UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Deep Neural Networks for Channel Compensated i-Vectors in Speaker Recognition

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Albert Jiménez Sanfiz

In partial fulfilment

of the requirements for the degree in

Ciències i Tecnologies de les Telecomunicacions

Advisors:

Javier Hernando Pericas

Omid Ghahabi Esfahani

Barcelona, June 2014

Abstract

This thesis explores the application of channel-compensation techniques in speaker verification and the posterior combination with deep learning technologies. The idea is to reduce the degradation of the performance due to mismatched environments when training and testing the system as well as increasing the accuracy and reliability of the speaker verification systems.

To achieve the goals, state-of-the-art techniques such as i-vector modeling, PLDA and DNNs will be applied. In this thesis we propose channel-compensated i-vectors that are extracted using the PLDA technique called Beta vectors. We apply deep learning using a hybrid DBN-DNN architecture with these Beta vectors as an input.

At the end, with the Beta vector proposal and scoring with the cosine metric we obtain a relative improvement of 21.4% and 21% in the EER and minDCF with respect the raw i-vectors. If we change the classifier to the DNN the relative improvement increases to 32.3% and 32.1%, respectively. Our Beta-DNN outperforms the i-vector-DNN baseline system with 18.9% and 25% relative improvement in ERR and minDCF.

Resum

Aquesta tesis explora l’aplicació de tècniques de compensació de canal a l’àmbit de verificació de parlant i la seva combinació posterior amb *deep learning*. La idea és reduir la degradació del funcionament deguda a que els entrenaments i els tests produeixen en diferents ambients i alhora incrementar la precisió i fiabilitat dels sistemes de verificació de parlant.

Per aconseguir els objectius aplicarem tècniques punteres com per exemple modelat amb *i-vectors*, *PLDA*, o *DNNs*. A aquesta tesis proposem uns i-vectors amb compensació de canal anomenats *Beta vectors* que són extrets utilitzant la tècnica del *PLDA*. Aplicarem *deep learning* amb una arquitectura híbrida *DBN-DNN* que tindrà com a entrada els *Beta vectors* proposats.

Al final, amb la proposta dels *Beta vectors* i utilitzant la distància de cosinus com a mètrica obtenim una millora relativa de 21.4% i 21% en el *EER* i el *minDCF* amb respecte de els i-vectors sense processar. Si canviem el classificador i apliquem la *DNN* proposada la millora relativa incrementa fins a 32.3% and 32.1% respectivament. Si comparem el nostre sistema *Beta-DNN* amb el sistema *i-vector-DNN* de referència veiem que el superem amb una millora de 18.9% en *EER* i un 25% en *minDCF*.

Resumen

Esta tesis explora la aplicación de técnicas de compensación de canal en el ámbito de verificación del hablante i su combinación posterior con *deep learning*. La idea es reducir la degradación del funcionamiento debida a que el entrenamiento y los test se realizan en diferentes ambientes y a la vez aumentar la precisión y fiabilidad de los sistemas de verificación del hablante.

Para conseguir los objetivos utilizaremos técnicas punteras como por ejemplo modelado con *i-vectors*, *PLDA* o *DNNs*. En esta tesis proponemos unos *i-vectors* con compensación de canal llamados *Beta vectors* que son extraídos utilizando la técnica del *PLDA*. Aplicaremos *deep learning* con una arquitectura híbrida *DBN-DNN* que tendrá como entrada los *Beta vectors* propuestos.

Al final, con la propuesta de los *Beta vectors* y utilizando la distancia de coseno como métrica obtenemos una mejora relativa de 21.4% i 21% en el *EER* i el *minDCF* con respecto a los *i-vectors* sin procesar. Si cambiamos el clasificador y aplicamos la *DNN* propuesta, la mejora relativa incrementa hasta un 32.3% y un 32.1% respectivamente. Si comparamos nuestro sistema *Beta-DNN* com el sistema *i-vector-DNN* de referencia vemos que lo superamos con una mejora de 18.9% en el *EER* y un 25% en el *minDCF*.

“El único lugar donde el éxito viene antes que trabajo es en el diccionario”

Acknowledgements

First I want to express my gratitude to my advisor, Javier. Thank you for choosing me to develop the project and for your guidance and patience during all this time. Thank you for teaching me some lessons not just academic but in life. I hope we can go running someday. I would also want to thank my other advisor, Omid, without him none of this could have been possible, I appreciate very much your support and help. Javier was right, I think that in the end I have found a دوست (friend) in you. Finally I would like to give thanks to Carlos, who helped me during my first experiences with the servers.

Desde que empecé esta etapa de mi vida en la universidad nada ha sido fácil. Ha habido muchos días de estudio, nervios pre y post-examen y mucho estrés. No quiero imaginar como hubiera sido este proceso sin el grupo de gente que conocí en primero y que me ha acompañado durante toda la carrera. Gracias Drop1s (Adrià, Albert, Chema, Ferran José y Víctor), por ser mis compañeros de fatigas y por todo vuestro apoyo que han hecho más llevaderos estos 4 años.

No quiero olvidarme de todos mis otros amigos, que aunque la vida nos separe, siempre han estado allí cuando los he necesitado. Estéis donde estéis, gracias por haber aparecido en mi camino, vuestros consejos, momentos y sobre todo por hacer de esta vida una experiencia inolvidable.

Y por último pero no menos importante, un agradecimiento sincero a toda mi familia. Gracias a mis padres Rafa y Nuria, por su apoyo incondicional y por aguantarme en los mis momentos de locura. A mi hermana Sandra por comprenderme y conseguir hacerme reír en todo momento. A mis dos abuelas, por cuidarme desde que era pequeño y darme su cariño siempre.

Muchas gracias a todos.

