CE+RL	C	4.58	0.627/0.645
Trloss	D	4.66	0.583/0.603
aAUC	D	4.25	0.579/0.704
A-Softmax	C	3.51	0.589/0.610
aDCF	C	**3.42**	**0.506/0.515**

11.4.5 Impact of the Score Normalization

In this section, we analyze the results without score normalization and calibration. To carry out this study, we only compare the same type of architectures: architecture *C*. Table 11.11 obtained *EER%* and *minDCF10* results for systems with and without *snorm*. For systems with snorm, aDCF achieves the best performance in Parts I and II. For systems without snorm in Part I, aDCF and A-Softmax present similar results without using score normalization. While the relative improvement in both metrics with respect to CE loss is higher than 50%. However, for the rows corresponding to Part II, A-Softmax loss outperforms aDCF loss. In the following, we further analyze the results without *snorm* by checking the performance for each phrase.

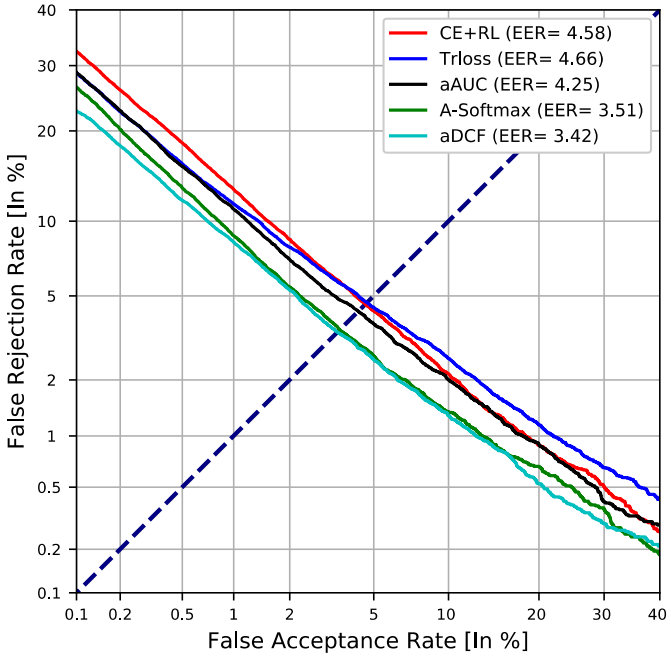


Figure 11.9: DET curves for female+male results on RSR2015-Part II using different loss functions.

Table 11.11: Results in terms of *EER%* and *minDCF10* for RSR2015-Part I and Part II [124] eval subset (female and male) with and without normalization (snorm).

	Architecture		**Without SNORM**		**With SNORM**	
Dataset	**Loss**	**Type**	**EER%**	**minDCF10**	**EER%**	**minDCF10**
RSR-Part I	CE+RL	C	1.87	0.373	0.72	0.159
	A-Softmax		0.85	0.171	0.70	0.178
	aDCF		0.82	0.174	**0.50**	**0.117**
RSR-Part II	CE+RL	C	9.64	0.964	4.58	0.627
	A-Softmax		5.16	0.800	3.51	0.589
	aDCF		6.56	0.876	**3.42**	**0.506**

Table 11.12 shows the average improvement of aDCF loss against CE+RL/A-Softmax in terms of *minDCF10* for the results without normalization. Positive values indicate that aDCF loss performs better regarding the other loss. This time we evaluate the phrases individually. Note that, at the phrase level, the system with aDCF loss achieved the best performance for both parts of the dataset. To explain this result, note that we have trained a model per phrase, so with aDCF loss, the model learns to obtain the best score distributions for each phrase. Thus, when we evaluate separately the phrases, the obtained scores have one optimal threshold by each. However, at joining together all scores (as in Table 11.10), we simulate how a decision system would work and there is a single threshold for all of them. The application of a single decision threshold is non-optimal for all phrases and this causes the drop in performance observed in Table 11.11.

Table 11.12: Average improvement of aDCF vs. CE+RL/A-Softmax without normalization (snorm) in terms of *minDCF10* by phrase on RSR2015-Part I and Part II [124] eval subset.

	minDCF10(%Improv.)	
	RSR-Part I	**RSR-Part II**
aDCF vs CE+RL	9.39	14.85
aDCF vs A-Softmax	13.41	7.36

11.5 Conclusions

In this chapter, we have conducted a wide analysis of the use of a metric learning approach based on aDCF loss. This loss function is an approximate measurement of the decision errors FAR and FRR in verification systems which allows end-to-end systems to optimize a metric used in the final verification process. To employ this loss function, we have used an efficient implementation that follows the philosophy of existing multi-class loss functions and allows us to take advantage of the interpretation of the matrix of weights of the last layer to directly obtain scores as training progress.

Experiments to evaluate the effectiveness of our approach are carried out on the RSR2015-Part I and Part II text-dependent speaker verification database. The results obtained by studying the sweep of aDCF loss parameters have shown a great performance on both datasets, and also that in function of the part of the database, the best parameter configuration is different. Furthermore, we have checked the improvement achieved when aDCF loss is combined with a cosine distance layer as the last layer in the DNN instead of the usual linear layer. It has also been observed that aDCF loss does not need the use of a complementary loss to improve the discrimination ability, while CE loss improves considerably with it. Finally, we have compared the results obtained using aDCF loss with some of the state-of-the-art approaches, and aDCF loss outperforms all of them with relative improvements bigger than 10% in the *EER* and DCF metrics.

12

Training Enrollment Models by Network Optimization

12.1 Motivation

As we have emphasized throughout this thesis, verification is a binary problem that consists of determining whether two different files belong to the same identity or different. Hence, systems should be trained to directly perform this verification process between the two files. These two files are widely known as enrollment file and test file. However, most current verification systems are trained for multi-class classification to obtain a representation for each of these files. This type of approach does not take into account the goal of the verification task to train discriminative embeddings. Therefore, after extracting the embeddings, a back-end is applied to obtain the final verification scores. This

back-end can be a simple similarity metric or a more sophisticated method that usually involves a more complex training process.

In pioneer Deep Neural Network (DNN) works with this framework, it was supposed that successful classification models would be able to achieve great results on different test data without the need for any back-end technique. Nevertheless, test data can have different variability than training data, so it may not be always possible to generalize properly on unseen data. For that reason, recently, Cross-Entropy (CE) loss has been substituted by different variants of classification loss functions to increase the discrimination ability [8,9,81]. On the other hand, different back-end approaches based on metric learning techniques are increasing as a relevant research focus since these approaches allow to make the training process more appropriate to the evaluation procedure, such as the approach based on triplet neural networks with *aAUC* loss proposed in Chapter 10 [3,7]. However, these approaches are very sensitive to the selection of training data to create the pairs or triplets, and this process also involves slow convergence and high computational cost. To alleviate these drawbacks, in Chapter 11 [8,9], we proposed aDCF loss which is an alternative to CE loss, and at the same time, it is a more suitable loss function for the verification task since this function is inspired by one of the main verification metrics employed. For the text-dependent speaker verification task, the use of aDCF loss for training has proven to be an effective approach. Nonetheless, as we observed in Chapter 11 [8,9], this approach was trained using a model for each phrase, so it suffers when the whole performance is given and score normalization is not applied.

Following the idea of improving the discrimination ability with back-ends based on metric learning techniques and to solve previous issues, instead of using a complex back-end, this chapter presents a novel and straightforward approach to perform the verification process. This approach is based on training enrollment models with a learnable vector for each identity. These models will be optimized using the enrollment data of each identity and taking advantage of the information learned in a main trained network from the training identity set which allows the system to be more robust in the test stage. The use of this information is possible since the matrix from the last layer in DNN models can be interpreted as an embedding dictionary where the identities of the training data are stored. Using this approach, we can consider the information stored in the last layer as competing identities, therefore negative examples, and the enrollment data represent positive examples. This process has to be carried out for each enrollment identity to produce a learned vector, which will be separated in terms of the detection metric from the training identities. The process is extremely efficient, as there is no need to select hard negatives, and only some learnable parameters are optimized while the rest of the network is frozen. To train the whole system, we optimize aDCF loss presented in Chapter 11, which is more appropriate for the verification task. Furthermore, this function can be easily incorporated into the new verification process to train the enrollment models since it is composed of two loss contributions to reduce false alarms, meaning the enrollment data is not similar to the training identities, and also to reduce misses, so that the enrollment data is similar enough to the learned model. Moreover, this kind of back-end allows us to directly obtain the final verification scores and at the same time, whether it

would be necessary, we could store the enrollment model of each identity to use them in a next step, as we will see in Chapter 13.

12.2 Training Enrollment Model

As mentioned, many verification systems apply a back-end approach to perform the final verification process. In DNN systems, the back-end is employed on the intermediate representations extracted from the previously trained DNN architecture. This back-end can be a simple cosine similarity [74] in which the verification scores when each target identity has more than one enrollment file are obtained by averaging all enrollment embeddings as,

$$e_{avg}^{tar} = \frac{1}{m} \sum_{i=1} e_i^{tar}, \tag{12.1}$$

where e_i^{tar} is the enrollment or target data. However, nowadays, some of the back-end methods employed [145] to perform the verification process are complex and strong. Most of them usually require a careful selection process of the input data, which makes these methods very sensitive to this process and increases the computational time.

Therefore, to reduce these disadvantages, this section proposes a novel approach to carry out the enrollment process in a verification system taking advantage of the information modelled during the initial training phase with aDCF loss developed in Chapter 11 to improve the system performance. Figure 12.2 shows that this initial training is termed in this chapter as the training phase. While the training of the back-end approach is called as the enrollment phase. The final evaluation is referred to as the testing phase.

To address the issue of data selection in this approach, we employ the matrix from the last layer of the initially trained architecture combined with the enrollment data to mimic the target/non-target process which is carried out in the verification task. In Figure 12.1, we interpret the matrix obtained from the last layer during the training phase as an embedding dictionary, since each row learns as the training progresses a representation of the identity information that is correctly classified when the embedding is multiplied by the corresponding row. Therefore, we can see each row weight as an embedding which represents a different identity.

Once this matrix is well-trained in the training phase, we pass to the second stage, which is the enrollment phase represented in Figure 12.2b). In this phase, we add a new learnable vector w_e for each enrollment identity that will also be evaluated similarly to (11.11) as we will see later. To initialize this layer, we have employed two different alternatives. First, we define random values for vector w_e, while in the other option, we initialize this vector using an averaging of the embeddings of the enrollment data of each identity which is obtained as,

$$w_e^0 = e_{avg}^{tar}, \tag{12.2}$$

where e_{avg}^{tar} is the average of the embeddings of the enrollment data obtained as in (12.1).

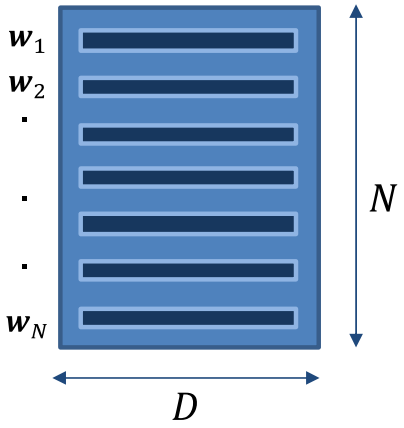


Figure 12.1: Embedding dictionary from the last layer in the training phase, where each row represents one of the N train speakers and D is the dimension of the embedding.

Moreover, during the training of the enrollment model, the same aDCF loss function that we have employed in the training phase is optimized. In this specific case of use of aDCF loss, we define the probability of false alarms and misses directly without the need of labels since we have a binary task with target and non-target data separately. Hence, (11.5) and (11.6) terms can be written as,

$$\hat{P}_{fa}(\theta, \Omega) = \frac{\sum_j \sigma_\alpha(s_\theta(e_j^{non}) - \Omega)}{N_{non}}, \tag{12.3}$$

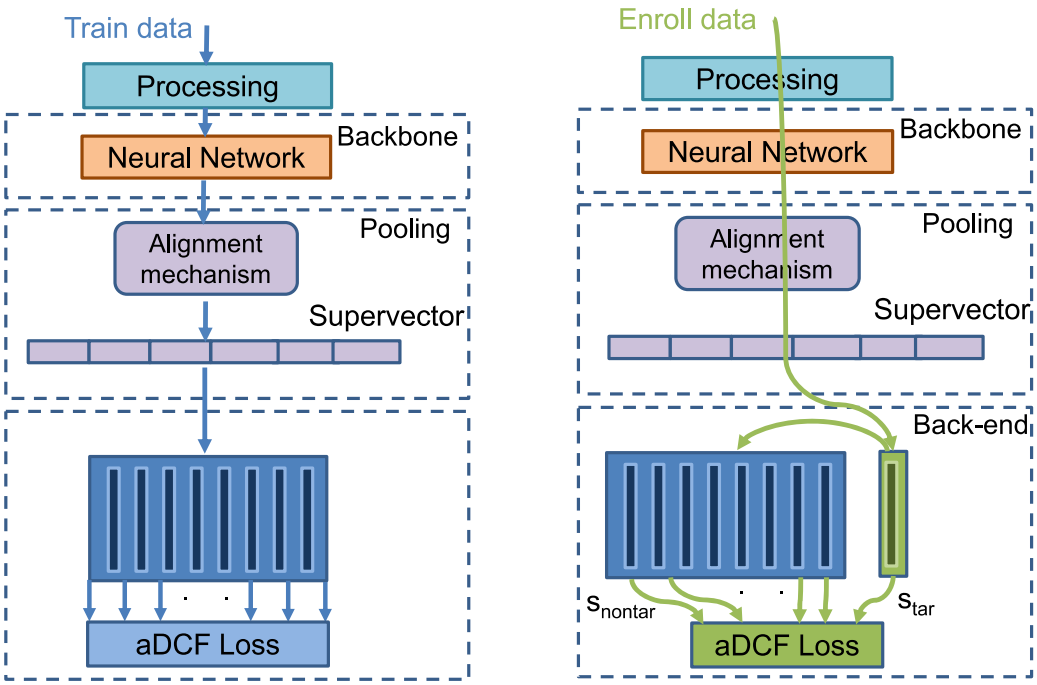
$$\hat{P}_{miss}(\theta, \Omega) = \frac{\sum_i \sigma_\alpha(\Omega - s_\theta(e_i^{tar}))}{N_{tar}}. \tag{12.4}$$

where N_{tar} is the number of target identities, N_{non} is the number of non-target identities, the score $s_\theta(e_j^{non})$ is the non-target score where e_j^{non} is the embedding of each non-target identity with $j \in \{1, ..., m_r\}$ where m_r is each row of the matrix from the last layer of the neural network, and the score $s_\theta(e_i^{tar})$ is the target score where e_i^{tar} is each enrollment embedding with $i \in \{1, ..., m_t\}$ and m_t is the number of target samples. To optimize this function now, the score can be expressed as,

$$s_\theta(e_i) = \frac{e_i \cdot w_e^T}{\|e_i\| \cdot \|w_e^T\|}, \tag{12.5}$$

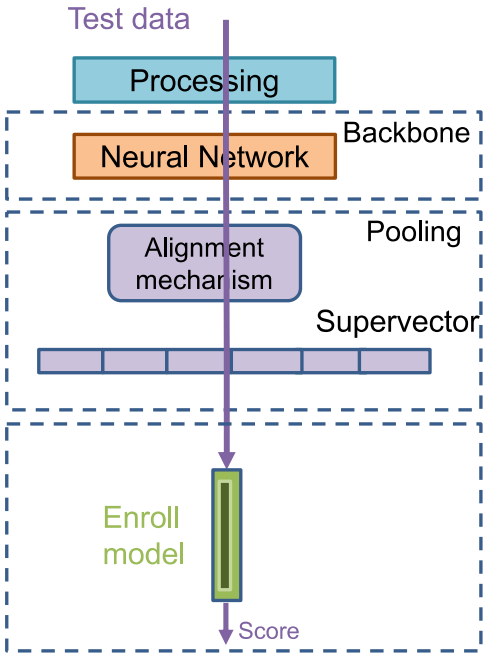
where $\|e_i\|$ is the normalized input to the enrollment model, and $\|w_e^T\|$ is the normalized layer parameters of the embedding obtained from the enrollment file. Using this expression, we obtain the scores of the enrollment files or targets s_{tar} comparing with enrollment model, and the scores of the embedding dictionary or non-targets s_{non} are computed comparing with the enrollment model, which are directly used to optimize aDCF loss. Therefore, the aDCF loss employed to train the enrollment models is obtained as in (11.8) combining these expressions to define the approximated loss function. The optimization using this function is possible since it can be seen as a loss designed to minimize one-versus-all confusions for negatives or training identities and maximize similarity for positives or enrollment data. Thus, we optimize the cost of classifying the

enrollment utterances as the correct enrollment identity avoiding the similarity to the stored models from the training set.



(a) Training phase

(b) Enrollment phase



(c) Testing phase

Figure 12.2: (a) Left: Training phase, where the last layer can be seen as an embedding dictionary of the training identities. (b) Right: Enrollment phase, where an enrollment model is trained for each target identity. (c) Bottom: Testing phase, where test data is compared with each enrollment model and the verification scores are obtained.

For the testing phase shown in Figure 12.2c), the test data is compared with each enrollment model trained during the enrollment phase, and we obtain directly the verification scores without the need of using another external metric. We also have to note that this procedure does not affect the efficiency or the computation cost at test time.

12.3 Supervector Neural Network combined with Enrollment Back-end as System

In the following section, we briefly describe the architecture of the system developed to apply this novel approach for text-dependent speaker verification, which is depicted in Figure 12.2. The backbone and pooling parts are shared during the three phases which compose our whole system. As we explained in the previous chapters, the backbone part employed is the best configuration of Chapter 6 for training one model for each phrase in this task. Moreover, the alignment method used as pooling part is also the best option found in Chapter 6, a GMM combined with a MAP adaptation [106]. In addition, the last layer that acts as an embedding dictionary during the enrollment phase is a cosine layer since it was proven in Chapter 11 that it is better than using a linear layer. Finally, to train this system, aDCF loss proposed in Chapter 11 is applied as objective loss function.

12.4 Experiments and Results

12.4.1 Experimental Setup

The experiments developed in this chapter have been performed with RSR2015-Part II. As input to train the alignment mechanism and as input to the DNN, we have employed a 20 dimensional Mel-Frequency Cepstral Coefficients (MFCCs) stacked with their first and second derivatives. Furthermore, a 64 component GMM has been trained per phrase. From these models, the alignment information is extracted to use in the alignment mechanism of our architecture. The bkg partition of the database has been used to train the alignment mechanism and the DNN architecture.

To show the potential and the behaviour of the new approach proposed in this chapter, the following set of experiments was carried out. The performance obtained using the architecture after the training phase to extract embeddings and applying a cosine similarity ($Cosine$) is compared to the results achieved with the training enrollment models approach ($EnrollModel$) proposed with two different alternatives for the layer initialization. The first alternative consists of a totally random initialization ($rand$), and the other alternative is initialized with an averaging of the enrollment embeddings (avg).

12.4.2 Results with RSR2015-Part II

Table 12.1 presents Equal Error Rate (*EER%*), minimum Detection Cost Function 10 (*minDCF10*), actual Detection Cost Function 10 (*actDCF10*), Log-Likelihood Ratio Cost (*CLLR*), and minimum Log-Likelihood Ratio Cost (*minCLLR*). We can observe that the proposed approach for the verification process with the two different initializations outperforms the baseline using a cosine similarity directly over the average of the embeddings extracted from the architecture without applying any other back-end technique. Furthermore, we can see as a good initialization leads the enrollment training to better performance, but a random initialization also improves the baseline results. This fact reflects that training specific enrollment models for each enroll speaker helps to improve the discriminative ability, and therefore the text-dependent speaker verification process.

Table 12.1: Experimental results on RSR2015-Part II [124] eval set, showing *EER%*, *CLLR*, *minCLLR*, *actDCF10* and *minDCF10*. These results were obtained by training with bkg subset to compare the approach proposed with the two alternatives as initialization and the cosine baseline.