Revision history and approval record

Revision	Date	Purpose
0	14/06/2015	Document creation
1	02/06/2015	Document revision
2	09/07/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Albert Jiménez Sanfiz	albertjimenezsanfiz@gmail.com
Javier Hernando Pericas	javier.hernando@upc.edu
Omid Ghahabi Esfahani	omid.ghahabi@upc.edu

Written by:		Reviewed and approved by:	
Date	14/06/2015	Date	09/07/2015
Name	Albert Jiménez	Name	Javier Hernando, Omid Ghahabi
Position	Project Author	Position	Project Supervisors

Table of Contents

List of Figures

List of Tables:

1. Introduction

1.1. Motivation and Applications

Numerous measurements and signals have been proposed and investigated for use in biometric recognition systems. Among the most popular measurements are fingerprint, face, and voice. While each has pros and cons relative to accuracy and deployment, there are two main factors that have made voice a compelling biometric. First, speech is a natural signal to produce that is not considered threatening by users to provide. In many applications, speech may be the main (or only, e.g., telephone transactions) modality, so users do not consider providing a speech sample for authentication as a separate or intrusive step. Second, the telephone system provides a ubiquitous, familiar network of sensors for obtaining and delivering the speech signal.

The applications in which this technology can be applied cover almost all the areas where it is desirable to secure actions, transactions, or any type of interactions by identifying or authenticating the person making the transaction. Regardless of forensic applications (police, judicial and legal use), there are four areas where speaker verification can be used: access control to facilities, secured transactions, structuring audio information and games. Its low implementation cost and the acceptability by the end users is giving speech authentication more popularity these days.

Most state-of the-art speaker verification systems perform well in controlled environments where data is collected from reasonably clean environments. However, acoustic mismatch due to different training and testing environments can severely deteriorate the performance of the speaker verification systems. Degradation of performance due to mismatched environments has been a barrier for deployment of speaker recognition technologies.

Having seen the importance and applications of speaker recognition technologies and their drawbacks, in this project we aim to apply state-of-the-art techniques to compensate that channel effect and to classify the voice with the objective of increasing the accuracy and reliability of those systems.

1.2. Project Overview and Goals

The project is carried out at the department of Signal Theory and Communications in the Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona (ETSETB).

In the scenario of speaker recognition we can distinguish between three tasks: segmentation and clustering, identification and verification. This project is focused on the technologies behind the verification task. The objective of these systems is assuring that the speaker who is talking is the same as the one he claims to be.

This project takes as a baseline the work of the PhD candidate Omid Ghahabi in the ambit of speaker verification where he applies deep learning for speaker verification [1] [2] [3] using Deep Neural Networks (DNNs) and modeling the speech audio signal using i-vectors. In order to outperform that baseline system, we will apply channel compensation techniques at feature and i-vector levels and we will try to find a combination that gives us suitable data for training the DNN. The project goals can be described as:

1. Apply Channel compensation after the feature extraction part. Check the performance at feature level and at i-vector level.
2. Apply Channel compensation at the i-vectors level. We will apply normalization to the raw i-vectors and the i-vectors obtained from the normalized feature vectors and we will study if there is an improvement that leads to combine them.
3. Find suitable data as an input of the DNN among the previous experiments. Train, tune and test the DNN system.

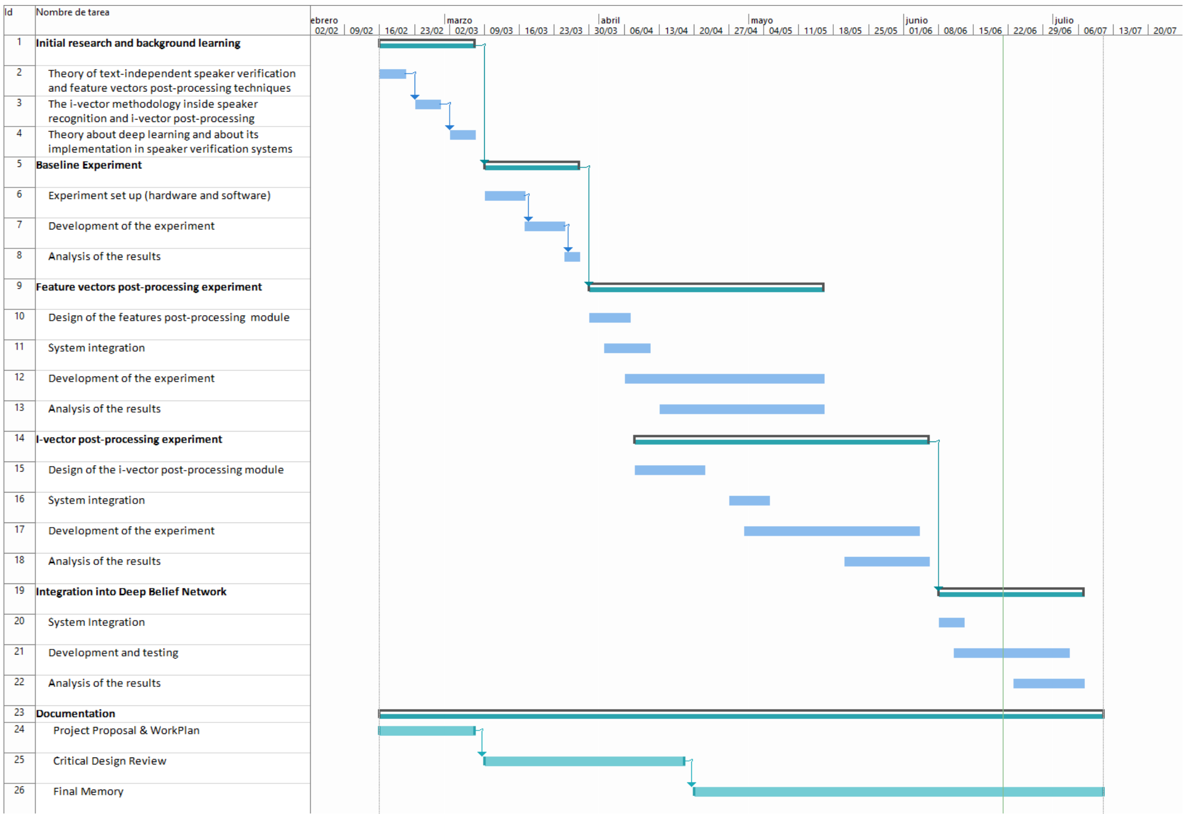
1.3. Work Plan

Incidences

In general the project has been developed as expected, there were some problems with the servers at the beginning but they were solved quickly. Due to length of processing time that spent some of the parts more things were done in parallel with respect to the first Proposal Plan as it is stated in the updated Gantt Diagram.

The work packages and the milestones can be found at the appendix.

Gantt Diagram



Id	Nombre de tarea
1	**Initial research and background learning**
2	Theory of text-independent speaker verification and feature vectors post-processing techniques
3	The i-vector methodology inside speaker recognition and i-vector post-processing
4	Theory about deep learning and about its implementation in speaker verification systems
5	**Baseline Experiment**
6	Experiment set up (hardware and software)
7	Development of the experiment
8	Analysis of the results
9	**Feature vectors post-processing experiment**
10	Design of the features post-processing module
11	System integration
12	Development of the experiment
13	Analysis of the results
14	**I-vector post-processing experiment**
15	Design of the i-vector post-processing module
16	System integration
17	Development of the experiment
18	Analysis of the results
19	**Integration into Deep Belief Network**
20	System Integration
21	Development and testing
22	Analysis of the results
23	**Documentation**
24	Project Proposal & WorkPlan
25	Critical Design Review
26	Final Memory

Fig. 1.1: Gantt Diagram

1.4. Thesis Outline

This thesis will be structured as follows:

Introduction. Includes a general description of the project, the motivation, its objectives, the structure and the work plan carried out.

State of the Art. This part contains a review of the related work relevant to the thesis.

Project Development. Throughout this chapter the reader can find the theoretical framework behind the experiments done.

Experimental Part. This part contains the description of the experimental set up and all the experiments that have been carried out with the final results explained in detail.

Budget. This is the economic part of the project; here an estimation of the project cost will be done.

Conclusions and Future Development. This part concludes the thesis with the final commentaries as well as it opens a way for future work in the same topic.

2. State of the art

2.1. Text-independent Speaker Verification Systems

In the world of speaker verification we can make a distinction between text-independent/dependent systems [2]. Text-dependent systems are used in applications based on scenarios with cooperative users. It implies fixed digit string passwords or repeating prompted phrases from a small vocabulary. Such constraints are quite reasonable and can greatly improve the accuracy of a system. A text-independent system provides a more flexible recognition system able to operate without explicit user cooperation and independent of the spoken utterance.

A speaker verification system is composed of two distinct phases, a training phase and a test phase. Each of them can be seen as a succession of independent modules.

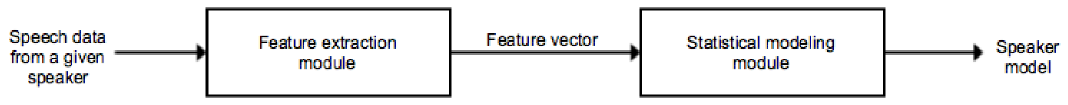

Speech data from a given speaker
Feature extraction module
Feature vector
Statistical modeling module
Speaker model

Fig. 2.1: Module representation of the training phase of a speaker verification system

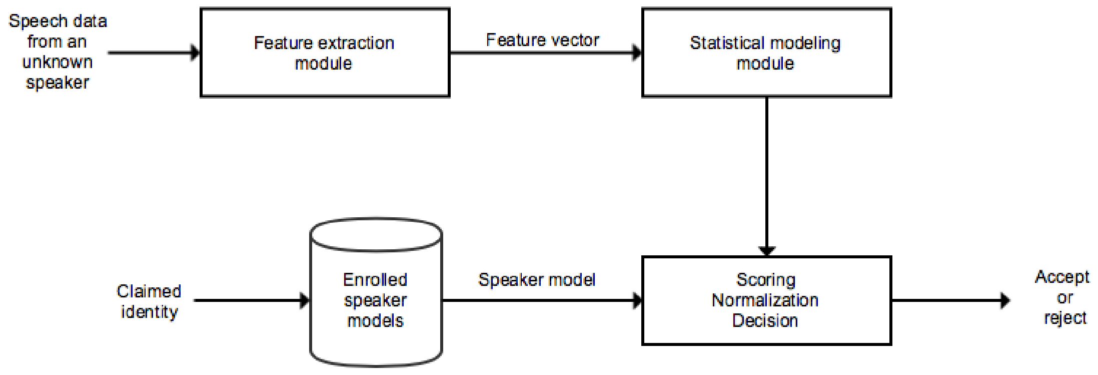

Speech data from an unknown speaker
Feature extraction module
Feature vector
Statistical modeling module
Claimed identity
Enrolled speaker models
Speaker model
Scoring Normalization Decision
Accept or reject

Fig. 2.2: Module representation of the test phase of a speaker verification system

Fig. 2.1 shows a modular representation of the training phase of a speaker verification system. The first step consists in extracting parameters from the speech signal to obtain a representation suitable for statistical modeling. The second step consists in obtaining a statistical model from the parameters.

Fig. 2.2 shows a modular representation of the test phase of a speaker verification system. The entries of the system are a claimed identity and the speech samples pronounced by an unknown speaker. First, speech parameters are extracted from the speech signal using exactly the same module as for the training phase. Then, the speaker model corresponding to the claimed identity is extracted from the set of statistical models calculated during the training phase. Finally, the last module computes some scores, normalizes them, and makes an acceptance or a rejection decision.