Back-end		**Female**		
Type	**Init**	**EER%**	**min/actDCF10**	**minCLLR/CLLR**
Baseline (Cosine)	–	4.19	0.72/0.78	0.159/0.164
Enroll Model	rand	3.77	0.74/0.77	0.143/0.147
Enroll Model	avg	**3.52**	**0.69**/**0.72**	**0.132**/**0.135**
Improvement (%)		**15.99**	**4.17**/**7.69**	**16.98**/**17.68**

(a) Female results

Back-end		**Male**		
Type	**Init**	**EER%**	**min/actDCF10**	**minCLLR/CLLR**
Baseline (Cosine)	–	5.80	0.91/1.02	0.218/0.228
Enroll Model	rand	5.42	0.89/0.92	0.204/**0.213**
Enroll Model	avg	**5.22**	**0.86**/**0.89**	**0.196**/0.228
Improvement %		**10.00**	**5.49**/**12.75**	**10.09**/**6.58**

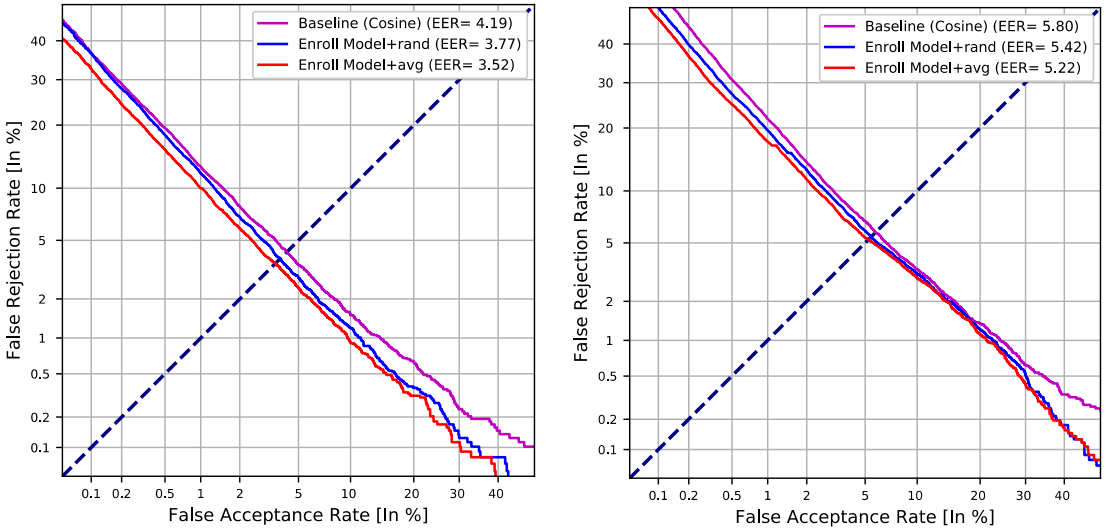
(b) Male results

Back-end		**Female+Male**		
Type	**Init**	**EER%**	**min/actDCF10**	**minCLLR/CLLR**
Baseline (Cosine)	–	5.10	0.85/0.93	0.193/0.201
Enroll Model	rand	4.72	0.83/0.85	0.180/0.185
Enroll Model	avg	**4.46**	**0.79**/**0.82**	**0.170**/**0.174**
Improvement (%)		**12.55**	**7.06**/**11.83**	**11.92**/**13.43**

(c) Female+Male results

Moreover, whether we pay the attention in the difference between the values of the optimal DCF (*minDCF10*) and CLLR (*minCLLR*) with their correspondent actual value, we note that both alternatives for the training of the enrollment models have a smaller difference between those values than using the cosine.

In addition, Figure 12.3 represents the Detection Error Trade-off (DET) curves. These curves are grouped by gender to show better the results for each part of the Table 12.1. These representations clearly demonstrate that the training of the enrollment models with both initializations have a great performance in both subsets (female and male). In Figure 12.3a), we can see the DET curves of female results where the three results follow the same trend than in the other figure. However, note that the *minDCF* result in Table 12.1 for the *EnrollModel + rand* in this data shows a slightly lower performance than *Cosine* result, but in the correspondent DET curve, we observe that the overall performance including EER point are also better than *Cosine* baseline.



(a) Female DET curves

(b) Male DET curves

Figure 12.3: (a) DET curves for female results of the three back-ends. (b) DET curves for male results of the three back-ends.

12.4.3 Analysis of the Detection Cost Function Evolution during the Enrollment Phase

Finally, we have conducted an analysis of this new approach where we have made a brief study of the results obtained individually for each phrase. In Figures 12.4 and 12.5, we represent the evolution of the DCF metric for two different phrases during the training of the enrollment phase by conducting a complete test evaluation at each training stage. Note that for each point in the curve, the full trial list is evaluated and DCF computed for the selected phrase. In these representations, we observe two different behaviours which demonstrate that even though the global performance is better with the proposed

approach, there is still room for improvement. We can find some phrases that the system works correctly with while some others do not behave as good as expected. For example, Figure 12.4 shows one of the phrases which has a great performance and where we can see that the results with the proposed method improves considerably the final DCF result. While in Figure 12.5, we observe that the use of the same training configuration for all the phrases may not be the best option. In this case, with less iterations we can find a better result, but the system does not converge to a better solution compared to the baseline.

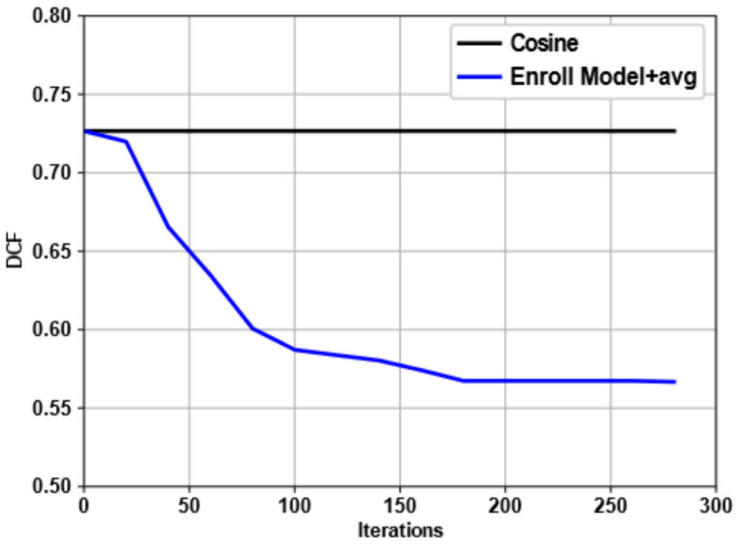


Figure 12.4: DCF evolution in one of the phrases (Phrase 046) from the evaluation data which individually has a great performance during the training of the enrollment model.

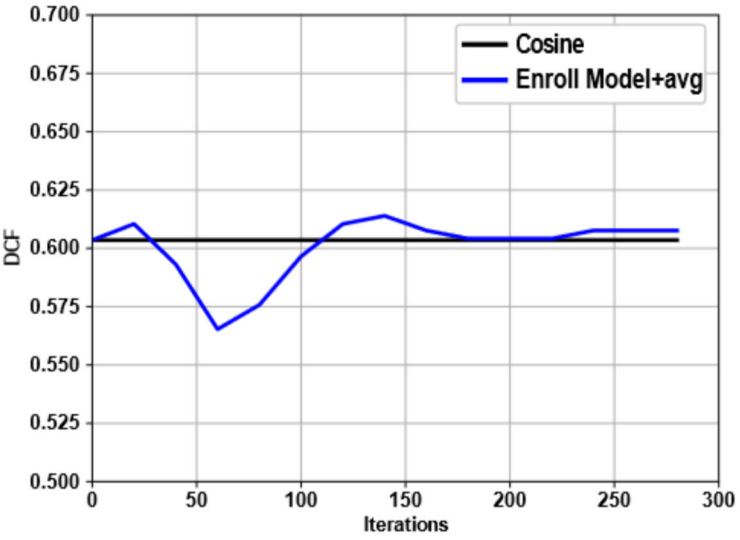


Figure 12.5: DCF evolution in one of the phrases (Phrase 054) from the evaluation data which has one of the worst performance during the training of the enrollment model.

12.5 Conclusions

Along this chapter, we have presented a novel approach to perform the verification process. This approach consists of the use of the embedding dictionary stored during the training phase in the matrix of the last DNN layer as negative examples to train an enrollment model for each speaker. With this system, we mimic the test process where enrollment utterances are compared with test utterances to determine whether each pair of utterances is a target or non-target trial. The proposal has been able to improve the system performance, although, in the analysis part, we have checked that some limitations still exist. The results confirm that this technique is an interesting line of research, so different alternatives to initialize the weight vector combined with different Bayesian estimation approaches could be applied to improve this new approach.

13

Multimodal Diarization Systems by Training Face Enrollment Models as Identity Representations

13.1 Motivation

Multimodal biometric verification field consists of the identification of persons by means of more than one biometric characteristics, as the use of two modalities makes the process more robust to potential problems. Typically, face and voice characteristics have been two of the preferred biometric data due to the ease of obtaining audiovisual resources to carry out the systems that perform this process. When this identification process is applied throughout a video file, and this information is kept over time, this kind of task is also known as multimodal diarization combined with identity assignment. In recent years, this field has been widely investigated due to its great interest, motivated by the fact that human perception uses not only acoustic information but also visual information to reduce speech uncertainty. Moreover, this task has been rarely addressed for uncontrolled data due to the lack of this type of datasets. However, several challenges focused on this topic have recently been developed [304–306], and also a large amount of multimedia and broadcast data is currently being produced such as news, talk shows, debates or series. Therefore, to develop a multimodal biometric system, different tools are required to process this data, detect the presence of people and address the identification of who is appearing and speaking. The need to find new efficient tools to process all the available audiovisual content has led to a wide variety of systems based on artificial intelligence algorithms such as Deep Neural Networks (DNNs).

To perform multimodal diarization, many studies focus on the simplest way which is based on independent systems for speaker and face diarization [306, 307]. Speaker diarization is a widespread task [308, 309] due to its usefulness as pre-processing for other speaker tasks. At the same time, it is still a challenging task because there is no prior information about the number and the identity of speakers in the audio files, and the domain mismatch between different scenarios can produce some difficulties. On the other hand, face diarization has been widely used as a video indexing tool, and the previous step for face verification [310, 311]. However, in real-world scenarios, face images can often appear with large variations, so this kind of system has also found some problems in unconstrained videos. For these reasons, a straightforward score-level fusion is usually employed to join the information from both types of systems.

The IberSPEECH-RTVE 2020 Challenges described in Chapter 3 aims to benchmark and further analyze this different kind of diarization systems. Therefore, two types of diarization evaluations are included in this challenge, Speaker Diarization and Identity Assignment (SDIA) [312], and Multimodal Diarization (MD) [240]. The former is the most extended kind of diarization combined with the speaker assignment. While the latter combines the previous one with face diarization and face identity assignment, which is obtaining more relevance in recent times. In this chapter, we have focused on this second challenge, and especially the characteristics of the face subsystem have been remarked.

This chapter presents the system submitted to the IberSPEECH-RTVE 2020 Challenge in the MD task. This challenge focuses on the segmentation of broadcast audiovisual documents and the assignment to segments of an identity from a closed set of different faces

and speakers. The face and speaker identities from this closed set are known as enrollment or target identities. For the challenge, we have processed audio and video tracks independently in order to separately improve their performance. However, the pipeline is very similar in both cases, where the differences are the exact approach used in each part of the process. Therefore, initially, both audio and video files are processed. After that, an embedding extractor is used to obtain the representations and, finally, clustering and assignment process is applied. The assignment process can be seen as a binary task that consists of comparing each face or speaker present in the audiovisual file with all the enrollment identities and determining whether it belongs to one of them or not. A simple approach employed is a cosine similarity by averaging the representations of all the enrollment files for each identity to obtain the verification scores and decide the identity. Nevertheless, these representations are extracted from DNN systems which are not trained with this objective, so instead of using only cosine similarity, complex back-ends [75, 139] are often applied to improve the discriminative ability of these representations. The drawbacks of this kind of back-end are that it involves a more complex training process and, therefore, a high computational cost. Thus, to carry out the assignment process in the face subsystem of this work, a new approach developed in Chapter 12 [10] has been applied to model the enrollment identities. This new approach was shown to be a promising technique to characterize each enrollment identity with a single learnable vector for the speaker verification task. For this reason, this technique has been applied in face verification in this chapter. To train this back-end, approximated Detection Cost Function (aDCF) loss introduced in Chapter 11 [8, 9] has been used as the objective loss. Apart from the analysis of the use of this new approach for this task, in this chapter, different parameter settings of this loss have been explored and their effect on the errors produced by the system has been studied.

The remainder of this chapter is laid out as follows. In Section 13.2, we describe the new approach based on training face enrollment models by network optimization and the loss function used as objective for the training. Section 13.3 details the face diarization subsystem. The speaker diarization employed is explained in Section 13.4. Finally, Section 13.5 presents and discusses results, and Section 13.6 concludes the chapter.

13.2 Face Enrollment Models

In verification tasks, a back-end is traditionally applied to compare enrollment and test embeddings and obtain the final verification scores to assign the correspondent identity. A widely used approach is the cosine similarity, where if an enrollment identity has more than one enrollment embedding, these embeddings are averaged to be compared with the test embedding. However, we have shown in Chapter 12 for the speaker verification task that a better solution to do this process consists of training an enrollment model for each enrollment identity. Therefore, in this chapter, we have applied this approach for the face verification task where we have trained a model for each of the face enrollment identities.

The loss function optimized and the process to carry out the training of these models with this loss function are explained in detail below.

Motivated by the demonstrated effectiveness in training DNNs using aDCF loss, in Chapter 12, we developed a straightforward and powerful back-end approach based on a network optimization with this loss function. The use of this differentiable version of the DCF metric proposed in Chapter 11 as objective loss function allows training DNN systems aimed at minimizing one of the main metrics employed in verification tasks. In addition, this function is based on the measurement of the two types of decision errors produced by verification systems using a threshold. These two errors are known as false alarms (FA) and misses (FR) which are also part of the Diarization Error Rate (DER) used to evaluate diarization systems as we detailed in Chapter 3. The former errors occur when an identity is incorrectly assigned, while the latter refer to a correct identity not being detected by the system. Therefore, we seek to minimize aDCF loss, which is composed of a weighted sum of the average number of times false alarms (P_{fa}) and misses (P_{miss}) occur.

Using this loss function as objective, this approach tries to mimic the target/non-target process performed in verification tasks. In addition, this back-end takes advantage of the data learned during a previous training step of a general embedding network. Thus, this approach avoids the need for a careful selection of input data to train the models as required by other complex back-ends such as Triplet Neural Network with Triplet loss [75] or Triplet Neural Network combined with aAUC loss introduced in Chapter 10 [3, 7].

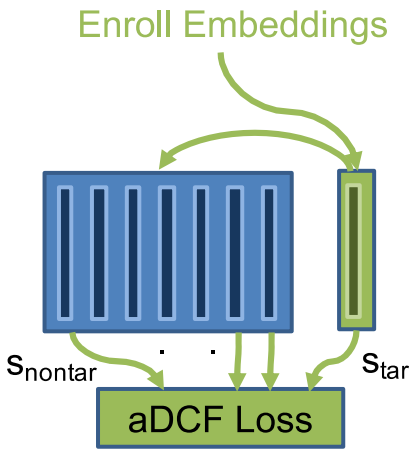


Figure 13.1: Training face enrollment models using target and non-target embeddings for each enroll or target identity.

Figure 13.1 shows the process to carry out this training where a learnable vector is obtained to represent each enrollment identity. This process is based on comparing the positive or target examples with enrollment model (s_{tar}), and also with the negative or non-target examples (s_{nontar}) using the aDCF loss function as training objective loss. As we did in Chapter 12, the original expressions (11.5) and (11.6) for this case are interpreted as,

$$\hat{P}_{fa}(\theta, \Omega) = \frac{\sum_j \sigma_\alpha(s_\theta(e_j^{non}) - \Omega)}{N_{non}}, \tag{13.1}$$

$$\hat{P}_{miss}(\theta, \Omega) = \frac{\sum_i \sigma_\alpha(\Omega - s_\theta(e_i^{tar}))}{N_{tar}}. \tag{13.2}$$

where N_{tar} is the number of target identities, N_{non} is the number of non-target identities, the score $s_\theta(e_j^{non})$ is the non-target score where e_j^{non} is the embedding of each non-target identity with $j \in \{1, ..., m_n\}$ where m_n is each embedding, and the score $s_\theta(e_i^{tar})$ is the target score where e_i^{tar} is each enrollment embedding with $i \in \{1, ..., m_t\}$ and m_t is the number of target samples. To optimize aDCF loss, the scores used are obtained with cosine similarity as,

$$s_\theta(e_i) = \frac{e_i \cdot w_e^T}{\|e_i\| \cdot \|w_e^T\|}, \tag{13.3}$$

where $\|e_i\|$ is the normalized input to the enrollment model, and $\|w_e^T\|$ is the normalized layer parameters of the embedding obtained from the enrollment utterance. Therefore, to train the enrollment models, these expressions are combined to define the approximated loss function employed as in (11.8).

The philosophy of the approach followed in this chapter has been the same as the original approach used for speaker verification, but there are several differences. First, in Chapter 12, the final verification scores were obtained directly in the last step of the training process by comparing all target and non-target samples with the trained vector. Whereas in this chapter, the learnable vector is stored as an enrollment model to be used in the final step of the face subsystem to assign the identity to each segment. On the other hand, the non-target examples employed in this system are directly the embeddings extracted from the pretrained model. These non-target examples belong to identities different from the enrollment identities, so we have used them to train the enrollment models instead of using the weight matrix of the last layer of the trained neural network to obtain them as we did in Chapter 12. Hence, in this chapter, the process for training the enrollment model for each identity is based on the following steps:

1. The target and non-target embeddings extracted from the pretrained model are employed as positive and negative examples.
2. Each enrollment model is trained using aDCF loss with all non-target embeddings and only the target embeddings of the corresponding enrollment identity.
3. The trained models are stored to use them in the assignment process.

13.3 Face Subsystem

In this section, we present the different blocks of the face system, including video processing, embedding extraction, training face enrollment models, clustering, tracking, and identity assignment scoring. The block diagram of the face system is depicted in Figure 13.2.