2.2. Feature Extraction

Feature extraction consists in transforming the speech signal to a set of feature vectors. The aim of this transformation is to obtain a new representation, which is more compact, less redundant, and more suitable for statistical modeling and the calculation of a distance or any other kind of score. Most of the speech parameterizations used in speaker verification systems relies on a cepstral representation of speech. Two cepstral representations have been proposed: Filterbank-based cepstral parameters (Fig. 2.3) and LPC-based cepstral parameters (Fig. 2.4). Both approaches are explained in [4].

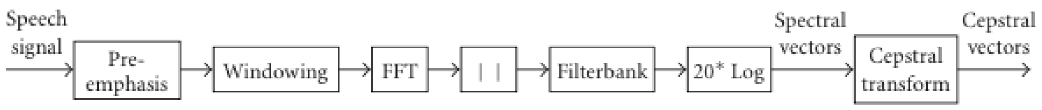


Fig. 2.3: Modular representation of a filterbank-based cepstral parameterization [4]

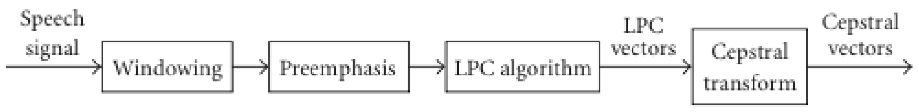


Fig. 2.4: Modular representation of an LPC-based cepstral parameterization [4]

After the cepstral coefficients have been calculated, we also incorporate in the vectors some dynamic information, that is, some information about the way these vectors vary in time. This is classically done by using the Δ and $\Delta\Delta$ parameters, which are polynomial approximations of the first and second derivatives [5]. At this step, one can choose whether to incorporate the log energy and the ∆ log energy in the feature vectors or not. In practice, the former one is often discarded and the latter one is kept.

Once all the feature vectors have been computed, in order to achieve a better performance in recognition, the last step that is done is keeping the vectors corresponding to speech portions of the signal and removing those corresponding to silence or background noise [4].

2.3. Feature Normalization

Feature normalization strategies are employed in speaker recognition systems to compensate for the effects of environmental mismatch. These techniques are preferred because a priori knowledge and adaptation are not required under any environment. Most of the normalization techniques are applied as a post-processing scheme on the Mel-frequency cepstral coefficient (MFCC) speech features.

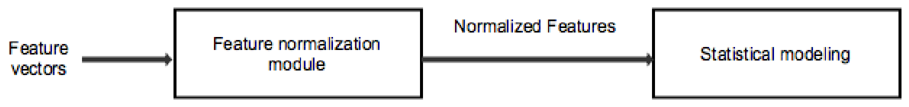


Fig 2.5: Module representation of the feature normalization stage

Normalization techniques can be classified as model-based or data distribution-based techniques. In model-based normalization techniques, certain statistical properties of speech such as mean, variance, moments, are normalized to reduce the residual mismatch in feature vectors. Data distribution-based techniques aim at normalizing the feature distribution towards a target distribution.

Several techniques have been proposed such as Mean and Variance Normalization (MVN) [6], feature warping [7], RelAtive SpecTrA (RASTA) [8], Short Time Gaussianization (STG) [9]. In this thesis we will apply and analyze the contribution in different stages of the system of including the techniques of MVN (model-based), feature warping (distribution-based) and a combination of both.

MVN

MVN is performed over the whole utterance with the assumption that the channel effect is constant over the entire utterance [6]. It includes Cepstral Mean Substraction (CMS) and variance normalization. Being x_{raw} the raw feature vector and x_{norm} the processed one:

$$x_{norm} = \frac{x_{raw} - \overline{x_{raw}}}{\sigma_{x_{raw}}} \quad (2.1)$$

The motivation for CMS is to remove from the cepstrum the contribution of slowly varying convolutive noises and the objective of the variance normalization is to decrease the range of values that the feature vectors can take as we aim to have normalized feature vectors with a Gaussian distribution and unit variance.

Feature Warping

The aim of feature warping is to construct a more robust representation of each cepstral feature distribution. This is achieved by conditioning and conforming the individual cepstral feature streams such that they follow a specific target distribution over a window of speech frames [7].

Once we have the set of cepstral coefficients, the process of warping begins by analyzing them independently as a separate feature stream over time for use in the warping process. A window of features is extracted from the feature stream and processed in the warping algorithm to determine a mapped feature for the initial cepstral feature in the middle of the window. A single frame shifts the sliding window each time and the analysis is repeated.

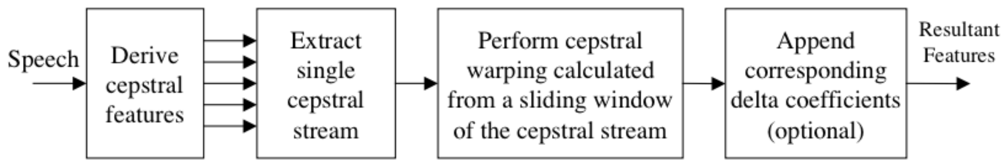

Speech
Derive cepstral features
Extract single cepstral stream
Perform cepstral warping calculated from a sliding window of the cepstral stream
Append corresponding delta coefficients (optional)
Resultant Features

Fig. 2.6: Block diagram of the feature warping process. [7]

For speech, the true distribution of a feature is speaker dependent and multi-modal in nature. However, various channel and additive noise influences can corrupt this distribution. We aim to perform a mapping that will condition the feature distribution. To simplify the mapping we assume that the target speaker features conform to a particular distribution type. Intuitively, this method compensates in part for the linear channel in that the short-term mean is removed, and attempts to conform the distributive shape and spread to limit additive noise effects.