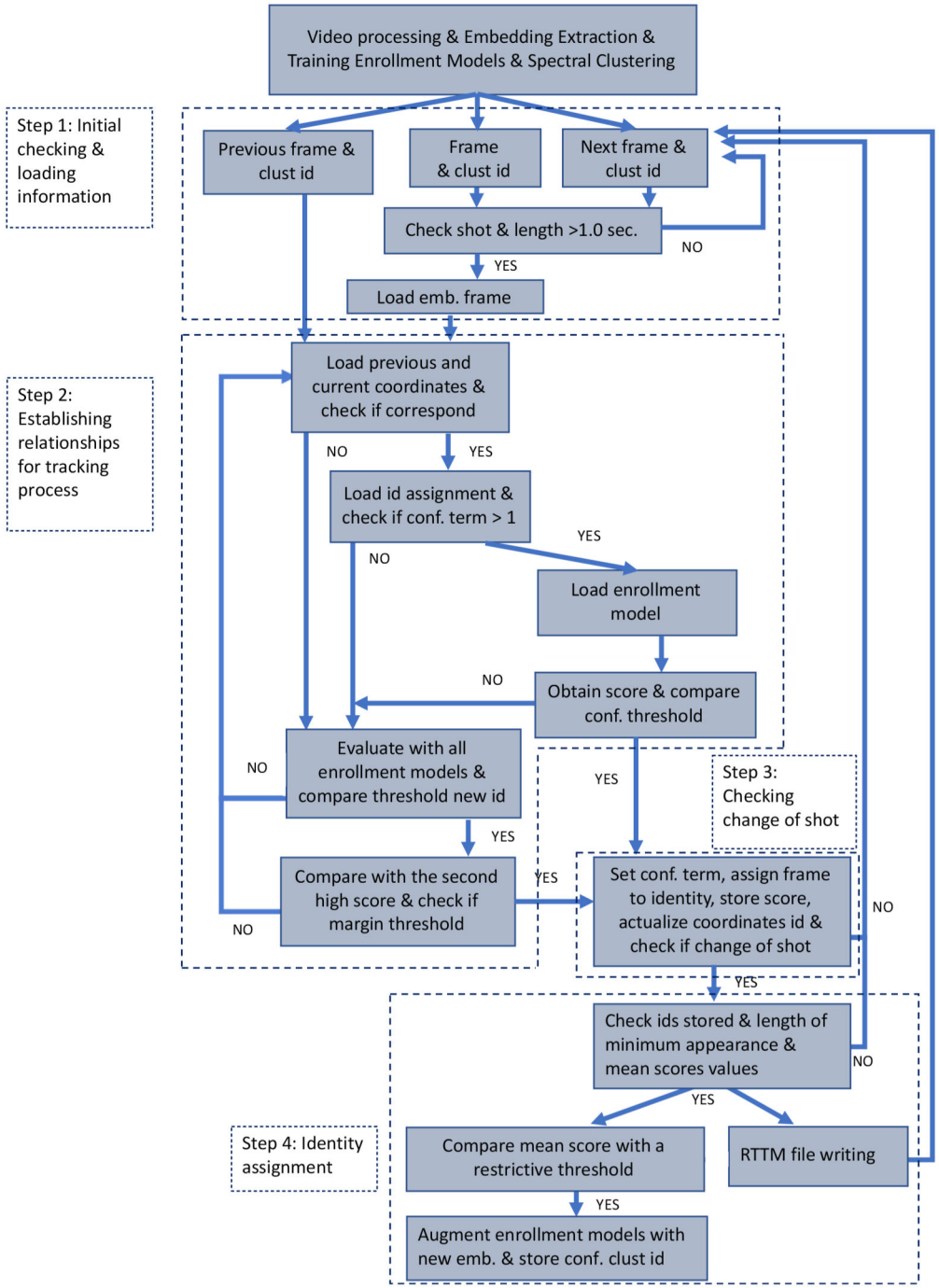


Figure 13.2: Block diagram of face system where the steps of tracking and identity assignment are remarked with dashed lines.

13.3.1 Video Processing

The video processing step used to develop this face subsystem consists of three blocks: frame extraction, face detection and change shot detection. In the following, we will detail all of them.

Frame Extraction

As the first step, the video is processed to extract five frames per second using $ffmpeg$ tool [1]. We decided to use five frames per second since this number of frames allows us to have a high precision to determine the limits of the characters appearance. Therefore, frames are extracted using a constant rate where one frame is obtained every 200 msec.

Face Detection

Another fundamental step in this process is the face detection because failures in this step could be crucial for the correct development in other parts of the face diarization system. In our system, as we also used in Chapter 4, the face detector employed is a system of alignment and detection based on a DNN which is called Multi-task Cascaded Convolutional Networks (MTCCN) [241]. In this part, we employ this implemented system as it is a proven and effective method for face detection. Furthermore, using this detector, we can store the bounding boxes created by the algorithm that correspond to the coordinates where a face is detected. This information is then employed in the tracking and identity assignment process.

Change Shot Detection

The videos employed in this challenge are obtained from television programs, so these programs are usually composed by a huge variability in the characteristics of the content and by constant changes of shots and scenes. Thus, to aid the tracking and clustering step, a change scene detection tool [2] is employed, as this tool effectively detects these changes using threshold-based detection mode. This detector finds areas where the difference between two subsequent frames exceeds a threshold value.

13.3.2 Embedding Extraction

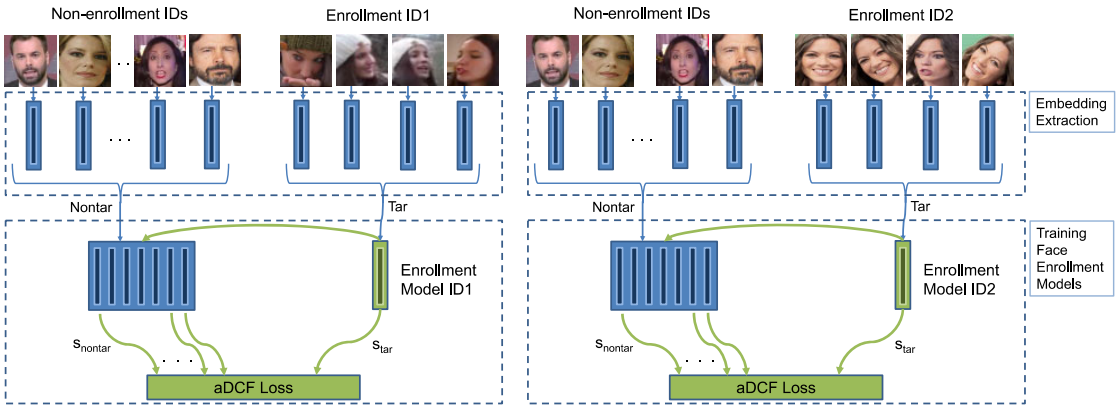
Once the video processing step is done, we process the face images using the bounding boxes, apply mean and variance normalization. Then, as the images have been processed with the information given by the face detector, the resulting images have centered faces.

[1] `https://www.ffmpeg.org/`
[2] `https://www.pyscenedetect.readthedocs.io/en/latest/`

Therefore, a center crop can be applied to resize the images to 160×160 pixels. After that, the processed images are passed through a trained model to obtain embedding representations. The indicated center crop is necessary since the model employed has been trained using images with this format. In this system, as a face extractor, we have employed the pretrained Inception ResNet [75] described in Chapter 4. This network was trained for a classification task on the CASIA-WebFace dataset [231], but the embeddings extracted from it have been proved previously in a verification task to check their discriminative ability with state-of-the-art results on Labeled Faces-in-the-Wild (LFW) [229, 230]. For this reason, we decide to use these embeddings of 128 dimensions to extract the representations of the enrollment and test files of this challenge.

13.3.3 Training Face Enrollment Models

As we explained in Section 13.2, we have applied the approach based on enrollment models proposed in Chapter 12 as a back-end for the face verification task where we have trained a model for each one of the 161 enrollment identities. To train these models, we have used the embeddings of the enrollment images, and the video files of the development and test sets of the IberSPEECH-RTVE 2020 Challenge [240] as positive or target examples. While the enrollment files of the development and test sets of the IberSPEECH-RTVE 2018 Challenge [239] are used as negative or non-target examples. Therefore, once these embeddings are extracted, we train each face enrollment model with them using aDCF loss. Figure 13.3 shows two examples of the steps presented in Section 13.2 of the process of extracting embeddings and training an enrollment model for each identity using the aforementioned data. Moreover, the impact of the amount of non-target data employed to train these models will be discussed in the experimental section.



(a) Model ID1 (b) Model ID2

Figure 13.3: (a) Left: Example of Embedding Extraction and Training Enrollment Model ID1, where the dashed line indicates the two steps of the process. (b) Right: Example of Embedding Extraction and Training Enrollment Model ID2, where the dashed line indicates the two steps of the process.

13.3.4 Clustering

As a source of complementary information, the face embeddings from the test videos are used to perform a spectral clustering technique [313] that attempts to find strongly connected segments. This technique provides an initial cluster assignment to group the frames of the video sequence. In this chapter, we have employed this clustering combined with the use of coordinates to improve the whole tracking process.

13.3.5 Tracking and Identity Assignment Scoring

Once all the above information is obtained, we have developed an algorithm to carry out the tracking and identity assignment process, which is depicted in Figure 13.2 and follows a similar philosophy to the one developed in [314]. In this algorithm, the tracking process has been developed by shot, so a change of shot restarts the tracking. Therefore, while the shot is the same, in Step 1 showed in Figure 13.2, the algorithm checks frame by frame the clustering information and the correspondence between the coordinates of the current frame and the previous frame to establish links to perform the tracking process in Step 2 of Figure 13.2. When a relationship exists between both frames and has a high confidence term, the identity assignment of the previous frame is used to select the enrollment model and obtain the score. This score is compared with a confidence threshold to determine whether the assigned identity is correct or not. However, when there is no relationship between the coordinates of the current and previous frame or the confidence term is low, the frame embedding is compared to all enrollment models to obtain a score and determine whether it is a new identity to assign. Once the identity assignment is performed on the current frame, the score is stored, the coordinates are updated, and the algorithm checks whether the shot changes. These processes are carried out in Step 3 of Figure 13.2.

Tracking is carried out with the above steps, but the identity assignment process performed is only an initial assignment. When a shot change is detected in Step 4 of Figure 13.2, the system checks the identities and scores stored in the shot to remove inconsistent segment assignments. After that, the final segments with their identity assignments are written to the Rich Transcription Time Marked (RTTM) file. In addition, score confidence values are stored when a final identity assignment is made. If these values are greater than a more restrictive threshold which is set with the development set, we augment the enrollment models with the current face embedding. The whole process is repeated with all detected shots.

13.4 Speaker Subsystem

This section describes the speaker subsystem which consists of similar blocks to the face subsystem, such as audio processing, embedding extraction, clustering, and identity as-

signment scoring, but using different approaches in each one. The different parts of just this subsystem have been explained in more detail in [315]. Although as part of the multimodal diarization system, the following sections present the main ideas for creating this subsystem.

13.4.1 Audio Processing

In this subsystem, the audio processing step is also composed of three blocks: a front-end, speech activity detection and speaker change point detection. Next, we will briefly explain each of them.

Front-end & Speech Activity Detection

The first block of this subsystem is a front-end to obtain the Mel-Frequency Cepstral Coefficients (MFCCs) features [96]. For a given audio, a stream of 32 coefficient feature vectors is estimated according to a 25ms window with a 15ms overlap. No derivatives are considered. Simultaneously, Speech Activity Detection (SAD) labels are estimated each 10 ms [316]. Our approach for SAD is based on a deep learning solution that is an evolution derived of our previous experience with SAD systems in different domains [207]. We use a convolutional recurrent neural network (CRNN) consisting of 3 blocks of 2D convolutional followed by 3 BiLSTM layers [317]. Then, the final speech score is obtained through a linear layer. As input features, 64 Mel filter banks and the frame energy are extracted from the raw audio and fed to the neural network. Cepstral Mean and Variance Normalization (CMVN) [318] normalization is applied.

The CRNN is trained on a combination of different broadcast datasets. Specifically, we include data from the Albayzín 2010 dataset (train and eval), Albayzín 2018 dataset (dev2 and eval) and a selection of data from the first MGB Challenge (train, dev.longitudinal and task3 eval). Furthermore, audios are augmented with a variety of noises that can be usually found in broadcast emissions (sitcom noises, crowd and laughter noises, babble, street music and stadium noises).

Speaker Change Point Detection

Feature vectors and SAD labels obtained are fed into the Speaker Change Point Detection (SCPD) block which is dedicated to infer the speaker turn boundaries. The differential form of Bayesian Information Criterion (ΔBIC) [319] has been used. This estimation works in terms of a 6-second sliding window, in which we assume there is, at most, a speaker turn boundary. Each involved speaker in the analysis is modeled by means of a full-covariance Gaussian distribution. Besides, the SAD labels delimit the parts of the audio where the analysis is performed. In the given data, the described configuration provides segments of an approximately 3-second length on average.

13.4.2 Embedding Extraction

Once the audio processing is done, each one of the identified segments will be converted into a compact representation or embedding. For this purpose, we have opted for an evolution of x-vectors [132] considering an extended version [320] of the Time Delay Neural Network (TDNN) architecture [321]. The modification is the inclusion of multi-head self-attention [268] in the pooling layer. This network, trained on VoxCeleb [282] and VoxCeleb2 [137], provides embeddings of dimension 512. These embeddings undergo centering and LDA whitening (reducing dimension to 200), both trained with MGB [322] as well as the Albayzín training subset, and finally length normalization [323]. These embeddings will be referred to as Φ. A similar extraction pipeline working offline is in charge of the enrollment audios. The enrollment embeddings will be named Φ_{enroll}.

13.4.3 Clustering

The obtained embeddings are modeled in a generative manner according to [324], where a tree-based Probabilistic Linear Discriminant Analysis (PLDA) clustering is proposed. This approach exploits the higher acoustic similarity of temporally close embeddings by sequentially assigning these representations to the available clusters at each time. These clusters are managed by the algorithm at the same time. This concept is boosted by [325], which helps to find the best possible sequence of decisions. The considered model has 100-dimension speaker subspace and it is trained with both MGB and Albayzín training subset.

13.4.4 Identity Assignment Scoring

The considered identity assignment block follows a state-of-the-art speaker recognition PLDA backend followed by score normalization and calibration stages. By means of the PLDA model, we estimate the score s_{ij}, which represents how likely the diarization cluster j shares the same speaker identity as the enrollment speaker i. Then, these scores are normalized according to adaptive S-norm [215], using MGB as extra cohort. Finally, normalized scores are calibrated according to a threshold ϵ. Whenever the score s_{ij} overcomes the threshold, we consider the cluster j contains audio from the enrolled person i, being different otherwise. Threshold ϵ is adjusted by Assignment Error Rate (AER) minimization according to a calibration subset Φ_{calib} and the enrollment embeddings Φ_{enroll} as follows:

$$\epsilon = \arg\min_{\epsilon}(\text{AER}(\Phi_{\text{calib}}, \Phi_{\text{enroll}}, \epsilon)) \tag{13.4}$$

where Φ_{calib} and Φ_{enroll} represent the set of embeddings from calibration as well as the enrollment speakers.

Final AER labels are obtained according to these normalized and calibrated scores. The audio from cluster j is assigned to the ith enrolled identity with highest score if s_{ij}

overcomes the calibration threshold. If s_{ij} is below the threshold for any enrolled identity i, the cluster is assigned to the generic unknown identity. Mathematically, the assigned identity (θ_j) for a subset of embeddings j with respect to the enrolled identity i is:

$$\theta_j = \begin{cases} \arg\max_i(s_{ij}|s_{ij} > \epsilon) & \text{if } \exists i|s_{ij} > \epsilon \\ \text{Unknown} & \text{if } \forall i,\, s_{ij} < \epsilon \end{cases} \tag{13.5}$$

where s_{ij} stands for the normalized PLDA log-likelihood ratio score between the embedding j and the enrolled identity i. By means of these decision making, we do not exclude the possibility of assigning multiple diarization clusters to the same identity. This design choice is taken to allow the fix of some diarization errors.

13.5 Experiments and Results

In this chapter, several experiments have been carried out to show the effect of different aspects on the face subsystem and the overall performance of both subsystems. First, we compare the use of a cosine similarity metric directly on the embeddings extracted from the pre-trained model ($AverageEmbedding$) to obtain the closest identity in each instance with the training face enrollment models approach ($EnrollmentModels$) for the identity assignment process. Moreover, in this first set of experiments, the relevance of employing more non-target data to train the enrollment models is also analyzed. After that, we have analyzed the behaviour of the system when different values of the aDCF loss parameters are employed. To conclude, a summary of the best results of the face subsystem in combination with the results of the speaker subsystem is presented.

13.5.1 Analysis of Training Enrollment Models for Face Subsystem

In this section, we have analyzed the performance of the system when enrollment models are trained and used for the identity assignment process or a cosine similarity is directly employed to compare the average of all enrollment embeddings with each frame of the video file. Furthermore, the effect of adding more non-target data to train the enrollment models is also checked.

Table 13.1 shows *DER%* results on the test set for the face subsystem with the different back-end approaches. As we can observe, the training of face enrollment models to characterize each enrollment identity achieves a large improvement over comparing each segment directly against the average of the enrollment embeddings. Note that whether the enrollment models are trained with more non-target examples, the variability that the learnable vectors have to model is higher so these models learn better to represent each identity and the *DER%* result obtained is lower.

Table 13.1: Experimental results on RTVE 2020 Multimodal Diarization test set, showing *DER%*. These results were obtained to compare the back-end approach proposed and the cosine baseline.

Back-end	Non-target Examples	DER%
Average Embedding	–	80.16
Enrollment Models	57	61.79
	3302	**56.86**

13.5.2 Effect of aDCF Parameters γ, β for Training Face Enrollment Models

A second set of experiments has been performed to observe the effect of training with different aDCF loss parameters as was done in Chapter 11 for speaker verification. In Table 13.2 and Figure 13.4, we observe that the system performance improves only adjusting the aDCF parameters without modifying the threshold values or the tracking and identity assignment algorithm for the reference number of training epochs. As reference number of epochs, we have considered 800 epochs since it is the number initially used in the previous set of experiments. In view of this result, we have explored in depth the behaviour of the training enrollment models for the different configurations of parameter and number of epochs. As a result of this sweep, we have obtained that the original parameter configuration, $\gamma = 0.75$ and $\beta = 0.25$, has still room for improvement, and training for 1200 epochs achieves the best result without modifying any other parameter. Moreover, note that giving a higher cost relevance (β) of the probability of misses during training with aDCF loss, regardless of the number of epochs, the results are worse in all situations.

Table 13.2: Experimental results on RTVE 2020 Multimodal Diarization test set, showing *DER%*. These results were obtained by sweeping of the parameters of aDCF loss function and by different number of training epochs.

γ	β	600 epochs	800 epochs	1000 epochs	1200 epochs	1400 epochs
0.75	0.25	62.56	56.86	**55.32**	**54.07**	**55.91**
0.50	0.50	**55.92**	**55.89**	56.02	56.47	57.16
0.25	0.75	58.30	59.29	59.58	60.01	60.02

In addition to the above results, we have analyzed the behaviour of the three types of errors that compose the *DER%* metric when the different possible system configurations are used for the same number of training epochs. As we can see in Table 13.3 and Figure 13.5, giving a higher cost relevance to the probability of false alarms (γ) in the aDCF loss results in a lower number of false alarms in the decomposition of the *DER%*. While the number of misses is higher than in the other two configurations. On the other hand, the same trend can be seen in reverse when the relevance term (β) is higher for the probability of misses.

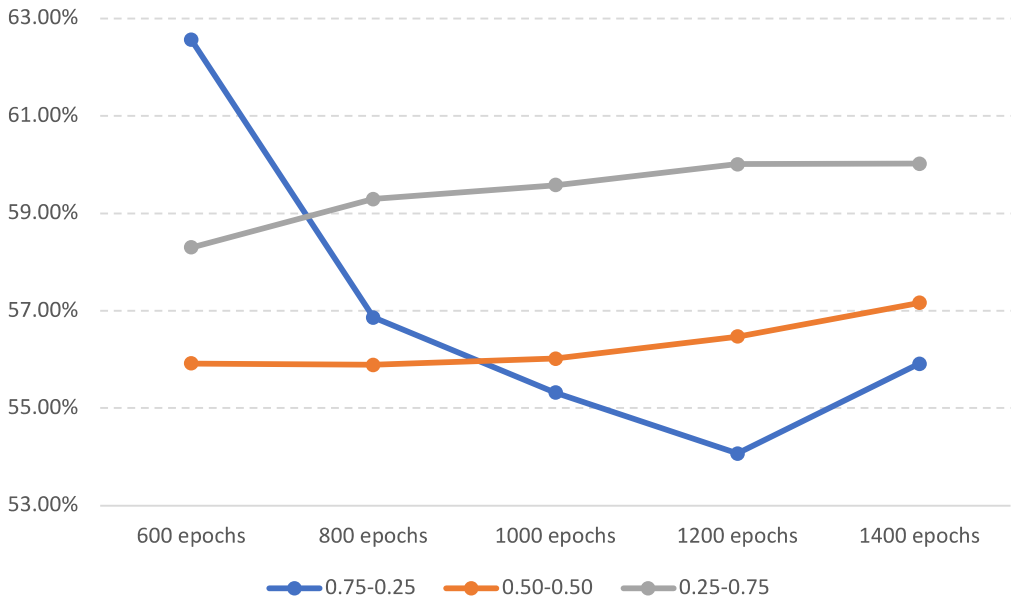


Figure 13.4: Evolution of the *DER%* results in function of the different parameter configurations.

Table 13.3: Experimental results on RTVE 2020 Multimodal Diarization test set, showing *DER%* and Decomposition of the *DER%* results in Miss (*MISS%*), False Alarm (*FA%*) and Identity (*ID%*) Errors. These results were obtained by sweeping of the parameters of aDCF loss function.