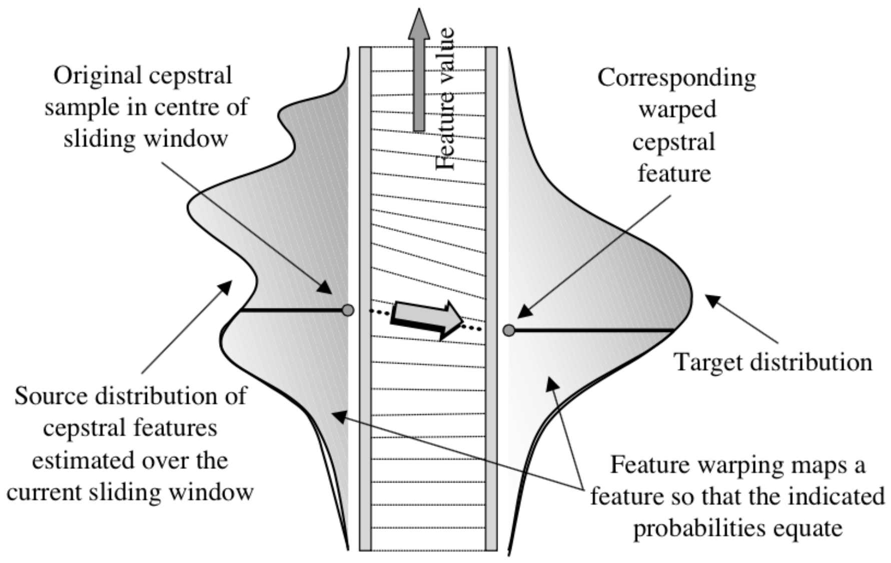

Original cepstral sample in centre of sliding window
Feature value
Corresponding warped cepstral feature
Target distribution
Source distribution of cepstral features estimated over the current sliding window
Feature warping maps a feature so that the indicated probabilities equate

Fig. 2.7: Warping of features according to a target distribution shape. [7]

2.4. Statistical Modeling

Once we have all the feature vectors, the next step is carrying out a statistical modeling of them to find an approximation of their distribution. In speaker verification a lot of models have been used and proposed. The ones that have been applied in this thesis will be stated below:

Gaussian Mixture Model (GMM)

GMMs are a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. It applies the Expectation Maximization (EM) algorithm to estimate the maximum likelihood model parameters. The most successful implementation [10] uses a Universal Background Model (UBM) to represent the speaker-independent distribution of features and then performs adaptation to train the target models. The scoring is carried out computing a log-likelihood ratio test.

i-Vectors

They are based on the JFA framework [11] were the speaker and channel factors consist in defining two distinct spaces: the speaker space and the channel space. In i-vectors we only define a single space [12]. This new space, which is referred to as total variability space contains the speaker and channel variabilities that appear in training utterances simultaneously. It is defined by the total variability matrix $\mathbf{T}$, which contains the eigenvectors with the largest eigenvalues of the total variability covariance matrix. Given the centralized Baum-Welch statistics from all available speech utterances, the low rank T is trained in an iterative process. The training process assumes that an utterance can be represented by the GMM mean supervector,

$$\mathbf{M} = \boldsymbol{\mu} + \mathbf{Tw} \tag{2.2}$$

where $\boldsymbol{\mu}$ is the speaker and session independent mean supervector from the UBM, and $\mathbf{w}$ is a low rank vector referred to as the identity vector or i-vector. The supervector **M** is assumed to be normally distributed with mean $\boldsymbol{\mu}$ and covariance $\mathbf{TT}^{\mathrm{T}}$, and the i-vectors have a standard normal distribution $N(0,1)$. Furthermore, in [12] cosine distance is proposed as a successful metric to make the scoring between the target and test i-vectors and some channel-compensation techniques are suggested. The first one is Linear Discriminant Analysis (LDA) and the second one is Within Class Covariance Normalization (WCCN).

WCCN

The idea behind it is to minimize the expected error rate of false acceptances and false rejections during the training step. The WCCN algorithm uses the within-class covariance matrix to normalize the cosine kernel functions in order to compensate for intersession variability, while guaranteeing conservation of directions in space in contrast with LDA [12].

We assume that all utterances of a given speaker belong to one class. The within class covariance matrix is computed as follows:

$$W = \frac{1}{S}\sum_{\boldsymbol{s=1}}^{\boldsymbol{S}}\frac{1}{n_s}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n_s}}(\boldsymbol{w_i} - \overline{\boldsymbol{w}}_{\boldsymbol{s}})(\boldsymbol{w_i} - \overline{\boldsymbol{w}}_{\boldsymbol{s}})^t \tag{2.3}$$

where $\overline{\boldsymbol{w}_s} = \frac{\mathbf{1}}{n_s}\sum_{\boldsymbol{i}=\mathbf{1}}^{n_s}(\boldsymbol{w_i})$ is the mean of i-vectors for each speaker, S is the number total of speakers and n_s is the number of utterances per speaker. In order to preserve the inner-product form of the cosine kernel, a feature-mapping function can be defined as follows:

$$\varphi(\boldsymbol{w}) = \boldsymbol{B}^t\boldsymbol{w} \tag{2.4}$$

$$\boldsymbol{w_{norm}} = \boldsymbol{B^t}\, \boldsymbol{w_{raw}} \tag{2.5}$$

where $\boldsymbol{B}$ is obtained through Cholesky decomposition of matrix $\boldsymbol{W}^{-1} = \boldsymbol{BB}^t$.

Probabilistic Linear Discriminant Analysis (PLDA)

PLDA is a probabilistic generative model that can accomplish a wide variety of recognition tasks. In our case, it carries out the modeling of the speaker and session variability [13] [14] [15] [16] [17]. This model will be explained with detail in section 3, as it has been very important during the thesis development.