γ	β	**MISS%**	**FA%**	**ID%**	**DER%**
0.75	0.25	32.60	**12.80**	**8.70**	**54.07**
0.50	0.50	30.70	16.00	9.80	56.47
0.25	0.75	**28.30**	19.80	12.00	60.01

13.5.3 Summary of Face and Speaker Results

In this section, we have collected the best result for the face and speaker subsystems, and also, we have divided into development and test set the results to better observe the difference behaviour of both subsystems. Moreover, our reference results obtained for the challenge [11] have been included to better reflect the improvement achieved. Thus, the results of the other two participating groups [326, 327] that are publicly available [3] have also been introduced as reference of the difficulty of this multimodal diarization challenge.

Table 13.4 shows the *DER%* results obtained in the development and test set for the face and speaker modalities. In addition to the separate results, we show the average result of the face and speaker diarization errors (*FACE + SPEAKER*). These results indicate a great mismatch between development and test results. We have analyzed what type of video files composed both subsets and the length of these files, and we have found that the

[3] `http://catedrartve.unizar.es/albayzin2020results.html`

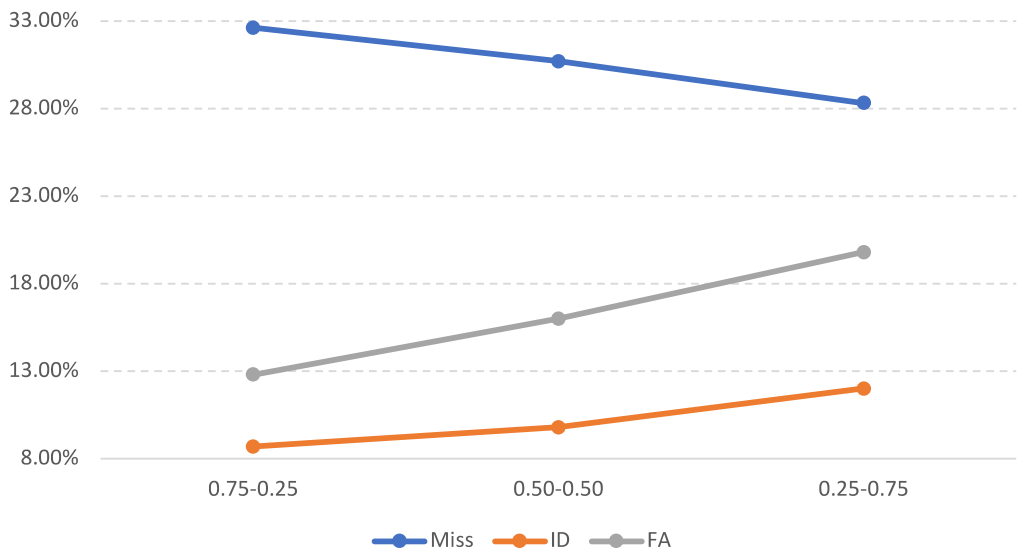


Figure 13.5: Evolution of the different types of errors (*MISS%*, *FA%*, *ID%*) for each different γ, β parameter configurations.

Table 13.4: Experimental results on RTVE 2020 Multimodal Diarization development and test sets, showing *DER%*. These *DER%* values were the result of the improvements introduced in this work, and the reference results for both modalities are also presented.

Subset	**Modality**	**DER%**	**DER% ours [11]**	**DER% [326]**	**DER% [327]**
DEV	FACE	51.26	51.66	–	–
	SPEAKER	37.45	47.90	–	–
	FACE+SPEAKER	44.36	49.78	–	–
TEST	FACE	54.07	61.79	44.55	67.31
	SPEAKER	60.34	72.63	61.61	131.59
	FACE+SPEAKER	57.20	67.21	53.08	99.45

development files are shorter and more similar than test files. Thus, we can see that the face and speaker subsystems perform better in the development files which are shorter videos, so the tracking process is easier to follow. Nevertheless, in the face subsystem, we observe that this difference is smaller than in the speaker subsystem.

To better analyze these results, Table 13.5 presents a decomposition of the *DER* metric into the three terms of error. Focusing on the face modality errors, in the case of the development subset, we observe that the main cause of error is the probability of misses, which indicates that a large number of segments of the target identities have not been detected. Therefore, this effect can be motivated by the fact of using a threshold value that is too restrictive. While in the test subset, misses term decreases and especially relevant is the increase in false alarm errors as this fact illustrates the problems in discarding segments of non-target faces when the number of enrollment identities is large. On the other hand, the distribution of errors produced in the speaker subsystem is quite different, as false alarms are much larger than misses in test subset of data. Note that it is also related to the chosen threshold. However, in this case, the threshold is lower, so the

target segments are mostly detected, but as a result, a high number of enroll identities are assigned to segments of unknown identity.

Table 13.5: Decomposition of the *DER%* results in Miss (*MISS%*), False Alarm (*FA%*) and Identity (*ID%*) Errors for the development and test sets in both modalities.

Modality	**Subset**	**MISS%**	**FA%**	**ID%**
FACE	DEV	41.70	5.10	4.50
	TEST	32.60	12.80	8.70
SPEAKER	DEV	26.10	9.60	1.75
	TEST	8.70	39.00	12.64

13.6 Conclusions

This chapter presents the system submitted to the IberSPEECH-RTVE 2020 Multimodal Diarization Challenge. In this chapter, we have developed two monomodal subsystems to address separately face and speaker diarization. Each system is based on state-of-the-art DNN approaches. In addition, we have introduced the new back-end approach proposed in Chapter 12 for the face subsystem. This approach consists of training a learnable vector with aDCF loss to represent each face enrollment identity. Using these enrollment models for the identity assignment process instead of just the cosine similarity, the results have achieved a relevant improvement over the average embedding directly and the application of cosine similarity. We have demonstrated that there is still room for improvement in each of the systems because the results obtained are too high in both subsets and in both systems. Moreover, future work can be done on the fusion of both systems, which could improve the final results, especially by disambiguating the identification process. The high *DER* values for misses and false alarms in the face and speaker subsystem, respectively, should be addressed by that fusion.

14

Log-Likelihood Ratio Cost as Training Objective Loss

14.1 Motivation

Along this third part of the thesis, we have focused on the analysis and implementation of suitable loss functions to train Deep Neural Network (DNN) architectures for verification systems since these systems are usually trained without considering their main goal. As we explained in Chapter 9, the recent research efforts have focused on the design of new loss functions to train these systems in two lines of study, on one side, the redesign of the identification loss function, and on the other side, the verification loss functions. However, these new loss functions have not been oriented to the goal of these systems which consists of providing a reliable binary decision of whether two samples belong to the same identity or not. In order to make reliable decisions, verification systems must

obtain proper log-likelihood ratios (LLRs) and compare them against a convenient threshold. The development of a method able to convert the outputs of the system into proper LLRs is usually known as calibration [226]. This process takes into account parameters such as the prior probability of a sample to belong to the legitimate identity and the costs of making wrong decisions to adjust a few parameters to meet the requirements of the application. In previous chapters, we presented different loss functions to address the issue of training the system using one of the final evaluation metrics employed to check whether this step has been correctly done. The first one implemented in Chapter 10 [3,7] is aAUC loss, which is an approximation of Area Under the ROC Curve combined with the triplet training philosophy. However, the use of a triplet strategy involves high computational cost and slows down the training process. While the second approach developed in Chapter 11 [8, 9] is aDCF loss which is an approximation inspired by Detection Cost Function (*DCF*). This function is based on the measure of the decision errors in verification systems. Unlike aAUC approach, aDCF loss is designed to address the minimization of the decision errors as a function of a single threshold following the philosophy of the existing multi-class loss function, so it is more efficient than the triplet training strategy. Despite its efficiency, this loss function has a main drawback since it is an application-dependent metric and needs some prior and cost parameters assumptions to be used as objective loss function. Thus, using aDCF loss, the system is trained to meet the requirements of a specific application. Furthermore, we had to make an approximation of the real *DCF* metric to train a neural network.

Therefore, the aim of this chapter is to propose a new objective loss function for training DNNs as an alternative to our previous aDCF loss to substitute the classical identification losses. This function is based on another verification metric, Log-Likelihood Ratio Cost (CLLR) [216, 226]. CLLR is an application-independent evaluation metric which measures the overall quality of the scores for making soft decisions using the expected costs where no assumptions of prior or cost parameters are needed. Moreover, CLLR has a differentiable expression, so we do not have to approximate it as in the case of *DCF*. Therefore, training with CLLR as objective loss allows the end-to-end system to learn how to minimize the expected costs by obtaining good scores.

14.2 Log-Likelihood Ratio Cost Loss

Motivated by the fact that aDCF is a good choice to train the verification systems, but it has the drawback of the fixed operating point, this chapter presents a loss function for verification systems based on Log-Likelihood Ratio Cost (CLLR) [226]. This loss function is a generalization of the previous aDCF since CLLR is formulated as an integral over all possible operating points of *DCF*. Besides, it is also related to another widely employed graph representation which is known as the Detection Error Trade-off (DET). DET curve is a representation of what happens whether the decision threshold is swept across its whole range, so CLLR can be viewed as a summary of the accuracy obtained over the

whole DET curve. The integral of CLLR is defined as,

$$CLLR = \int_{\Omega} DCF(\Omega)\, d\Omega, \tag{14.1}$$

where Ω is the overall spectrum of operating points to integrate over them.

This integral expression is not differentiable, so it can not be employed to train a DNN. However, in [226], an analytical closed-form expression was presented to solve this integral, so we do not need to make any approximation as we did in Chapter 11 with *DCF*. Using this expression, we can introduce it directly in the DNN as objective loss to optimize. CLLR defined with this differentiable solution measures a sum of the expected log costs of target examples ($Ctar$) and the expected log costs of non-target examples ($Cnon$). $Ctar$ is defined by the sum of the cost for each target example where whether the system assigns it correctly a high score for the target hypothesis, the cost will be low. While $Cnon$ is determined by the sum of the cost for each non-target example where this cost will be low whether the assigned score is low. The expected log costs of target and non-target can be written as,

$$Ctar(\theta) = \sum_{y_i \in y_{tar}} \log(1 + exp(-s_\theta(x_i, y_i))), \tag{14.2}$$

$$Cnon(\theta) = \sum_{y_i \in y_{non}} \log(1 + exp(s_\theta(x_i, y_i))), \tag{14.3}$$

where $s_\theta(x_i, y_i)$ is the score obtained from the last layer of the neural network. Furthermore, in the implementation used in our system, we have introduced a η parameter to divide the scores which is known as temperature scaling parameter [328].

Combining the previous expressions, we propose to minimize CLLR loss as objective loss defined as,

$$CLLR(\theta) = \frac{1}{2\log 2}\left(\frac{Ctar(\theta)}{N_{tar}} + \frac{Cnon(\theta)}{N_{non}}\right), \tag{14.4}$$

where N_{tar} is the number of target identities, and N_{non} are the non-target identities. Using this expression, we can apply it directly to our system without any other assumption. As in Chapter 11, the optimization is performed iteratively by computing CLLR for each minibatch and updating the model. Each minibacth is interpreted as a verification evaluation where the outputs of the last layer are interpreted as scores. Therefore, using the class labels, we define the target and non-target scores. In Figure 14.1, a graphical example of this process is depicted where we observe that target scores with lower values produce high-cost value in (14.2) term, and non-target scores with higher values involves a high-cost value in (14.3). Therefore, with the training process and the gradient backpropagation, the system learns to minimize these costs.

In addition, note that CLLR metric can be also interpreted as a measure of loss of information [216] since a CLLR value close to 0 represents that good scores have been

obtained. Therefore, this means that these scores store a large amount of information which allows reducing the uncertainty about the identity hypothesis to accept or reject one person. While whether a CLLR value higher is obtained, there is a great loss of information, so the error rate is similar to the reference detector.

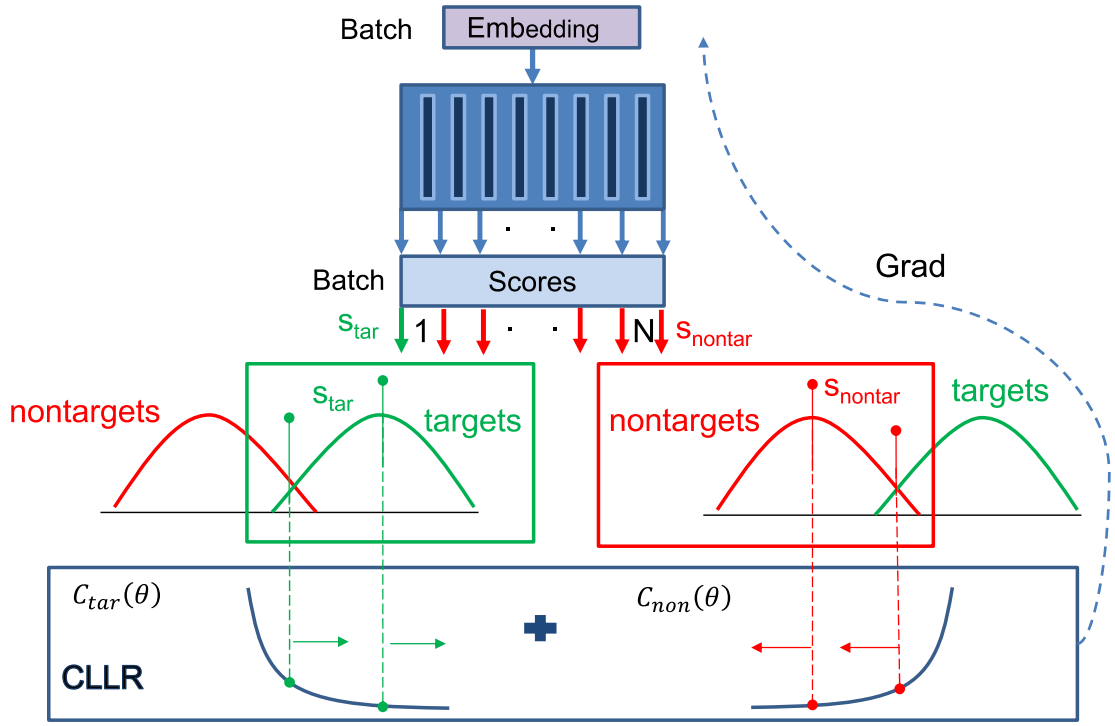


Figure 14.1: Graphical example of the main idea behind how DNNs are optimized by minibatch using CLLR loss. In this example, the target and non-target scores distributions are shown and the contribution of two target and non-target scores to each cost of CLLR loss can be observed.

14.3 Residual Network Architecture combined with Multi-head Self-Attention and Memory Layers using CLLR Loss

In this section, we briefly present the system architecture used to evaluate the effectiveness of the proposed CLLR loss in this chapter for text-dependent speaker verification which is depicted in Figure 14.2. This architecture is the same introduced in Chapter 7 where the backbone is composed of two Residual Network blocks (ResBlock) with three layers each block. Furthermore, this architecture needs positional information [268] for the Multi-head Self-Attention (MSA) layers to provide good performance. Instead of using temporal positional information as many language modelling applications, we use the output of a phonetic classifier bottleneck [269, 275]. We concatenate this information before each Residual block.

For the pooling part, two MSA layers with two memory layers are alternated. Since we use a concatenation of MSA layers, it is equivalent to the encoder part of a transformer, which can be seen analogously as an alignment method that allows assigning embeddings to several categories. This approach has been found useful for text-dependent tasks [2, 279]. In addition, with the integration of the phoneme embeddings in the backbone part, the performance of the attention mechanism improves since the phoneme embeddings help to guide to the attention mask.

The main difference of the architecture used in this chapter with respect to the architecture of Chapter 7 is that we have modified the part of the objective loss function to introduce the proposed new CLLR loss and also, the previous developed aDCF loss and A-Softmax loss.

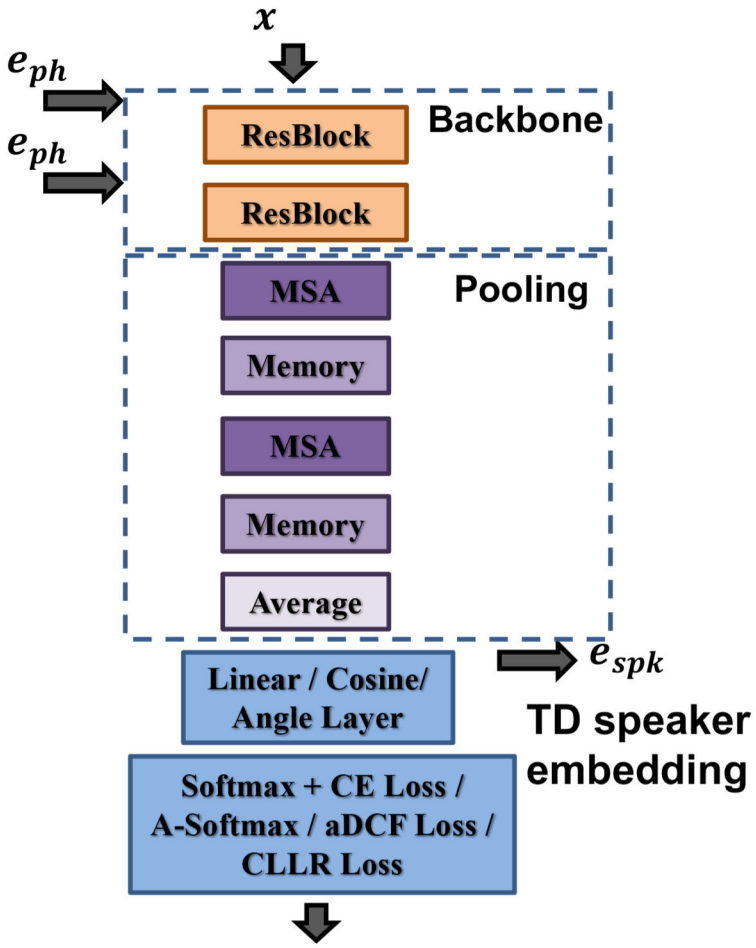


Figure 14.2: Architecture for ResBlock, MSA and Memory layers network, composed of a backbone, a pooling and a embedding extraction.

14.4 Experiments and Results

14.4.1 Experimental Setup

To develop our experiments with the previous architecture, RSR2015-Part II has been employed. 20 dimensions Mel-Frequency Cepstral Coefficients (MFCCs) stacked with their first and second derivates are used as input to train the architecture. Moreover, we have extracted phonetic embeddings of 256 dimensions from a phonetic classifier network which are employed as positional information. Unlike the previous chapters in this third part of this thesis, for these experiments, we have trained a single DNN architecture with all phrases of the database instead of one model for each phrase. Motivated by the fact that we need to use more data to successfully train an architecture with a self-attention mechanism. To train this architecture for the different experiments, we have used only the bkg partition. The eval data is used for enrollment and test evaluation.

In this chapter, a set of experiments has been carried out to evaluate the new loss function proposed. We compare the system trained using some of the state-of-the-art loss functions with the proposed CLLR loss. The architecture of the different systems are described in Tables A.20, A.21 and A.22 in Appendix A. Once these systems are trained, we have evaluated them using only a cosine similarity without any score normalization technique or calibration step to show the effectiveness of the scores obtained by training with the different loss functions. Finally, a score normalization is applied to confirm that the results have the same trend even then.

14.4.2 Results with RSR2015-Part II

Table 14.1 presents Equal Error Rate (*EER%*), minimum Detection Cost Function 08 (*minDCF08*), minimum Detection Cost Function 10 (*minDCF10*), and minimum Log-Likelihood Ratio Cost (*minCLLR*) for the mentioned loss functions. Observing these results, we can conclude that the proposed CLLR loss achieves the best results within the four metrics. Thus, we have checked that using this loss function to train the speaker verification system, we have improved the quality of the scores for all the operating points since apart from the improvement of *minCLLR*, this system achieves the best results in the operating points where EER and *DCF* are evaluated. Additionally to the previous metrics, in the last two rows of the table, we present the relative improvement achieved comparing the system with CLLR with CE+RL and aDCF systems. These comparatives allow us to remark the fact that training with a loss function oriented to the goal task improves significantly the whole system performance.