Deep Learning

Deep learning refers to a rather wide class of machine learning techniques and architectures, with the hallmark of using many layers of non-linear information processing that are hierarchical in nature. Their power relies in that they can model complex non-linear relationships. According to [18] we can classify the deep learning architectures and techniques depending on their final function. We have three categories:

- *Deep networks for unsupervised or generative learning*, which are intended to capture high order correlation of the observed or visible data for pattern analysis or synthesis purposes when no information about target class labels is available.
- *Deep networks for supervised learning*, which are intended to directly provide discriminative power for pattern classification purposes, often by characterizing the posterior distributions of classes conditioned on the visible data. Target label data are always available in direct or indirect forms for such supervised learning.
- *Hybrid deep networks*, where the goal is discrimination. The network is assisted, often in a significant way, with the outcomes of generative or unsupervised deep networks.

2.5. Evaluation

In a speaker verification system there two types of error can occur: false rejection and false acceptance. A false rejection (or non-detection) error happens when a valid identity claim is rejected. A false acceptance (or false alarm) error consists in accepting an identity claim from an impostor. Both types of error depend on the threshold θ used in the decision making process [4].

The performance of a system can be represented plotting the false acceptance rate P_{fa} as a function of the false rejection rate P_{fr}. This curve (Fig. 2.8) is known as the Detection Error Trade-off (DET) curve and it is monotonous and decreasing. This curve shows all the operating points.

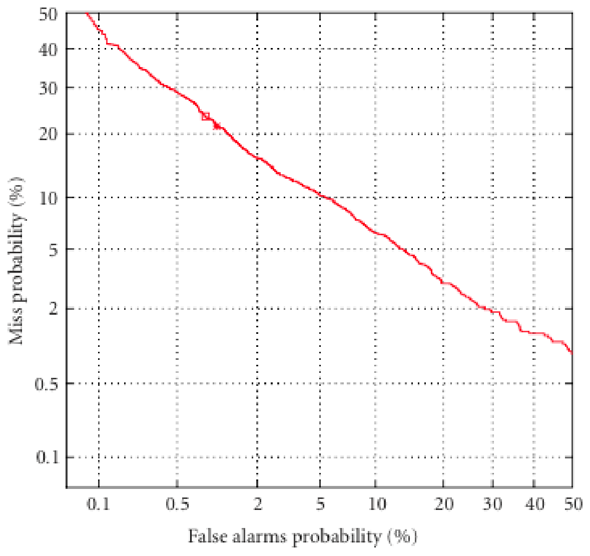


Fig. 2.8: Example of a DET curve [4]

There are other measures to summarize the performance in one single figure, the two more popular are the Equal Error Rate (EER) and the Minimum Decision Cost Function (minDCF). The EER corresponds to the operating point where $P_{fa} = P_{fr}$ and it measures the ability of a system to separate impostors from true speakers. The minDCF corresponds to the value that minimizes the cost function:

$$C = C_{fa}P_{fa}(1 - P_{target}) + C_{fr}P_{fr}P_{target} \quad (2.6)$$

where C_{fa} and C_{fr} are the costs given to false acceptances and rejections and P_{target} is the a priori probability of the target speaker [19]. The values of those variables depend on the application.

3. Project Development

With the objective of improving the baseline system proposed in [2], in this project we will use channel-compensation techniques to reduce the environmental mismatch and find a better input for the DNN stage. First we will see that it is not worth applying channel-compensation techniques at feature vectors level, because using the recent i-vector framework [12] on raw feature vectors and performing i-vector channel-compensation at this point totally outperforms those techniques.

Then, once we are working with i-vectors we want to assess the different methods to reduce the environmental mismatch. In this scenario, we observe that applying PLDA stands out among all the other methods of normalization (LDA, WCCN). It turns out to be the technique that gives us the best results. Given that fact, we want to extract from PLDA the channel-compensated i-vectors and give them as an input to the DNN.

In this part we explain PLDA in depth and the process of obtaining channel-compensated vectors. We also explain how we apply deep learning in the subject of speaker verification, showing our network's architecture, how it is trained and how we compute the scoring.

3.1. Probabilistic Linear Discriminant Analysis

We have seen before that linear dimensionality reduction methods such as LDA are often used in object recognition for feature extraction, but they don't address the problem of how to use the features for recognition. PLDA does both: extract features and combine them for recognition. As it is probabilistic it gives more weight to the most discriminative features (more impact on recognition). We can also perform dimensionality reduction with PLDA, by imposing an upper limit on the rank of the between-class variance.

The main advantage against other methods is that allows us to make inference about the classes not present during training. This is useful in speaker verification because the system have to deal with examples of novel individuals when testing.

Two different implementations have been proposed: Gaussian PLDA (G-PLDA) in [13] and Heavy Tailored PLDA (HT-PLDA) in [16]. The results presented in [15] [16] showed superior performance of the HT-PLDA model over G-PLDA. This provides strong empirical evidence of non-Gaussian behaviour of speaker and channel effects in i-vector representations. In our project we have chosen to implement G-PLDA because is more efficient computationally and also since we can perform a length normalization transformation as in [14] to the i-vectors to reduce the Gaussian behaviour and close the gap between HT-PLDA and G-PLDA.

3.1.1 Model Characterization

The i-vector of the jth session of the ith speaker ($\boldsymbol{w_{i,j}}$) can be represented as:

$$\boldsymbol{w_{i,j}} = \boldsymbol{m} + \boldsymbol{\Phi}\,\boldsymbol{\beta_i} + \boldsymbol{\Gamma\alpha_{i,j}} + \boldsymbol{\epsilon_{i,j}} \tag{3.1}$$

where

$\boldsymbol{m}$ denotes the global mean

$\boldsymbol{\Phi}\,\boldsymbol{\beta_i}$ is the speaker-specific part and describes the between-speaker variability and does not depend on the particular utterance.

$\boldsymbol{\Phi}$ is the Eigenvoices matrix (speaker-specific subspace).

$\boldsymbol{\beta_i}$ is a latent identity vector. It has a standard normal distribution N~(0,1).

$\boldsymbol{\Gamma\alpha_{i,j}} + \boldsymbol{\epsilon_{i,j}}$ is the channel component part which is utterance dependent and describes the within-speaker variability.

$\boldsymbol{\Gamma}$ is the Eigenchannel matrix (channel-specific subspace).

$\boldsymbol{\alpha_{i,j}}$ is a latent identity vector. It has a standard normal distribution N~(0,1).

$\boldsymbol{\epsilon_{i,j}}$ is a residual term vector, assumed to be Gaussian with zero mean and diagonal covariance $\boldsymbol{\Sigma}$.

$N_{\boldsymbol{\Phi}}$: is the rank of Eigenvoices matrix.

$N_{\boldsymbol{\Gamma}}$: is the rank of Eigenchannel matrix.

Since the i-vectors we are dealing with in our experiments are of sufficiently low-dimension (400) we can assume that $\boldsymbol{\Sigma}$ is a full covariance matrix, and remove the Eigenchannels $\boldsymbol{\Gamma}$ from eq. (3.1) [14].

So our final model for the G-PLDA is as follows:

$$\boldsymbol{w_{i,j}} = \boldsymbol{m} + \boldsymbol{\Phi}\, \boldsymbol{\beta_i} + \boldsymbol{\epsilon_{i,j}} \tag{2.1}$$

Training

In this step, we aim to take a set of data points $\boldsymbol{w_{i,j}}$ (i-vectors) and find the parameters $\theta = \{\boldsymbol{m}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}\}$ under which the data is more likely. We use the Expectation-Maximization algorithm to estimate the two sets of parameters in a way that likelihood is guaranteed to increase at each iteration.

E step: We compute a full posterior distribution over the latent variable $\boldsymbol{\beta_i}$

For a speaker **i** with number of sessions$\boldsymbol{N_{si}}$, we can rewrite the model as follows:

$$\begin{bmatrix} \boldsymbol{w_{i,1}} \\ \boldsymbol{w_{i,2}} \\ \vdots \\ \boldsymbol{w_{i,N_{si}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m} \\ \vdots \\ \boldsymbol{m} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Phi} \\ \boldsymbol{\Phi} \\ \vdots \\ \boldsymbol{\Phi} \end{bmatrix} \boldsymbol{\beta_i} + \begin{bmatrix} \boldsymbol{\epsilon_{i,1}} \\ \boldsymbol{\epsilon_{i,2}} \\ \vdots \\ \boldsymbol{\epsilon_{i,N_{si}}} \end{bmatrix} \tag{3.3}$$

We can write these supervectors as:

$$\boldsymbol{w_i'} = \boldsymbol{m}' + \boldsymbol{\Phi}'\, \boldsymbol{\beta_i} + \boldsymbol{\epsilon_{i,}}' \tag{3.4}$$

And we can compute the conditional probabilities as [13]:

$$\Pr\,(\boldsymbol{w_i'} \mid \boldsymbol{\beta_i}\,, \boldsymbol{\theta}\,) = \boldsymbol{N_{w_i'}}\,[\boldsymbol{\Phi}'\,\boldsymbol{\beta_i}\,, \boldsymbol{\Sigma}'\,] \tag{3.5}$$

$$\Pr(\boldsymbol{\beta_i}) = \boldsymbol{N_{\beta_i}}\,[\boldsymbol{0}\,, \mathbf{I}\,] \tag{3.6}$$

where

$$\boldsymbol{\Sigma}' = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{\Sigma} \end{bmatrix}$$

This has a form of a standard factor analyser whose likelihood is:

$$\Pr\,(\boldsymbol{w_i'}\,) = \boldsymbol{N_{w_i'}}\,[\mathbf{m}', \boldsymbol{\Phi}'\boldsymbol{\Phi}'^{\boldsymbol{T}} + \boldsymbol{\Sigma}'\,] \tag{3.7}$$

If we apply Bayes Rule:

$$\Pr\ (\boldsymbol{\beta_i} \mid \boldsymbol{w'_i}, \boldsymbol{\theta}\ ) \propto \Pr\ (\boldsymbol{w'_i} \mid \boldsymbol{\beta_i}\ , \boldsymbol{\theta}\ )\ \Pr(\boldsymbol{\beta_i}) \tag{3.8}$$

Since both terms on the right are Gaussian, the term on the left must be Gaussian. In fact, it can be shown that the first two moments of this Gaussian are:

$$E[\boldsymbol{\beta_i}] = (\boldsymbol{\Phi'^T}\ \boldsymbol{\Sigma'^{-1}}\ \boldsymbol{\Phi} + \boldsymbol{I})^{-1}\ \boldsymbol{\Phi'^T}\ \boldsymbol{\Sigma'^{-1}}(\boldsymbol{w'_i} - \boldsymbol{m'}) \tag{3.9}$$

$$E[\boldsymbol{\beta_i}\boldsymbol{\beta_i}^{\boldsymbol{T}}] = (\boldsymbol{\Phi'^T}\ \boldsymbol{\Sigma'^{-1}}\ \boldsymbol{\Phi} + \boldsymbol{I})^{-1}\ E[\boldsymbol{\beta_i}]E[\boldsymbol{\beta_i}]^{\boldsymbol{T}} \tag{3.10}$$

M step: Update the values of the parameters $\theta = \{\boldsymbol{m}, \boldsymbol{\Phi}, \boldsymbol{\Sigma}\}$

We recall eq. 3.2:

$$\boldsymbol{w_{i,j}} = \boldsymbol{m} + \boldsymbol{\Phi}\ \boldsymbol{\beta_i} + \boldsymbol{\epsilon_{i,j}} \tag{3.2}$$

We optimize:

$$Q(\theta_t, \theta_{t-1}) = \sum_{i}^{I}\sum_{j}^{\boldsymbol{N_{si}}}\int \Pr(\ \boldsymbol{\beta_i} \,\big|\, \boldsymbol{w_{i,1}}, \dots \boldsymbol{w_{i,N_{si}}}, \theta_{t-1}\big) \log[\Pr\ (\boldsymbol{w'_i} \mid \boldsymbol{\beta_i}\ )\ Pr(\ \boldsymbol{\beta_i})]\ d\boldsymbol{\beta_i} \tag{3.11}$$

where t is the iteration index.