Moreover, note that these performances have been obtained training a single model with all the phrases and without applying score normalization. Therefore, it shows that speaker verification systems trained with one of the final verification metrics produce good scores to achieve promising results without the need for normalization. Nevertheless, we have also included a score normalization (snorm) to confirm that the results

Table 14.1: Experimental results on RSR2015-Part II [124] eval set, showing *EER%*, *minDCF08*, *minDCF10*, and *minCLLR*. These results were obtained by training with bkg subset to compare the approach proposed with different loss functions.

	Female			
Loss Function	**EER%**	**minDCF08**	**minDCF10**	**minCLLR**
CE	5.87	0.286	0.740	0.214
CE+RL	4.64	0.228	0.669	0.171
A-Softmax	4.99	0.251	0.703	0.189
aDCF	4.20	0.201	0.660	0.158
CLLR	**3.64**	**0.170**	**0.532**	**0.139**
CLLR vs CE+RL (%)	**21.55**	**25.43**	**20.47**	**18.71**
CLLR vs aDCF (%)	**13.33**	**15.42**	**19.39**	**12.02**

(a) Female results

	Male			
Loss Function	**EER%**	**minDCF08**	**minDCF10**	**minCLLR**
CE	6.37	0.304	0.803	0.236
CE+RL	4.92	0.244	0.712	0.184
A-Softmax	6.44	0.309	0.777	0.239
aDCF	4.90	0.232	0.668	0.182
CLLR	**4.07**	**0.200**	**0.588**	**0.157**
CLLR vs CE+RL (%)	**17.27**	**18.03**	**17.41**	**14.67**
CLLR vs aDCF (%)	**16.94**	**13.79**	**11.97**	**13.73**

(b) Male results

	Female+Male			
Loss Function	**EER%**	**minDCF08**	**minDCF10**	**minCLLR**
CE	6.23	0.302	0.784	0.229
CE+RL	4.80	0.237	0.706	0.179
A-Softmax	5.80	0.283	0.746	0.217
aDCF	4.64	0.226	0.677	0.175
CLLR	**3.96**	**0.189**	**0.567**	**0.151**
CLLR vs CE+RL (%)	**17.32**	**20.25**	**19.68**	**15.64**
CLLR vs aDCF (%)	**14.65**	**16.37**	**16.25**	**13.71**

(c) Female+Male results

maintain the same trend than without score normalization and even improve slightly the performance as we can see in Table 14.2.

In addition to the previous tables, Figures 14.3 and 14.4 depicts the DET curves which represent the decision errors sweeping the threshold over all the operating points. These curves show the results for female+male experiments. Note that these representations clearly demonstrate that the DET curve obtained with CLLR system shows better results

Table 14.2: Experimental results on RSR2015-Part II [124] eval set, showing *EER%*, *minDCF08*, *minDCF10*, and *minCLLR*. These results were obtained by training with bkg subset to compare the approach proposed with different loss functions with normalization (snorm).

		Female		
Loss Function	**EER%**	**minDCF08**	**minDCF10**	**minCLLR**
CE	5.64	0.278	0.722	0.207
CE+RL	4.62	0.232	0.708	0.172
A-Softmax	4.67	0.236	0.706	0.178
aDCF	3.99	0.195	0.647	0.153
CLLR	**3.46**	**0.167**	**0.543**	**0.135**
CLLR vs CE+RL (%)	**25.10**	**28.02**	**23.30**	**21.51**
CLLR vs aDCF (%)	**13.28**	**14.74**	**16.07**	**11.76**

(a) Female results

		Male		
Loss Function	**EER%**	**minDCF08**	**minDCF10**	**minCLLR**
CE	6.19	0.304	0.813	0.232
CE+RL	4.93	0.246	0.738	0.186
A-Softmax	6.00	0.292	0.781	0.224
aDCF	4.83	0.241	0.674	0.184
CLLR	**3.99**	**0.194**	**0.572**	**0.153**
CLLR vs CE+RL (%)	**19.07**	**21.14**	**22.49**	**17.74**
CLLR vs aDCF (%)	**17.39**	**19.50**	**15.13**	**16.84**

(b) Male results

		Female+Male		
Loss Function	**EER%**	**minDCF08**	**minDCF10**	**minCLLR**
CE	6.09	0.298	0.783	0.226
CE+RL	4.89	0.243	0.744	0.182
A-Softmax	5.42	0.267	0.753	0.205
aDCF	4.58	0.226	0.681	0.175
CLLR	**3.91**	**0.186**	**0.584**	**0.149**
CLLR vs CE+RL (%)	**20.08**	**23.46**	**21.50**	**18.13**
CLLR vs aDCF (%)	**14.84**	**17.69**	**14.24**	**14.85**

(c) Female+Male results

in all the operating points while the DET of aDCF system only outperforms the DET curve of CE+RL system in some points. This fact can be motivated by the assumption of the selected parameters to train with aDCF the system. Furthermore, A-Softmax loss is a sensitive method to the training parameters to achieve good results.

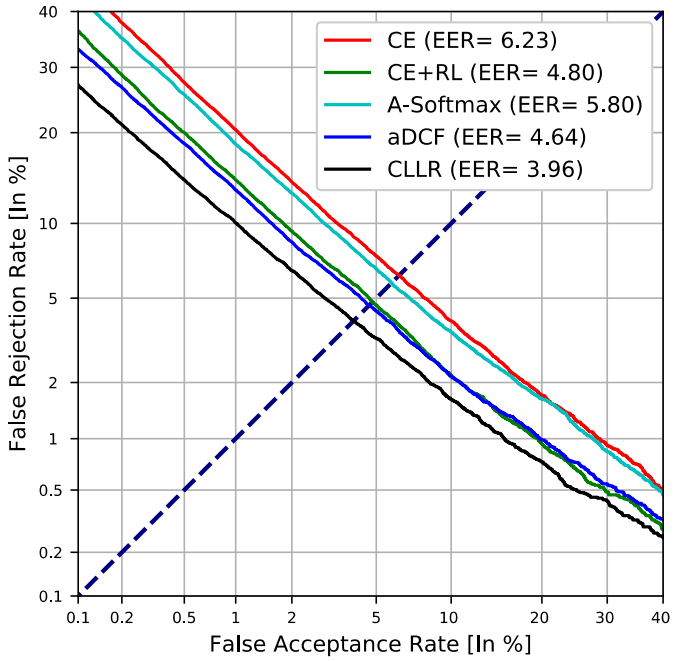


Figure 14.3: DET curves for female+male results using the different loss functions evaluated.

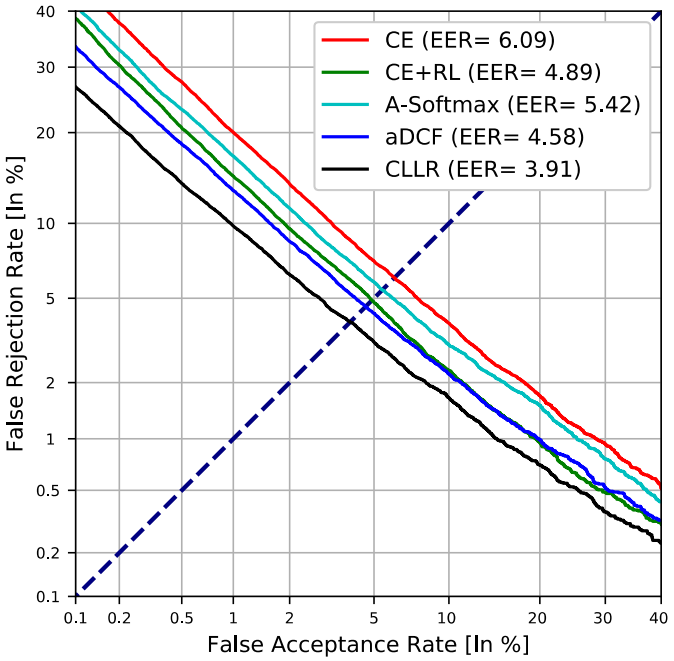


Figure 14.4: DET curves for female+male results using the different loss functions evaluated with normalization (snorm).

14.5 Conclusions

In this chapter, we have presented a new loss function based on optimizing the quality of the scores over all the operating points. This CLLR loss is an alternative to aDCF loss to replace the traditional identification loss functions as CE loss without the need

of making prior or cost assumptions, and also we have not made any approximation of the real metric. Moreover, the use of CLLR loss allows the system to learn how to reduce the expected log costs of target and non-target examples. The evaluation was carried out in the RSR2015-Part II text-dependent speaker verification database. Results confirm that the speaker verification systems trained with specific verification metrics are a good choice to improve the generalization of the learned representations.

Part IV

Conclusions

15

Conclusions and Future Work

15.1 Conclusions

In recent years, deep learning techniques have became the dominant approaches in many different fields and tasks, including biometric recognition. As advanced as these techniques are, they still have some problems when the task has limited data or a successful approach in one task is intended to be used for another task. Therefore, along this thesis, we have presented different alternative approaches to deal with these issues. First, we have focused our efforts on improving the generation of signal representations, also known as representation learning, for the text-dependent speaker verification task, since this task has a strong dependency of the phonetic content. Thus, not only the information related to the speaker identity is needed to generate a good enough representation for the final verification process. While in the last part of the thesis, we have analyzed the fact that the verification systems available in the state-of-the-art are not always optimized towards the goal task. Hence, we have proposed several approaches using new training loss functions that are based on the final verification metrics. These training loss

functions can be applied for each verification task. The following subsections summarise and present the conclusions for each specific part of this dissertation.

15.1.1 Representation Learning

At the beginning of this thesis, we established the baseline systems for face and text-dependent speaker verification systems using state-of-the-art deep learning architectures. However, in Chapter 4, we discovered that these architectures did not perform as we expected for the text-dependent speaker verification task. Thus, an analysis of what it could be happening was carried out in Chapter 5. With this analysis, we observed that the use of deep neural networks (DNNs) combined with global average pooling was not the most suitable approach for the text-dependent speaker verification task. This issue could be motivated by the fact that the order of the phonetic information in this task is relevant, since the speaker identity and the correct phrase are jointly verified to grant access to the systems. For this reason, in this thesis, we presented a new successful approach based on replacing the global average pooling by an alignment mechanism. The use of the alignment mechanism allows us to keep the temporal structure and encode the speaker and phrase information in a neural network supervector as representation for each utterance. Moreover, different types of alignment mechanism, such as Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM) combined with Maximum A Posteriori (MAP), were employed to confirm that this approach achieved competitive results on the RSR2015-Part I and Part II text-dependent speaker verification database. On the other hand, the effect of varying the size of the training data was also studied to check that an improvement could be achieved if the available training data was larger.

Aside from the issue of phonetic information, with the above analysis, we concluded that the limited amount of training data in the RSR2015 database was another relevant problem to the application of powerful DNNs such as those used in face or text-independent speaker verification. This lack of data could produce overconfident predictions. To address this problem in Chapter 6, we introduced an architecture based on the Knowledge Distillation (KD) approach, which consists of two simultaneously trained networks, known as teacher-student architecture. Using this architecture, the student network learns to mimic the predictions of the teacher network. Thus, this approach provides robustness to the systems during the training process. In addition, we included two alternatives to introduce variability in the input of the networks with the Random Erasing (RE) data augmentation technique which helps to handle a potential overfitting issue due to the lack of data. The results achieved with the RSR2015-Part I and Part II database confirmed that the teacher-student architectures improved the generalization capability and better model the variability introduced by the input signals.

The alignment mechanism introduced previously showed to be effective for the text-dependent speaker verification task, but this technique has a main problem, since the temporal alignment is obtained using an external method such as HMM or GMM. Therefore, in Chapter 7, we presented an alternative architecture based on residual blocks with

a different type of processing for temporal information that is composed of Multi-head Self-Attention (MSA) layers combined with phonetic embeddings and memory layers. MSA layers allow the model to focus on the most relevant frames of the sequence to keep phonetic information and better discriminate between utterances and speakers. The use of phonetic embeddings compared to standard positional embeddings improves the performance of attention mechanism, as these embeddings help to guide the attention mask to focus on certain phonetic information. Furthermore, the model capacity was improved by the use of the memory layers. Apart from these techniques, in Chapter 8, we also introduced in the above architecture two learnable tokens which are called class and distillation tokens. These tokens were concatenated to the input before the first MSA layer and used to obtain a global utterance descriptor similar to a supervector approach. The combination of these techniques was evaluated on the RSR2015-Part II and DeepMine-Part I text-dependent speaker verification databases. Results achieved shown on both databases the power of this kind of approach, even for the RSR2015 database which has suffered of overfitting problems when larger DNN architectures were used in the initial attempts of this thesis due to the small size of the database.

15.1.2 Metric Learning

Throughout the previous part, robust representations were obtained using different approaches based on DNNs. Nevertheless, all the presented approaches were trained with the same loss function which was the traditional Cross-Entropy (CE) loss combined with Ring loss as complementary loss. Although this training strategy provided reasonably good results, it was not designed oriented to optimize the verification task itself. Moreover, many systems trained with this strategy usually apply a back-end to perform the verification process. Therefore, the first approach proposed in this part was the use of a triplet neural network as back-end with a training loss function based on the Area Under the ROC Curve (AUC) metric described in Chapter 10. This metric is employed to evaluate performance in verification systems. Thus, we presented a differentiable approximation of this metric called *aAUC* loss which allows training the triplet neural network with an objective loss oriented to the goal task. In order to correctly train this kind of approach, the triplet data selection employed was very important. For this reason, we implemented a smart algorithm to carry out the selection of this triplet training data. The use of this type of training strategy achieved a great influence on the performance of the system for each of the verification tasks in which it was applied, since this approach was employed to develop text-dependent speaker, language and face verification systems using the RSR2015 database, LRE databases and MOBIO dataset respectively.

Metric learning approaches such as the triplet neural network with *aAUC* loss proposed in this thesis have achieved relevant success in improving the generalization capability of verification systems. However, these techniques combined with smart data selection have a high computational cost. Therefore, different approaches were proposed to take advantage of the efficiency and speed of multi-class training while a loss function focused on the goal task was optimized. In Chapter 11, approximated Detection

Cost Function (aDCF) was developed, since the original DCF metric was based on measuring the cost of decision errors of verification systems in terms of misses and false alarms. With this loss function, we trained a text-dependent verification system using the RSR2015-Part I and Part II database and the results achieved a great performance in both parts. Apart from evaluating the effectiveness of this loss function for training multi-class DNN architectures, in this chapter, we also checked the effects of employing a cosine layer as the last layer in the DNN architecture, and compared the performance with some of the state-of-the-art loss functions. Nevertheless, the systems with aDCF loss were trained using one model for each phrase, so it had some problems when score normalization was not applied to achieve the final performance. Thus, motivated by this issue, we presented a new straightforward back-end that was applied to the DNN architecture trained with aDCF loss to improve the discrimination ability and mitigate the effect of not using a score normalization. This novel back-end employed the matrix from the last layer of the DNN architecture combined with the enrollment data to train enrollment models for each identity as a binary task. Thus, this training strategy mimicked the final verification process which improved the system performance and also, the calibration of the resulting systems for the text-dependent speaker verification task with the RSR2015-Part II database. Moreover, in Chapter 13, this new back-end approach was applied to a multimodal diarization task where face enrollment models were trained for each identity. The use of these models for identity assignment allowed us to achieve a relevant improvement in the IberSPEECH-RTVE 2020 Multimodal Diarization Challenge over average embeddings directly and the application of only a simple cosine similarity.

aDCF loss was proven to be an effective approach to train DNN architectures, but this loss function had a major drawback since it was an application-dependent metric and needed some prior or cost parameter assumptions to be used. In addition, an approximation of the real DCF metric had to be made to use it as objective loss function. Hence, as an alternative approach with the same idea of aDCF loss, in Chapter 14, Log-Likelihood Ratio Cost (CLLR) was implemented. CLLR is an application-independent evaluation metric which measures the expected log costs of the scores generated for each example by the systems. Therefore, using CLLR as objective loss, the end-to-end system was trained to learn to minimize these costs by obtaining good scores. The performance of systems trained with CLLR loss was evaluated on the RSR2015-Part II text-dependent speaker verification database, and the results confirmed the improvement achieved in the generalization of the learned representations when one of the specific verification metrics was applied to develop the systems.

15.2 Award and Research Contributions

The research conducted during this PhD Thesis has produced multiple contributions to peer-review journals and conference proceedings. These contributions are detailed below.

15.2.1 Award

- Best Paper Award. **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Differentiable Supervector Extraction for Encoding Speaker and Phrase Information in Text Dependent Speaker Verification**. *Proceedings of IberSPEECH 2018*, pp. 1–5.

15.2.2 Journal Articles

- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Supervector Extraction for Encoding Speaker and Phrase Information with Neural Networks for Text-Dependent Speaker Verification**. *Applied Sciences*, vol. 9, no. 16, p. 3295, 2019.
- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Optimization of the area under the ROC curve using neural network supervectors for text-dependent speaker verification**. *Computer Speech Language*, vol. 63,p. 101078, 2020.
- P. Gimeno, **V. Mingote**, A. Ortega, A. Miguel, and E. Lleida, **Generalising AUC Optimisation to Multiclass Classification for Audio Segmentation with Limited Training Data**. *IEEE Signal Processing Letters*, vol.28, p. 1135-1139, 2021.
- **V. Mingote**, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, **aDCF Loss Function for Deep Metric Learning in End-to-End Text-Dependent Speaker Verification Systems.** *IEEE/ACM Transactions on Audio, Speech and Language*, vol. 30, pp. 772-784, 2022.
- **V. Mingote**, I. Viñals, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, **Multimodal Diarization Systems by Training Enrollment Models as Identity Representations**. *Applied Sciences*, vol. 12, no. 3, p. 1141, 2022.
- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Class Token and Knowledge Distillation for Multi-head Self-Attention Speaker Verification Systems.** arXiv preprint arXiv:2111.03842, *Submitted to IEEE/ACM Transactions on Audio, Speech and Language.*

15.2.3 Conference Papers

- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Differentiable Supervector Extraction for Encoding Speaker and Phrase Information in Text Dependent Speaker Verification**. *Proceedings of IberSPEECH 2018*, pp. 1–5.
- **V. Mingote**, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, **Optimization of False Acceptance/Rejection Rates and Decision Threshold for End-to-End Text-Dependent Speaker Verification Systems**. *Proceedings of INTERSPEECH 2019*, pp.2903-2907.

- **V. Mingote**, D. Castan, M. McLaren, M. K. Nandwana, A. Ortega, and E. Lleida, A. Miguel, **Language Recognition using Triplet Neural Networks**. *Proceedings of INTERSPEECH 2019*, pp. 4025-4029.
- I. Viñals, D. Ribas, **V. Mingote**, J. Llombart, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, **Phonetically-aware embeddings, Wide Residual Networks with Time-Delay Neural Networks and Self Attention models for the 2018 NIST Speaker Recognition Evaluation**. *Proceedings of INTERSPEECH 2019*, pp. 4310-4314.
- **V. Mingote**, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, **Knowledge Distillation and Random Erasing Data Augmentation for Text-Dependent Speaker Verification**. *Proceedings of ICASSP 2020*, pp. 6824-6828.
- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Training Speaker Enrollment Models by Network Optimization**. *Proceedings of INTERSPEECH 2020*, pp. 3810-3814.
- P. Gimeno, **V. Mingote**, A. Ortega, A. Miguel, and E. Lleida, **Partial AUC Optimisation using Recurrent Neural Networks for Music Detection with Limited Training Data**. *Proceedings of INTERSPEECH 2020*, pp. 3067-3071.
- **V. Mingote**, I. Viñals, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, **ViVoLAB Multimodal Diarization System for RTVE 2020 Challenge**. *Proceedings of IberSPEECH 2020*, pp. 76-80.
- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Memory Layers with Multi-Head Attention Mechanism for Text Dependent Speaker Verification**. *Proceedings of ICASSP 2021*, pp. 6154-6158.
- **V. Mingote**, A. Miguel, A. Ortega, and E. Lleida, **Log-Likelihood-Ratio Cost Function as Objective Loss for Speaker Verification Systems.** *Proceedings of INTERSPEECH 2021*, pp. 2361-2365.

15.3 Future Research

Along this thesis, we have developed different approaches to enhance many parts of the DNN architecture that improves the training process and the power to learn more general representations. Although each of these approaches present improvements for every shown scenario, there are still some limitations that may be solved in future works. Furthermore, several new challenges have been created during this thesis. Some future research lines to study and address the limitations and challenges are described below.

- As noted in the conclusions, architectures based on MSA layers combined with memory layers and class and distillation tokens have shown to be very effective and

promising for text-dependent speaker verification, even when the data available to train the systems is not too large. In addition, the introduction of this MSA and memory mechanism based on self-attention allows to enhance the results and also the interpretability of them, as we showed in the initial analysis carried out in Chapter 8. For this reason, an interesting line of research could be focused on a wide analysis of the information learned by this type of mechanism that allows a better understanding of the trained verification systems.

- Moreover, since the previous architectures employ the phonetic embeddings to help to guide the attention mask of the MSA layers, we could also investigate the option of replacing these embeddings with a learnable matrix which learns during training similar information and avoids the need of this external information.

- On the other hand, the audio processing step used in the architectures based on MSA to obtain the features was initially fixed for each text-dependent speaker verification database and we did not do any exploration related to these configurations. Thus, we could think about combining them or propose another alternatives to use them as input for training the architectures.

- Another line of future work could focus on improving the audiovisual systems developed in this thesis. Therefore, we could also introduce MSA layers with class token to fully train our own face verification system thanks to which we could replace the pre-trained model used.

- Furthermore, in the initial face processing stage of verification systems, the fundamental detection step produces as output the detected faces and also, the landmarks of where the main points of these faces are located in the images. Thus, this information could be included in the DNN training to keep the structure of the face which would be similar to the states of the supervectors presented for text-dependent speaker verification systems.

- Regarding the loss function employed to train the architectures based on MSA layers with class tokens, we used the traditional CE loss. So we conjecture that a benefit could still be achieved by applying some of the approaches proposed in the metric learning part of this thesis to train these architectures such as aDCF loss or CLLR loss. In addition, we could also incorporate to develop our architectures for face verification one of this loss function as objective training losses.

- In the case of multimodal diarization system, we noted that there is still much room for improvement, so it could also be interesting to search effective fusion strategies for the speaker and face subsystems. This fusion could involve a great improvement as it could help to disambiguate the identification process.

- Finally, the vast majority of our findings can also be extended and applied to the text-independent speaker verification systems, and also to other fields.

Part V

Appendix

A Detailed Architectures

A.1 Introduction

This appendix provides a detailed explanation of the architectures employed throughout this thesis to develop each proposed approach. To define the architectures used in each part of the thesis, we have divided the information of the models into the following two sections. The former contains the architectures of Part II, while the latter has the architectures employed in Part III.

A.2 Architectures Part II

A.2.1 Baseline Systems

In this section, the architectures used to develop the baseline systems of this thesis are explained. First, a Wide Residual Neural Network (WideResnet) has been implemented in this thesis to establish the face verification baseline system detailed in Table A.1. On the other hand, to develop the baseline system for text-dependent speaker verification, two architectures have been employed, a WideResnet described in Table A.2 and a Convolutional Neural Network (CNN) detailed in Table A.3.

Table A.1: Topology for WideResnet architecture for face verification baseline.

Layer	Layer type	Channels	Output
	Input	3	3×160×160
1	Conv2D-Relu	32	32×80×80
2	Conv2D-Relu	32	32×40×40
3	WRNBlock-ReLU(x4)	160	160×20×20
4	WRNBlock-ReLU(x4)	320	320×10×10
5	WRNBlock-ReLU(x4)	640	640×5×5
6	BatchNorm2D	640	640×5×5
7	Average	–	640
8	Linear+Softmax	–	N
	CE Loss		

Table A.2: Topology for WideResnet architecture for text-dependent speaker verification baseline.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	256	256×200
2	Conv1D-LRelu	256	256×200
3	WRNBlock-LReLU(x4)	160	160×100
4	WRNBlock-LReLU(x4)	320	320×50
5	WRNBlock-LReLU(x4)	640	640×25
6	BatchNorm1D	640	640×25
7	Average	–	640
8	Linear+Softmax	–	N
	CE Loss		

Table A.3: Topology for CNN architecture for text-dependent speaker verification baseline.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×100
2	Conv1D-LRelu	128	128×50
3	Conv1D	32	32×25
4	Average	–	32
5	Linear+Softmax	–	N
	CE Loss + Ring Loss		

A.2.2 Architectures based on Alignment Mechanism

The following section includes the description of the architectures developed in this thesis for text-dependent speaker verification systems to replace the global average pooling

of Deep Neural Networks (DNNs) by an alignment mechanism such as Hidden Markov Model (HMM) or Gaussian Mixture Model (GMM) combined with Maximum A Posteriori (MAP). First, different configurations of the number of layers of a CNN combined with both alignment mechanisms are described in Tables A.4, A.5 and A.6, where SC indicates the number of HMM states or GMM components used. Moreover, in this type of approach using the alignment mechanisms, another philosophy of architecture based on Knowledge Distillation (KD) and Teacher-Student architectures was proposed. Therefore, Table A.7 and A.8 describe the configuration of the teacher network and the student network employed.

Table A.4: Topology for CNN architecture with one convolutional layer combined with alignment mechanism for text-dependent speaker verification. SC indicates the number of HMM states or GMM components, and N is the number of classes.

Layer	**Layer type**	**Channels**	**Output**
	Input	60	60×200
1	Conv1D	32	32×200
2	Alignment Mechanism	–	$32 * SC$
3	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.5: Topology for CNN architecture with three convolutional layers combined with alignment mechanism for text-dependent speaker verification. SC indicates the number of HMM states or GMM components, and N is the number of classes.

Layer	**Layer type**	**Channels**	**Output**
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D-LRelu	128	128×200
3	Conv1D	32	32×200
4	Alignment Mechanism	–	$32 * SC$
5	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.6: Topology for CNN architecture with four convolutional layers combined with alignment mechanism for text-dependent speaker verification. *SC* indicates the number of HMM states or GMM components, and *N* is the number of classes.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D-LRelu	128	128×200
3	Conv1D-LRelu	256	256×200
4	Conv1D	32	32×200
5	Alignment Mechanism	–	$32 * SC$
6	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.7: Topology for CNN architecture combined with alignment mechanism for teacher neural network for text-dependent speaker verification. *SC* indicates the number of HMM states or GMM components, and *N* is the number of classes.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D	32	32×200
3	Alignment Mechanism	–	$32 * SC$
4	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.8: Topology for CNN architecture combined with alignment mechanism for student neural network for text-dependent speaker verification. *SC* indicates the number of HMM states or GMM components, and *N* is the number of classes.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D	32	32×200
3	Alignment Mechanism	–	$32 * SC$
4	Linear+Softmax	–	N
	KLD Loss + Ring Loss		

A.2.3 Architectures with Multi-head Self-Attention and Memory Layers

The last group of architectures employed in this part have been composed of Residual Neural Network (ResBlock) combined with Multi-head Self-Attention (MSA) and Memory Layers. Table A.9 shows the basic architecture with these approaches and applying a global average pooling as reduction mechanism. As an evolution of this archi-

tecture, a teacher-student architecture has been developed. Thus, Table A.10 describes the teacher network configuration and Table A.11 details the student network. Finally, on this teacher-student architecture, a new approach has been proposed to replace the global average pooling. In this approach, a class token for the teacher network and class and distillation tokens for the student network are used as output representations. The configuration of these two networks is shown in Tables A.12 and A.13.

Table A.9: Topology for ResBlock, MSA and Memory layers architecture for text-dependent speaker verification. N is the number of classes.

Layer	**Layer type**	**Channels**	**Output**
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Average	–	256
10	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.10: Topology for ResBlock, MSA and Memory layers architecture for teacher neural network for text-dependent speaker verification. N is the number of classes.

Layer	**Layer type**	**Channels**	**Output**
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Average	–	256
10	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.11: Topology for ResBlock, MSA and Memory layers architecture for student neural network for text-dependent speaker verification. N is the number of classes.

Layer	Layer type	Channels	Output
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Average	–	256
10	Linear+Softmax	–	N
	KLD Loss + Ring Loss		

Table A.12: Topology for ResBlock, MSA and Memory layers architecture with class token for teacher neural network for text-dependent speaker verification. N is the number of classes.

Layer	Layer type	Channels	Output
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Class Token	–	256
10	Linear+Softmax	–	N
	CE Loss + Ring Loss		

Table A.13: Topology for ResBlock, MSA and Memory layers architecture with class and distillation tokens for student neural network for text-dependent speaker verification. N is the number of classes.

Layer	Layer type	Channels	Output
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
$9a$	Distillation Token	–	256
$9b$	Class Token	–	256
$10a$	Linear+Softmax	–	N
$10b$	Linear+Softmax	–	N
a	KLD Loss + Ring Loss		
b	CE Loss + Ring Loss		

A.3 Architectures Part III

A.3.1 Triplet Neural Network as Back-end Network

The first type of architectures trained in Part III consists of triplet neural networks. These networks are used as back-end network for different verification tasks. Table A.14 contains the topology information of the architecture employed for the case of face and speaker verification systems. Whereas Table A.15 shows the configuration of the triplet neural network for language recognition systems.

Table A.14: Topology for Triplet Neural Network for back-end network for face and text-dependent speaker verification. SC indicates the number of HMM states or GMM components.

Layer	Layer type	Channels	Output
	Input	–	128/640/32 $*$ SC
1	Linear	–	3000
2	Linear	–	1024
	aAUC Loss		

Table A.15: Topology for Triplet Neural Network for back-end network for language verification.

Layer	Layer type	Channels	Output
	Input	–	512
1	Linear	–	1024
	aAUC Loss		

A.3.2 Convolutional Neural Networks for Metric Learning Loss Functions

In this section, the CNN teacher-student architecture with alignment mechanism developed in Part II for the text-dependent speaker verification task is employed to implement a new loss function oriented to the goal task. This function is aDCF loss and the exact architecture is described in Tables A.16 and A.17. Moreover, to compare with other state-of-the-art loss functions, the same architecture using Angular Softmax loss is developed and detailed in Tables A.18 and A.19.

Table A.16: Topology for CNN architecture combined with alignment mechanism and aDCF loss for teacher neural network for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D	32	32×200
3	Alignment Mechanism	–	32 * 64
4	Linear/Cosine	–	N
	aDCF Loss		

Table A.17: Topology for CNN architecture combined with alignment mechanism and aDCF loss for student neural network for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D	32	32×200
3	Alignment Mechanism	–	32 * 64
4	Linear/Cosine	–	N
	aDCF Loss		

Table A.18: Topology for CNN architecture combined with alignment mechanism and A-Softmax loss for teacher neural network for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D	32	32×200
3	Alignment Mechanism	–	32 * 64
4	Angle Linear	–	N
	A-Softmax Loss		

Table A.19: Topology for CNN architecture combined with alignment mechanism and A-Softmax loss for student neural network for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×200
1	Conv1D-LRelu	64	64×200
2	Conv1D	32	32×200
3	Alignment Mechanism	–	32 * 64
4	Angle Linear	–	N
	A-Softmax Loss		

A.3.3 Residual Neural Networks with Multi-head Self-Attention and Memory Layers for Metric Learning Loss Functions

To conclude, the last architectures developed in this thesis are based on the Residual Neural Networks (ResBlock) with Multi-head Self-Attention (MSA) and Memory layers implemented in Part II. Taking the basic architecture of Table A.9, we have modified the loss function used in that architecture to validate the proposed new loss functions and also other state-of-the-art loss function for text-dependent speaker verification tasks. Therefore, Table A.20 presents the architecture with CLLR loss, Table A.21 has the modified architecture for aDCF loss, and finally, Table A.22 shows the architecture with A-Softmax loss.

Table A.20: Topology for ResBlock, MSA and Memory layers architecture with CLLR loss for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Average	–	256
10	Cosine	–	N
	CLLR Loss		

Table A.21: Topology for ResBlock, MSA and Memory layers architecture with aDCF loss for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Average	–	256
10	Cosine	–	N
	aDCF Loss		

Table A.22: Topology for ResBlock, MSA and Memory layers architecture with A-Softmax loss for text-dependent speaker verification.

Layer	Layer type	Channels	Output
	Input	60	60×300
1	Conv1D	128	128×300
2	ResBlock-ReLU(x3)	160	160×300
3	ResBlock-ReLU(x3)	256	256×300
4	BatchNorm1D	256	256×300
5	MSA (heads=16)	256	256×300
6	Memory	256	256×300
7	MSA (heads=16)	256	256×300
8	Memory	256	256×300
9	Average	–	256
10	Angle Linear	–	N
	A-Softmax Loss		

References

[1] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Differentiable Supervector Extraction for Encoding Speaker and Phrase Information in Text Dependent Speaker Verification," *Proceedings of IberSPEECH 2018*, pp. 1–5, 2018. [Online]. Available: http://dx.doi.org/10.21437/IberSPEECH.2018-1

[2] ——, "Supervector Extraction for Encoding Speaker and Phrase Information with Neural Networks for Text-Dependent Speaker Verification," *Applied Sciences*, vol. 9, no. 16, p. 3295, 2019.

[3] ——, "Optimization of the area under the ROC curve using neural network supervectors for text-dependent speaker verification," *Computer Speech & Language*, vol. 63, p. 101078, 2020.

[4] V. Mingote, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, "Knowledge Distillation and Random Erasing Data Augmentation for Text-Dependent Speaker Verification," *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6824–6828, 2020.

[5] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Memory Layers with Multi-Head Attention Mechanisms for Text-Dependent Speaker Verification," *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6154–6158, 2021.

[6] ——, "Class Token and Knowledge Distillation for Multi-head Self-Attention Speaker Verification Systems," *arXiv preprint arXiv:2111.03842*, 2021.

[7] V. Mingote, D. Castan, M. McLaren, M. K. Nandwana, A. Ortega, E. Lleida, and A. Miguel, "Language Recognition Using Triplet Neural Networks." *Proceedings of INTERSPEECH 2019*, pp. 4025–4029, 2019.

[8] V. Mingote, A. Miguel, D. Ribas, A. Ortega, and E. Lleida, "Optimization of False Acceptance/Rejection Rates and Decision Threshold for End-to-End Text-Dependent Speaker Verification Systems," *Proceedings of INTERSPEECH 2019*, pp. 2903–2907, 2019.

[9] ——, "aDCF Loss Function for Deep Metric Learning in End-to-End Text-Dependent Speaker Verification Systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 772–784, 2022.

[10] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Training Speaker Enrollment Models by Network Optimization," *Proceedings of INTERSPEECH 2020*, pp. 3810–3814, 2020.

[11] V. Mingote, I. Vinals, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, "ViVoLAB Multimodal Diarization System for RTVE 2020 Challenge," *Proceedings of IberSPEECH 2021*, pp. 76–80, 2021.

[12] ——, "Multimodal Diarization Systems by Training Enrollment Models as Identity Representations," *Applied Sciences*, vol. 12, no. 3, p. 1141, 2022.

[13] V. Mingote, A. Miguel, A. Ortega, and E. Lleida, "Log-Likelihood-Ratio Cost Function as Objective Loss for Speaker Verification Systems," *Proceedings of INTERSPEECH 2021*, pp. 2361–2365, 2021.

[14] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705–741, 1995.

[15] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM computing surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[16] R. Jafri and H. R. Arabnia, "A survey of face recognition techniques," *Journal of information processing systems*, vol. 5, no. 2, pp. 41–68, 2009.

[17] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, 2021.

[18] M. Taskiran, N. Kahraman, and C. E. Erdem, "Face recognition: Past, present and future (a review)," *Digital Signal Processing*, p. 102809, 2020.

[19] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, present, and future of face recognition: A review," *Electronics*, vol. 9, no. 8, p. 1188, 2020.

[20] Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, "Face recognition systems: A survey," *Sensors*, vol. 20, no. 2, p. 342, 2020.

[21] J. P. Campbell, "Speaker recognition: A tutorial," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, 1997.

[22] F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-García, D. Petrovska-Delacrétaz, and D. A. Reynolds, "A tutorial on text-independent speaker verification," *EURASIP Journal on Advances in Signal Processing*, no. 4, pp. 1–22, 2004.

[23] S. Furui, "50 years of progress in speech and speaker recognition research," *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, vol. 1, no. 2, pp. 64–74, 2005.

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

[25] S. Debnath, B. Soni, U. Baruah, and D. Sah, "Text-dependent speaker verification system: A review," *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1–7, 2015.

[26] C. D. Shaver and J. M. Acken, "A brief review of speaker recognition technology," 2016.

[27] Z. Bai and X.-L. Zhang, "Speaker recognition based on deep learning: An overview," *Neural Networks*, vol. 140, pp. 65–99, 2021.

[28] R. M. Hanifa, K. Isa, and S. Mohamad, "A review on speaker recognition: Technology and challenges," *Computers & Electrical Engineering*, vol. 90, p. 107005, 2021.

[29] Y. Muthusamy, *A review of research in automatic language identification.* Citeseer, 1992.

[30] D. Martínez González, E. Lleida Solano, and A. Miguel Artiaga, "Subspace Gaussian Mixture Models for Language Identification and Dysarthric Speech Intelligibility Assessment," Ph.D. dissertation, Universidad de Zaragoza, 2015.

[31] O. P. Singh, "Exploration of sparse representation techniques in language recognition," Ph.D. dissertation, 2019.

[32] I. S. Bruner and R. Tagiuri, "The perception of people," *Handbook of Social Psychology*, vol. 2, p. 634–654, 1954.

[33] W. W. Bledsoe, "The model method in facial recognition," Panoramic research Inc., Palo Alto, CA, Tech. Rep., 1964.

[34] M. D. Kelly, *Visual identification of people by computer.* Department of Computer Science, Stanford University., 1970, no. 130.

[35] T. Kanade, "Picture processing system by computer complex and recognition of human faces," 1974.

[36] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Josa a*, vol. 4, no. 3, pp. 519–524, 1987.

[37] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[38] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pp. 586–587, 1991.

[39] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 696–710, 1997.

[40] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[41] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[42] M. S. Bartlett and T. J. Sejnowski, "Independent components of face images: A representation for face recognition," *Proceedings of 4th Annual J. Symp. Neural Computation*, pp. 3–10, 1997.

[43] L. Wiskott, N. Krüger, N. Kuiger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[44] A. Lanitis, C. J. Taylor, and T. F. Cootes, "Automatic face identification system using flexible appearance models," *Image and vision computing*, vol. 13, no. 5, pp. 393–401, 1995.

[45] J. F. G. Pérez, A. F. Frangi, E. L. Solano, and K. Lukas, "Lip reading for robust speech recognition on embedded devices," *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, pp. I–473, 2005.

[46] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[47] C. I. Podilchuk and X. Zhang, "Face recognition using DCT-based feature vectors," Sep. 1 1998, uS Patent 5,802,208.

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

[49] F. Samaria and S. Young, "Hmm-based architecture for face identification," *Image and vision computing*, vol. 12, no. 8, pp. 537–543, 1994.

[50] B. Achermann and H. Bunke, "Combination of face classifiers for person identification," *Proceedings of 13th International Conference on Pattern Recognition*, vol. 3, pp. 416–420, 1996.

[51] A. V. Nefian and M. H. Hayes, "Face detection and recognition using hidden Markov models," *Proceedings 1998 international conference on image processing (ICIP)*, vol. 1, pp. 141–145, 1998.

[52] V. V. Kohir and U. B. Desai, "Face recognition using a DCT-HMM approach," *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. (WACV)*, pp. 226–231, 1998.

[53] P. Phillips, "Support vector machines applied to face recognition," *Advances in Neural Information Processing Systems*, vol. 11, pp. 803–809, 1998.

[54] G. Guo, S. Z. Li, and K. Chan, "Face recognition by support vector machines," *Proceedings fourth IEEE international conference on automatic face and gesture recognition*, pp. 196–201, 2000.

[55] O. Déniz, M. Castrillon, and M. Hernández, "Face recognition using independent component analysis and support vector machines," *Pattern recognition letters*, vol. 24, no. 13, pp. 2153–2157, 2003.

[56] A. Tefas, C. Kotropoulos, and I. Pitas, "Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 7, pp. 735–746, 2001.

[57] K. Fukunaga, *Instruction to Statistical Pattern Recognition.* Elsevier, 1972.

[58] R. O. Duda, P. E. Hart *et al.*, *Pattern classification and scene analysis.* Wiley New York, 1973, vol. 3.

[59] R. Gross, J. Yang, and A. Waibel, "Growing Gaussian mixture models for pose invariant face recognition," *Proceedings 15th International Conference on Pattern Recognition (ICPR)*, vol. 1, pp. 1088–1091, 2000.

[60] C. Sanderson and K. K. Paliwal, "Fast features for face authentication under illumination direction changes," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2409–2419, 2003.

[61] S. Lucey and T. Chen, "A GMM parts based face representation for improved verification through relevance adaptation," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. II–II, 2004.

[62] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[63] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," *European conference on computer vision*, pp. 469–481, 2004.

[64] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[65] S. J. Prince and J. H. Elder, "Tied Factor Analysis for Face Recognition Across Large Pose Changes." *BMVC*, vol. 3, pp. 889–898, 2006.

[66] S. J. Prince, J. H. Elder, J. Warrell, and F. M. Felisberti, "Tied factor analysis for face recognition across large pose differences," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 6, pp. 970–984, 2008.

[67] S. Ioffe, "Probabilistic linear discriminant analysis," *European Conference on Computer Vision*, pp. 531–542, 2006.

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

[69] R. Wallace, M. McLaren, C. McCool, and S. Marcel, "Inter-session variability modelling and joint factor analysis for face authentication," *2011 International Joint Conference on Biometrics (IJCB)*, pp. 1–8, 2011.

[70] R. Wallace and M. McLaren, "Total variability modelling for face verification," *Iet Biometrics*, vol. 1, no. 4, pp. 188–199, 2012.

[71] A. Krizhevsky, I. Sulskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information and Processing Systems (NIPS)*, pp. 1–9, 2012.

[72] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[73] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 539–546, 2005.

[74] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," *Asian conference on computer vision*, pp. 709–720, 2010.

[75] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[76] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE conference on computer vision and pattern recognition (ICCV)*, pp. 1–9, 2015.

[77] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," 2015.

[78] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

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

[80] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range loss for deep face recognition with long-tailed training data," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5409–5418, 2017.

[81] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

[82] S. Zagoruyko and N. Komodakis, "Wide Residual Networks," *Arxiv*, pp. 1–15, 2016. [Online]. Available: http://arxiv.org/abs/1605.07146

[83] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," *European conference on computer vision*, pp. 499–515, 2016.

[84] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for face recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5089–5097, 2018.

[85] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

[86] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

[87] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 1735–1742, 2006.

[88] G. Fant, *Acoustic theory of speech production.* Walter de Gruyter, 1970, no. 2.

[89] L. G. Kersta, "Voiceprint identification," *The Journal of the Acoustical Society of America*, vol. 34, no. 5, pp. 725–725, 1962.

[90] S. Pruzansky, "Pattern-Matching Procedure for Automatic Talker Recognition," *The Journal of the Acoustical Society of America*, vol. 35, no. 3, pp. 354–358, 1963.

[91] G. R. Doddington, "A Method or Speaker Verification," *The Journal of the Acoustical Society of America*, vol. 49, no. 1A, pp. 139–139, 1971.

[92] J. E. Luck, "Automatic speaker verification using cepstral measurements," *The Journal of the Acoustical Society of America*, vol. 46, no. 4B, pp. 1026–1032, 1969.

[93] B. S. Atal, "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification," *the Journal of the Acoustical Society of America*, vol. 55, no. 6, pp. 1304–1312, 1974.

[94] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 2, pp. 254–272, 1981.

[95] K. Li and E. Wrench Jr, "Text-independent speaker recognition with short utterances," *The Journal of the Acoustical Society of America*, vol. 72, no. S1, pp. S29–S30, 1982.

[96] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[97] G. R. Doddington, "Speaker recognition—Identifying people by their voices," *Proceedings of the IEEE*, vol. 73, no. 11, pp. 1651–1664, 1985.

[98] F. K. Soong, A. E. Rosenberg, B.-H. Juang, and L. R. Rabiner, "Report: A vector quantization approach to speaker recognition," *AT&T technical journal*, vol. 66, no. 2, pp. 14–26, 1987.

[99] J. Buck, D. Burton, and J. Shore, "Text-dependent speaker recognition using vector quantization," *1985 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 10, pp. 391–394, 1985.

[100] A. Higgins and R. Wohlford, "A new method of text-independent speaker recognition," *1986 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 11, pp. 869–872, 1986.

[101] M. Savic and S. K. Gupta, "Variable parameter speaker verification system based on hidden Markov modeling," *1990 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 281–284, 1990.

[102] N. Z. Tisby, "On the application of mixture AR hidden Markov models to text independent speaker recognition," *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 563–570, 1991.

[103] C. Che and Q. Lin, "Speaker recognition using HMM with experiments on the YOHO database," *Fourth European Conference on Speech Communication and Technology*, 1995.

[104] D. A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech communication*, vol. 17, no. 1-2, pp. 91–108, 1995.

[105] D. A. Reynolds and B. A. Carlson, "Text-dependent speaker verification using decoupled and integrated speaker and speech recognizers." *Eurospeech*, 1995.

[106] D. A. Reynolds, R. C. Rose *et al.*, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

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

[108] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field, and M. Contolini, "Eigenvoices for speaker adaptation," *Fifth International Conference on Spoken Language Processing*, 1998.

[109] O. Thyes, R. Kuhn, P. Nguyen, and J.-C. Junqua, "Speaker identification and verification using eigenvoices," *Sixth International Conference on Spoken Language Processing*, 2000.

[110] P. Nguyen, R. Kuhn, J.-C. Junqua, N. Niedzielski, and C. Wellekens, "Eigenvoices: a compact representation of speakers in model space," *Annales des télécommunications*, vol. 55, no. 3, pp. 163–171, 2000.

[111] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 695–707, 2000.

[112] G.-J. Jang, S.-J. Yun, and Y.-H. Oh, "Feature vector transformation using independent component analysis and its application to speaker identification," *Sixth European Conference on Speech Communication and Technology*, 1999.

[113] G.-J. Jang, T.-W. Lee, and Y.-H. Oh, "Learning statistically efficient features for speaker recognition," *Neurocomputing*, vol. 49, no. 1-4, pp. 329–348, 2002.

[114] A. F. Martin and M. A. Przybocki, "The NIST speaker recognition evaluations: 1996-2001," *2001: A Speaker Odyssey-The Speaker Recognition Workshop*, 2001.

[115] V. Wan and W. M. Campbell, "Support vector machines for speaker verification and identification," *Proceedings of the 2000 IEEE Signal Processing Society Workshop*, vol. 2, pp. 775–784, 2000.

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

[117] P. Kenny and P. Dumouchel, "Experiments in speaker verification using factor analysis likelihood ratios," *Proceedings of Odyssey*, 2004.

[118] P. Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, vol. 14, no. 28-29, p. 2, 2005.

[119] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[120] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." *Proceedings of Odyssey 2010*, vol. 14, 2010.

[121] C. Dong, Y. Dong, J. Li, and H. Wang, "Support vector machines based text dependent speaker verification using HMM supervectors," *Proceedings of Odyssey 2008*, p. 31, 2008.

[122] H. Aronowitz, "Text dependent speaker verification using a small development set," *Proceedings of Odyssey 2012*.

[123] A. Larcher, K. A. Lee, B. Ma, and H. Li, "RSR2015: Database for text-dependent speaker verification using multiple pass-phrases," *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[124] ——, "Text-dependent speaker verification: Classifiers, databases and RSR2015," *Speech Communication*, vol. 60, pp. 56–77, 2014.

[125] S. Cumani, N. Brümmer, L. Burget, P. Laface, O. Plchot, and V. Vasilakakis, "Pairwise Discriminative Speaker Verification in the I-Vector Space," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 6, pp. 1217–1227, 2013.

[126] J. Franco-Pedroso and J. Gonzalez-Rodriguez, "Linguistically-constrained formant-based i-vectors for automatic speaker recognition," *Speech Communication*, vol. 76, pp. 61–81, 2016.

[127] C.-T. Do, C. Barras, V.-B. Le, and A. Sarkar, "Augmenting short-term cepstral features with long-term discriminative features for speaker verification of telephone data," *Proceedings of INTERSPEECH 2013*.

[128] A. K. Sarkar, C.-T. Do, V.-B. Le, and C. Barras, "Combination of cepstral and phonetically discriminative features for speaker verification," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1040–1044, 2014.

[129] A. Lozano-Diez, A. Silnova, P. Matejka, O. Glembek, O. Plchot, J. Pešán, L. Burget, and J. Gonzalez-Rodriguez, "Analysis and optimization of bottleneck features for speaker recognition," *Proceedings of Odyssey 2016*, pp. 352–357, 2016.

[130] Y. Lei, N. Scheffer, L. Ferrer, and M. McLaren, "A novel scheme for speaker recognition using a phonetically-aware deep neural network," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1695–1699, 2014.

[131] O. Ghahabi and J. Hernando, "Deep belief networks for i-vector based speaker recognition," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1700–1704, 2014.

[132] 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," *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 165–170, 2016.

[133] J. Rohdin, A. Silnova, M. Diez, O. Plchot, P. Matějka, and L. Burget, "End-to-end DNN based speaker recognition inspired by i-vector and PLDA," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4874–4878, 2018.

[134] G. Bhattacharya, M. J. Alam, and P. Kenny, "Deep Speaker Embeddings for Short-Duration Speaker Verification," *Proceedings of INTERSPEECH 2017*, pp. 1517–1521, 2017.

[135] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333, 2018.

[136] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification." *Proceedings of INTERSPEECH 2017*, pp. 999–1003, 2017.

[137] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep Speaker Recognition," *Proceedings of INTERSPEECH 2018*, pp. 1086–1090, 2018.

[138] J.-W. Jung, H.-S. Heo, I.-H. Yang, H.-J. Shim, and H.-J. Yu, "Avoiding speaker overfitting in end-to-end dnns using raw waveform for text-independent speaker verification," *Proceedings of INTERSPEECH 2018*, pp. 3583–3587, 2018.

[139] J. Jung, H. Heo, J. Kim, H. Shim, and H. Yu, "RawNet: Advanced End-to-End Deep Neural Network Using Raw Waveforms for Text-Independent Speaker Verification," *Proceedings of INTERSPEECH 2019*, pp. 1268–1272, 2019.

[140] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," *Proceedings of INTERSPEECH 2019*, pp. 4305–4309, 2019.

[141] C. Zhang and K. Koishida, "End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances." *Proceedings of INTERSPEECH 2017*, pp. 1487–1491, 2017.

[142] C. Zhang, K. Koishida, and J. H. Hansen, "Text-independent speaker verification based on triplet convolutional neural network embeddings," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 26, no. 9, pp. 1633–1644, 2018.

[143] Y. Liu, L. He, and J. Liu, "Large Margin Softmax Loss for Speaker Verification," *Proceedings of INTERSPEECH 2019*, pp. 2873–2877, 2019.

[144] W. Cai, J. Chen, and M. Li, "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System," *Proceedings of Odyssey 2018*, pp. 74–81, 2018.

[145] Y. Li, F. Gao, Z. Ou, and J. Sun, "Angular softmax loss for end-to-end speaker verification," *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 190–194, 2018.

[146] Z. Bai, X. Zhang, and J. Chen, "Partial AUC Optimization Based Deep Speaker Embeddings with Class-Center Learning for Text-Independent Speaker Verification," *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6819–6823, 2020.

[147] P. Gimeno, V. Mingote, A. Ortega, A. Miguel, and E. Lleida, "Partial AUC Optimisation Using Recurrent Neural Networks for Music Detection with Limited Training Data." *Proceedings of INTERSPEECH 2020*, pp. 3067–3071, 2020.

[148] ——, "Generalising AUC Optimisation to Multiclass Classification for Audio Segmentation with Limited Training Data," *IEEE Signal Processing Letters*, vol. 28, pp. 1135–1139, 2021.

[149] S. Ramoji, P. Krishnan, and S. Ganapathy, "NPLDA: A Deep Neural PLDA Model for Speaker Verification," *Proceedings of Odyssey 2020*, pp. 202–209, 2020.

[150] J. S. Chung, J. Huh, S. Mun, M. Lee, H.-S. Heo, S. Choe, C. Ham, S. Jung, B.-J. Lee, and I. Han, "In Defence of Metric Learning for Speaker Recognition," *Proceedings of INTERSPEECH 2020*, pp. 2977–2981, 2020.

[151] A. Miguel, J. Villalba, A. Ortega, E. Lleida, and C. Vaquero, "Factor Analysis with Sampling Methods for Text Dependent Speaker Recognition," *Proceedings of INTERSPEECH 2014*, pp. 1342–1346, 2014.

[152] T. Stafylakis, P. Kenny, M. J. Alam, and M. Kockmann, "Speaker and channel factors in text-dependent speaker recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 65–78, 2015.

[153] T. Stafylakis, M. J. Alam, and P. Kenny, "Text-dependent speaker recognition with random digit strings," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 24, no. 7, pp. 1194–1203, 2016.

[154] H. Zeinali, H. Sameti, and L. Burget, "HMM-based phrase-independent i-vector extractor for text-dependent speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 7, pp. 1421–1435, 2017.

[155] G. Valenti, A. Daniel, and N. W. Evans, "On the Influence of Text Content on Pass-Phrase Strength for Short-Duration Text-Dependent Automatic Speaker Authentication." *Proceedings of INTERSPEECH 2016*, pp. 3623–3627, 2016.

[156] R. K. Das, M. Madhavi, and H. Li, "Compensating Utterance Information in Fixed Phrase Speaker Verification," *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1708–1712, 2018.

[157] H. You, W. Li, L. Li, and J. Zhu, "Lexicon-based local representation for text-dependent speaker verification," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 3, pp. 587–589, 2017.

[158] A. Larcher, P.-M. Bousquet, K. A. Lee, D. Matrouf, H. Li, and J.-F. Bonastre, "I-vectors in the context of phonetically-constrained short utterances for speaker verification," *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4773–4776, 2012.

[159] H. Zeinali, L. Burget, H. Sameti, O. Glembek, and O. Plchot, "Deep neural networks and hidden Markov models in i-vector-based text-dependent speaker verification," *Proceedings of Odyssey 2016*, pp. 24–30, 2016.

[160] Y. Huang, Y. Zou, and Y. Liu, "Investigating the stacked phonetic bottleneck feature for speaker verification with short voice commands," *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 706–711, 2017.

[161] S. Dey, S. Madikeri, M. Ferras, and P. Motlicek, "Deep neural network based posteriors for text-dependent speaker verification," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5050–5054, 2016.

[162] A. Miguel, J. Llombart, A. Ortega, and E. Lleida, "Tied Hidden Factors in Neural Networks for End-to-End Speaker Recognition," *Proceedings of INTERSPEECH 2017*, pp. 2819–2823, 2017.

[163] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4052–4056, 2014.

[164] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5115–5119, 2016.

[165] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, pp. 1–13, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.specom.2015.07.003

[166] E. Malykh, S. Novoselov, and O. Kudashev, "On residual CNN in text-dependent speaker verification task," *International Conference on Speech and Computer*, pp. 593–601, 2017.

[167] F. R. rahman Chowdhury, Q. Wang, I. L. Moreno, and L. Wan, "Attention-based models for text-dependent speaker verification," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5359–5363, 2018.

[168] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4879–4883, 2018.

[169] S. Dey, S. Madikeri, and P. Motlicek, "End-to-end text-dependent speaker verification using novel distance measures," *Proceedings of INTERSPEECH 2018*, pp. 3598–3602, 2018.

[170] R. G. Leonard and G. R. Doddington, "Automatic Language Identification." TEXAS INSTRUMENTS INC DALLAS CENTRAL RESEARCH LABS, Tech. Rep., 1974.

[171] ——, "Automatic Language Identification." TEXAS INSTRUMENTS INC DALLAS CENTRAL RESEARCH LABS, Tech. Rep., 1975.

[172] ——, "Automatic Language Discrimination." TEXAS INSTRUMENTS INC DALLAS CENTRAL RESEARCH LABS, Tech. Rep., 1978.

[173] R. G. Leonard, "Language Recognition Test and Evaluation." TEXAS INSTRUMENTS INC DALLAS CENTRAL RESEARCH LABS, Tech. Rep., 1980.

[174] K. Li and T. Edwards, "Statistical models for automatic language identification," *1980 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 5, pp. 884–887, 1980.

[175] D. Cimarusti and R. Ives, "Development of an automatic identification system of spoken languages: Phase I," *1982 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 7, pp. 1661–1663, 1982.

[176] J. Foil, "Language identification using noisy speech," *1986 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 11, pp. 861–864, 1986.

[177] M. Sugiyama, "Automatic language recognition using acoustic features," *1991 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 813–816, 1991.

[178] S. Nakagawa, Y. Ueda, and T. Seino, "Speaker-independent, text-independent language identification by hmm," *Second International Conference on Spoken Language Processing*, 1992.

[179] Y. Muthusamy, K. Berkling, T. Arai, R. Cole, and E. Barnard, "A comparison of approaches to automatic language identification using telephone speech," *Third European Conference on Speech Communication and Technology*, vol. 1, pp. 1307–1310, 1993.

[180] S. Itahashi, J. X. Zhou, and K. Tanaka, "Spoken language discrimination using speech fundamental frequency," *Third International Conference on Spoken Language Processing*, 1994.

[181] E. Wong and S. Sridharan, "Methods to improve Gaussian mixture model based language identification system," *Seventh International Conference on Spoken Language Processing*, 2002.

[182] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller Jr, "Approaches to language identification using gaussian mixture models and shifted delta cepstral features," *Seventh international conference on spoken language processing*, 2002.

[183] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell, and D. A. Reynolds, "Acoustic, phonetic, and discriminative approaches to automatic language identification," *Eighth European conference on speech communication and technology*, 2003.

[184] W. M. Campbell, E. Singer, P. A. Torres-Carrasquillo, and D. A. Reynolds, "Language recognition with support vector machines," *Proceedings of Odyssey 2004*, vol. 4, p. 3, 2004.

[185] F. Castaldo, E. Dalmasso, P. Laface, D. Colibro, and C. Vair, "Language identification using acoustic models and speaker compensated cepstral-time matrices," *2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, pp. IV–1013, 2007.

[186] F. Castaldo, S. Cumani, P. Laface, and D. Colibro, "Language recognition using language factors," *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[187] N. Brümmer, A. Strasheim, V. Hubeika, P. Matějka, L. Burget, and O. Glembek, "Discriminative acoustic language recognition via channel-compensated GMM statistics," *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[188] D. Martinez, O. Plchot, L. Burget, O. Glembek, and P. Matějka, "Language recognition in ivectors space," *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[189] N. Dehak, P. A. Torres-Carrasquillo, D. Reynolds, and R. Dehak, "Language recognition via i-vectors and dimensionality reduction," *Twelfth annual conference of the international speech communication association*, 2011.

[190] M. Penagarikano, A. Varona, M. Diez, L. J. Rodriguez-Fuentes, and G. Bordel, "Study of different backends in a state-of-the-art language recognition system," *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[191] L. Ferrer, Y. Lei, M. McLaren, and N. Scheffer, "Study of senone-based deep neural network approaches for spoken language recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 105–116, 2016.

[192] M. K. Rai, M. S. Fahad, J. Yadav, K. S. Rao *et al.*, "Language identification using PLDA based on I-vector in noisy environment," *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1014–1020, 2016.

[193] Y. Song, B. Jiang, Y. Bao, S. Wei, and L.-R. Dai, "I-vector representation based on bottleneck features for language identification," *Electronics Letters*, vol. 49, no. 24, pp. 1569–1570, 2013.

[194] Y. Lei, L. Ferrer, A. Lawson, M. McLaren, and N. Scheffer, "Application of Convolutional Neural Networks to Language Identification in Noisy Conditions." *Proceedings of Odyssey 2014*, 2014.

[195] R. Fér, P. Matějka, F. Grézl, O. Plchot, and J. Černockỳ, "Multilingual bottleneck features for language recognition," *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[196] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5337–5341, 2014.

[197] 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," *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[198] Z. Tang, D. Wang, Y. Chen, L. Li, and A. Abel, "Phonetic temporal neural model for language identification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 1, pp. 134–144, 2017.

[199] S. O. Sadjadi, T. Kheyrkhah, A. Tong, C. Greenberg, D. Reynolds, E. Singer, L. Mason, and J. Hernandez-Cordero, "The 2017 NIST Language Recognition Evaluation ," *Proceedings of Odyssey 2018*, pp. 82–89, 2018.

[200] A. Lozano-Diez, O. Plchot, P. Matejka, and J. Gonzalez-Rodriguez, "Dnn based embeddings for language recognition," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5184–5188, 2018.

[201] D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, D. Povey, and S. Khudanpur, "Spoken language recognition using x-vectors," *Proceedings of Odyssey 2018*, pp. 105–111, 2018.

[202] J. Villalba, N. Brümmer, and N. Dehak, "End-to-end versus embedding neural networks for language recognition in mismatched conditions," *Proceedings of Odyssey 2018*, pp. 112–119, 2018.

[203] M. McLaren, A. Lawson, Y. Lei, and N. Scheffer, "Adaptive Gaussian backend for robust language identification." *Proceedings of INTERSPEECH 2013*, pp. 84–88, 2013.

[204] G. Gelly and J. Gauvain, "Spoken Language Identification Using LSTM-Based Angular Proximity," *Proceedings of INTERSPEECH 2017*, pp. 2566–2570, 2017.

[205] R. Duroselle, D. Jouvet, and I. Illina, "Metric learning loss functions to reduce domain mismatch in the x-vector space for language recognition," *Proceedings of INTERSPEECH 2020*, pp. 447–451, 2020.

[206] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 13 001–13 008, 2020.

[207] I. Viñals, P. Gimeno, A. Ortega, A. Miguel, and E. Lleida, "Estimation of the Number of Speakers with Variational Bayesian PLDA in the DIHARD Diarization Challenge," *Proceedings of INTERSPEECH 2018*, pp. 2803–2807, 2018.

[208] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[209] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[210] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[211] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[212] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015.

[213] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International conference on machine learning*, pp. 6105–6114, 2019.

[214] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 42–54, 2000.

[215] N. Brümmer and A. Strasheim, "Agnitio's speaker recognition system for evalita 2009," *The 11th Conference of the Italian Association for Artificial Intelligence*, 2009.

[216] N. Brümmer and J. Du Preez, "Application-independent evaluation of speaker detection," *Computer Speech & Language*, vol. 20, no. 2-3, pp. 230–275, 2006.

[217] D. A. Van Leeuwen and N. Brummer, "Channel-dependent gmm and multi-class logistic regression models for language recognition," *Proceedings of Odyssey 2006*, pp. 1–8, 2006.

[218] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification.* John Wiley & Sons, 2012.

[219] A. K. Jain, A. Ross, S. Prabhakar *et al.*, "An introduction to biometric recognition," *IEEE Transactions on circuits and systems for video technology*, vol. 14, no. 1, 2004.

[220] J. Gonzalez-Rodriguez, "Evaluating automatic speaker recognition systems: An overview of the nist speaker recognition evaluations (1996-2014)," *Loquens*, 2014.

[221] S. Z. Li, *Encyclopedia of Biometrics*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[222] R. Togneri and D. Pullella, "An overview of speaker identification: Accuracy and robustness issues," *IEEE Circuits and Systems Magazine*, vol. 11, no. 2, pp. 23–61, 2011.

[223] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *Speech Communication*, vol. 66, pp. 130–153, 2015.

[224] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[225] J. M. Bernardo and A. F. Smith, *Bayesian theory*. John Wiley & Sons, 2009, vol. 405.

[226] D. A. Van Leeuwen and N. Brümmer, "An introduction to application-independent evaluation of speaker recognition systems," *Speaker classification I*, pp. 330–353, 2007.

[227] "The NIST Year 2008 Speaker Recognition Evaluation Plan," 2008. [Online]. Available: https://www.nist.gov/sites/default/files/documents/2017/09/26/sre08$_$evalplan$_$release4.pdf

[228] "The NIST Year 2010 Speaker Recognition Evaluation Plan," 2010. [Online]. Available: https://www.nist.gov/sites/default/files/documents/itl/iad/mig/NIST$_$SRE10$_$evalplan-r6.pdf.

[229] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[230] G. Huang, M. Mattar, H. Lee, and E. G. Learned-Miller, "Learning to align from scratch," *Advances in neural information processing systems*, pp. 764–772, 2012.

[231] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[232] C. McCool, S. Marcel, A. Hadid, M. Pietikäinen, P. Matejka, J. Cernockỳ, N. Poh, J. Kittler, A. Larcher, C. Levy *et al.*, "Bi-modal person recognition on a mobile phone: using mobile phone data," *2012 IEEE International Conference on Multimedia and Expo Workshops*, pp. 635–640, 2012.

[233] H. Zeinali, L. Burget, and J. Cernocky, "A Multi Purpose and Large Scale Speech Corpus in Persian and English for Speaker and Speech Recognition: the DeepMine Database," *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 397–402, 2019.

[234] H. Zeinali, K. A. Lee, J. Alam, and L. Burget, "Short-duration Speaker Verification (SdSV) Challenge 2020: the Challenge Evaluation Plan." arXiv preprint arXiv:1912.06311, Tech. Rep., 2020.

[235] "The 2009 NIST language recognition evaluation plan, 2009." [Online]. Available: http://www.itl.nist.gov/iad/mig/tests/lre/2009.

[236] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "Calibration Approaches for Language Detection." *Proceedings of INTERSPEECH 2017*, pp. 2804–2808, 2017.

[237] "The 2015 NIST language recognition evaluation plan, 2015." [Online]. Available: https://www.nist.gov/itl/iad/mig/upload/LRE15$_$Eval$_$Plan$_$v23.pdf.

[238] "NIST 2017 Language recognition evaluation plan, 2017." [Online]. Available: https://www.nist.gov/sites/default/files/documents/2017/09/29/lre17$_$eval$_$plan-2017-09-29$_$v1.pdf.

[239] E. Lleida, A. Ortega, A. Miguel, V. Bazán-Gil, C. Pérez, M. Gómez, and A. de Prada, "Albayzin 2018 evaluation: the iberspeech-RTVE challenge on speech technologies for spanish broadcast media," *Applied Sciences*, vol. 9, no. 24, p. 5412, 2019.

[240] E. Lleida, A. Ortega, A. Miguel, V. Bazán, C. Pérez, M. Gómez, and A. de Prada, "Albayzin evaluation: IberSPEECH-RTVE 2020 Multimodal Diarization and Scene Description Challenge," Tech. Rep., 2020.

[241] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[242] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *Thirty-first AAAI conference on artificial intelligence*, 2017.

[243] V. Mingote Bueno and A. Miguel Artiaga, "Estudio de técnicas de aprendizaje automático basado en redes neuronales para reconocimiento biométrico de personas." Master's thesis, Universidad de Zaragoza, 2016.

[244] S. Marcel, C. McCool, P. Matejka, T. Ahonen, and J. Cernocky, "Mobile Biometry (MOBIO) Face and Speaker Verification Evaluation," Idiap, Tech. Rep., 2010.

[245] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental," *Pattern recognition and artificial intelligence*, vol. 116, pp. 374–388, 1976.

[246] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[247] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International conference on machine learning*, pp. 448–456, 2015.

[248] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of ICLR*, pp. 1–15, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[249] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *NIPS-W*, 2017.

[250] T. Stafylakis, P. Kenny, P. Ouellet, J. Perez, M. Kockmann, and P. Dumouchel, "I-Vector/PLDA variants for text-dependent speaker recognition," *Centre de Recherche Informatique de Montreal (CRIM)*, 2013.

[251] P. Kenny, T. Stafylakis, P. Ouellet, and M. J. Alam, "JFA-based front ends for speaker recognition," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1705–1709, 2014.

[252] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[253] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *Journal of Animal and Plant Sciences*, vol. 27, no. 3, pp. 797–802, 2015.

[254] A. Korattikara, V. Rathod, K. Murphy, and M. Welling, "Bayesian dark knowledge," *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 3438–3446, 2015.

[255] B. B. Sau and V. N. Balasubramanian, "Deep model compression: Distilling knowledge from noisy teachers," *arXiv preprint arXiv:1610.09650*, 2016.

[256] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria," *Proceedings of INTERSPEECH 2014*, pp. 1910–1914, 2014.

[257] Y. Chebotar and A. Waters, "Distilling Knowledge from Ensembles of Neural Networks for Speech Recognition." *Proceedings of INTERSPEECH 2016*, pp. 3439–3443, 2016.

[258] M. Huang, Y. You, Z. Chen, Y. Qian, and K. Yu, "Knowledge Distillation for Sequence Model." *Proceedings of INTERSPEECH 2018*, pp. 3703–3707, 2018.

[259] P. Shen, X. Lu, S. Li, and H. Kawai, "Feature Representation of Short Utterances Based on Knowledge Distillation for Spoken Language Identification." *Proceedings of INTERSPEECH 2018*, pp. 1813–1817, 2018.

[260] ——, "Interactive Learning of Teacher-student Model for Short Utterance Spoken Language Identification," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5981–5985, 2019.

[261] J. Li, R. Zhao, Z. Chen, C. Liu, X. Xiao, G. Ye, and Y. Gong, "Developing far-field speaker system via teacher-student learning," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5699–5703, 2018.

[262] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[263] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in Neural Information Processing Systems (NIPS) 27*, pp. 2654–2662, 2014.

[264] S. Wang, Y. Yang, T. Wang, Y. Qian, and K. Yu, "Knowledge Distillation for Small Foot-print Deep Speaker Embedding," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6021–6025, 2019.

[265] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7893–7897, 2013.

[266] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," *Proceedings of ICML*, pp. 681–688, 2011.

[267] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, pp. 577–585, 2015.

[268] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[269] I. Viñals, D. Ribas, V. Mingote, J. Llombart, P. Gimeno, A. Miguel, A. Ortega, and E. Lleida, "Phonetically-Aware Embeddings, Wide Residual Networks with Time-Delay Neural Networks and Self Attention Models for the 2018 NIST Speaker Recognition Evaluation," *Proceedings of INTERSPEECH 2019*, pp. 4310–4314, 2019.

[270] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

[271] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *Proceedings of ICLR 2021*, 2021.

[272] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *International Conference on Machine Learning*, pp. 10 347–10 357, 2021.

[273] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-Centric Learning with Slot Attention," *NeurIPS 2020*, vol. 33, pp. 11 525–11 538, 2020.

[274] G. Lample, A. Sablayrolles, M. Ranzato, L. Denoyer, and H. Jégou, "Large memory layers with product keys," *Advances in Neural Information Processing Systems*, pp. 8546–8557, 2019.

[275] T. Zhou, Y. Zhao, J. Li, Y. Gong, and J. Wu, "CNN with phonetic attention for text-independent speaker verification," *2019 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 718–725, 2019.

[276] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *preprint arXiv:1410.5401*, 2014.

[277] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[278] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10 781–10 790, 2020.

[279] W. Cai, Z. Cai, X. Zhang, X. Wang, and M. Li, "A novel learnable dictionary encoding layer for end-to-end language identification," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5189–5193, 2018.

[280] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[281] H. Zeinali, K. A. Lee, J. Alam, and L. Burget, "SdSV Challenge 2020: Large-Scale Evaluation of Short-Duration Speaker Verification." *Proceedings of INTERSPEECH 2020*, pp. 731–735, 2020.

[282] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A Large-Scale Speaker Identification Dataset," *Proceedings of INTERSPEECH 2017*, pp. 2616–2620, 2017.

[283] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *NIPS 2014 Deep Learning Workshop*, pp. 1–9, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[284] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," *International Conference on Machine Learning*, pp. 1613–1622, 2015.

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

[286] Y. Srivastava, V. Murali, and S. R. Dubey, "A performance evaluation of loss functions for deep face recognition," *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics*, pp. 322–332, 2019.

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

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

[289] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9370, no. 2010, pp. 84–92, 2015.

[290] L. P. Garcia-Perera, J. A. Nolazco-Flores, B. Raj, and R. Stern, "Optimization of the DET curve in speaker verification," *2012 IEEE Spoken Language Technology Workshop (SLT)*, pp. 318–323, 2012.

[291] K.-A. Toh, J. Kim, and S. Lee, "Maximizing area under ROC curve for biometric scores fusion," *Pattern Recognition*, vol. 41, no. 11, pp. 3373–3392, 2008.

[292] A. Herschtal and B. Raskutti, "Optimising area under the roc curve using gradient descent," *Proceedings of the twenty-first international conference on Machine learning*, p. 49, 2004.

[293] A. Soriano, L. Vergara, B. Ahmed, and A. Salazar, "Fusion of scores in a detection context based on Alpha integration," *Neural Computation*, vol. 27, no. 9, pp. 1983–2010, 2015.

[294] S. Novoselov, V. Shchemelinin, A. Shulipa, A. Kozlov, and I. Kremnev, "Triplet Loss Based Cosine Similarity Metric Learning for Text-independent Speaker Recognition," *Proceedings of INTERSPEECH 2018*, pp. 2242–2246, 2018.

[295] M. McLaren, D. Castan, M. Nandwana, L. Ferrer, and E. Yilmaz, "How to train your speaker embeddings extractor," *Proceedings of Odyssey 2018*, pp. 327–334, 2018.

[296] K. Lee, H. Li, L. Deng, V. Hautamäki, W. Rao, X. Xiao, A. Larcher, H. Sun, T. Nguyen, G. Wang *et al.*, "The 2015 NIST language recognition evaluation: the shared view of I2R, Fantastic4 and SingaMS," *Proceedings of INTERSPEECH 2016*, pp. 3211–3215, 2016.

[297] O. Plchot, P. Matejka, O. Glembek, R. Fer, O. Novotny, J. Pesan, L. Burget, N. Brummer, and S. Cumani, "BAT System Description for NIST LRE 2015," *Proceedings of Odyssey 2016*, pp. 166–173, 2016.

[298] A. Martin and M. Przybocki, "The NIST 1999 speaker recognition evaluation—An overview," *Digital signal processing*, vol. 10, no. 1-3, pp. 1–18, 2000.

[299] S. Bengio and J. Mariéthoz, "The expected performance curve: a new assessment measure for person authentication," IDIAP, Tech. Rep., 2003.

[300] I. Kukanov, T. N. Trong, V. Hautamäki, S. M. Siniscalchi, V. M. Salerno, and K. A. Lee, "Maximal Figure-of-Merit Framework to Detect Multi-Label Phonetic Features for Spoken Language Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 682–695, 2020.

[301] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained Softmax Loss for Discriminative Face Verification," 2017. [Online]. Available: http://arxiv.org/abs/1703.09507

[302] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "Normface: L2 hypersphere embedding for face verification," *Proceedings of the 25th ACM international conference on Multimedia*, pp. 1041–1049, 2017.

[303] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Proceedings of the 30th International Conference on Neural Information Processing Systems*, vol. 29, pp. 901–909, 2016.

[304] J. Poignant, H. Bredin, and C. Barras, "Multimodal person discovery in broadcast tv at mediaeval 2015," *MediaEval 2015 working notes proceedings*, 2015.

[305] H. Bredin, C. Barras, and C. Guinaudeau, "Multimodal person discovery in broadcast TV at MediaEval 2016," *MediaEval 2016 working notes proceedings*, 2016.

[306] O. Sadjadi, C. Greenberg, E. Singer, D. Reynolds, L. Mason, and J. Hernandez-Cordero, "The 2019 NIST Audio-Visual Speaker Recognition Evaluation," *Proceedings of Odyssey 2020*, pp. 259–265, 2020.

[307] R. K. Das, R. Tao, J. Yang, W. Rao, C. Yu, and H. Li, "HLT-NUS Submission for NIST 2019 Multimedia Speaker Recognition Evaluation," *arXiv preprint arXiv:2010.03905*, 2020.

[308] D. Garcia-Romero, D. Snyder, G. Sell, D. Povey, and A. McCree, "Speaker diarization using deep neural network embeddings," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4930–4934, 2017.

[309] I. Viñals, P. Gimeno, A. Ortega, A. Miguel, and E. Lleida, "In-domain Adaptation Solutions for the RTVE 2018 Diarization Challenge," *Proceedings of IberSPEECH 2018*, pp. 220–223, 2018.

[310] E. Khoury, P. Gay, and J.-M. Odobez, "Fusing matching and biometric similarity measures for face diarization in video," *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pp. 97–104, 2013.

[311] N. Le, A. Heili, D. Wu, and J.-M. Odobez, "Efficient and Accurate Tracking for Face Diarization via Periodical Detection," *International Conference on Pattern Recognition*, no. CONF, 2016.

[312] A. Ortega, A. Miguel, E. Lleida, V. Bazán, C. Pérez, M. Gómez, and A. de Prada, "Albayzin evaluation: IberSPEECH-RTVE 2020 Speaker Diarization and Identity Assignment," Tech. Rep., 2020.

[313] J. Shi and J. Malik, "Normalized cuts and image segmentation," Tech. Rep., 2000.

[314] E. Ramos-Muguerza, L. Docío-Fernández, and J. L. Alba-Castro, "The GTM-UVIGO System for Audiovisual Diarization," *Proceedings of IberSPEECH 2018*, pp. 204–207, 2018.

[315] I. Viñals, A. Ortega, A. Miguel, and E. Lleida, "The Domain Mismatch Problem in the Broadcast Speaker Attribution Task," *Applied Sciences*, vol. 11, no. 18, p. 8521, 2021.

[316] P. Gimeno, D. Ribas, A. Ortega, A. Miguel, and E. Lleida, "Convolutional recurrent neural networks for speech activity detection in naturalistic audio from apollo missions," *Proceedings of IberSPEECH 2021*, pp. 26–30, 2021.

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

[318] M. J. Alam, P. Ouellet, P. Kenny, and D. O'Shaughnessy, "Comparative evaluation of feature normalization techniques for speaker verification," *International Conference on Nonlinear Speech Processing*, pp. 246–253, 2011.

[319] S. Chen, P. Gopalakrishnan *et al.*, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," *Proceedings of DARPA broadcast news transcription and understanding workshop*, vol. 8, pp. 127–132, 1998.

[320] J. Villalba, N. Chen, D. Snyder, D. Garcia-Romero, A. McCree, G. Sell, J. Borgstrom, F. Richardson, S. Shon, F. Grondin *et al.*, "State-of-the-Art Speaker Recognition for Telephone and Video Speech: The JHU-MIT Submission for NIST SRE18." *Proceedings of INTERSPEECH 2019*, pp. 1488–1492, 2019.

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

[322] P. Bell, M. J. Gales, T. Hain, J. Kilgour, P. Lanchantin, X. Liu, A. McParland, S. Renals, O. Saz, M. Wester *et al.*, "The mgb challenge: Evaluating multi-genre broadcast media recognition," *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 687–693, 2015.

[323] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," *Twelfth annual conference of the international speech communication association*, 2011.

[324] I. Viñals, P. Gimeno, A. Ortega, A. Miguel, and E. Lleida, "ViVoLAB Speaker Diarization System for the DIHARD 2019 Challenge." *Proceedings of INTERSPEECH 2019*, pp. 988–992, 2019.

[325] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[326] M. Porta-Lorenzo, J. L. Alba-Castro, and L. Docío-Fernández, "The GTM-UVIGO System for Audiovisual Diarization 2020," *Proceedings of IberSPEECH 2021*, pp. 81–85, 2021.

[327] C. Luna-Jiménez, R. Kleinlein, F. Fernández-Martınez, J. Manuel, and J. M. M.-F. Pardo-Munoz, "GTH-UPM System for Albayzin Multimodal Diarization Challenge 2020," *Proceedings of IberSPEECH 2021*, pp. 71–75, 2021.

[328] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *International Conference on Machine Learning*, pp. 1321–1330, 2017.