Taking derivatives of these equations with respect to $\boldsymbol{\Phi}$ and $\boldsymbol{\Sigma}$, equating them to zero and after some algebra [13], we get the following update rules:

$$\boldsymbol{m} = \frac{1}{Ntotal}\sum_{i,j}\boldsymbol{w_{i,j}} \tag{3.12}$$

$$\boldsymbol{\Phi} = (\textstyle\sum_{i,j}(\boldsymbol{w_{i,j}} - \boldsymbol{m}\ )\ E[\boldsymbol{\beta_i}]^{\boldsymbol{T}})(\sum_{\boldsymbol{i,j}} E[\boldsymbol{\beta_i}\boldsymbol{\beta_i}^{\boldsymbol{T}}])^{-1} \tag{3.13}$$

$$\boldsymbol{\Sigma} = \frac{1}{Ntotal}\sum_{i,j} Diag[(\boldsymbol{w_{i,j}} - \boldsymbol{m}\ )(\boldsymbol{w_{i,j}} - \boldsymbol{m}\ )^T - \boldsymbol{\Phi}\ E[\boldsymbol{\beta_i}](\boldsymbol{w_{i,j}} - \boldsymbol{m}\ )^T] \tag{3.14}$$

$$Ntotal = \sum_i N_{s_i} \tag{3.15}$$

being $Ntotal$ the sum of all the sessions of all the speakers.

Scoring

For the speaker verification task, given the two i-vectors $\boldsymbol{w_m}$ and $\boldsymbol{w_t}$ involved in a trial, we are interested in testing two alternative hypotheses:

- H_s: Both i-vectors share the same speaker identity latent variable $\boldsymbol{\beta}$.
- H_d: The i-vectors were generated using different identity variables $\boldsymbol{\beta_m}$ and $\boldsymbol{\beta_t}$.

The verification score can now be computed as the log- likelihood ratio for this hypothesis test as:

$$score = \frac{\Pr(\boldsymbol{w_m}, \boldsymbol{w_t} \mid H_s)}{\Pr(\boldsymbol{w_m} \mid H_d)\Pr(\boldsymbol{w_t} \mid H_d)} \tag{3.16}$$

For the G-PLDA case, this log-likelihood ratio is easily computed in closed-form solution since the marginal likelihoods (i.e., the evidence) are Gaussian. That is,

$$score = \log N(\begin{bmatrix} \boldsymbol{w_m} \\ \boldsymbol{w_t} \end{bmatrix}; \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma_{tot}} & \boldsymbol{\Sigma_{ac}} \\ \boldsymbol{\Sigma_{ac}} & \boldsymbol{\Sigma_{tot}} \end{bmatrix}) - \log N(\begin{bmatrix} \boldsymbol{w_m} \\ \boldsymbol{w_t} \end{bmatrix}; \begin{bmatrix} \boldsymbol{m} \\ \boldsymbol{m} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma_{tot}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma_{tot}} \end{bmatrix}) \tag{3.17}$$

$$\boldsymbol{\Sigma_{tot}} = \boldsymbol{\Phi\Phi^T} + \boldsymbol{\Sigma} \tag{3.18}$$

$$\boldsymbol{\Sigma_{ac}} = \boldsymbol{\Phi\Phi^T} \tag{3.19}$$

Moreover by setting m = 0 (since it is a global offset that can be precomputed and removed from all the i-vectors) and expanding we get:

$$score = \boldsymbol{w_m}^{\boldsymbol{T}}\boldsymbol{Q}\,\boldsymbol{w_m} + \boldsymbol{w_t}^{\boldsymbol{T}}\boldsymbol{Q}\,\boldsymbol{w_t} + \boldsymbol{2}\boldsymbol{w_m}^{\boldsymbol{T}}\boldsymbol{P}\,\boldsymbol{w_t} \tag{3.20}$$

$$\boldsymbol{Q} = \boldsymbol{\Sigma_{tot}}^{-1} - (\boldsymbol{\Sigma_{tot}} - \boldsymbol{\Sigma_{ac}}\,\boldsymbol{\Sigma_{tot}}^{-1}\,\boldsymbol{\Sigma_{ac}})^{-1} \tag{3.21}$$

$$\boldsymbol{P} = \boldsymbol{\Sigma_{tot}}^{-1}\boldsymbol{\Sigma_{ac}} - (\boldsymbol{\Sigma_{tot}} - \boldsymbol{\Sigma_{ac}}\,\boldsymbol{\Sigma_{tot}}^{-1}\,\boldsymbol{\Sigma_{ac}})^{-1} \tag{3.22}$$

Even though not immediately apparent, it can be shown that **P** and **Q** both and have rank equal to the rank of $\boldsymbol{\Phi}$. This opens the door for a fast computation of the score. Based on the symmetry of **P** and assuming that $\boldsymbol{\Phi} \in \mathbb{R}^{DxK}$ $with\ K < D$ (Being K the rank of the Eigenvoices matrix = N_Φ and D the i-vectors dimension)

$$\boldsymbol{P} = [\boldsymbol{U_K}\,|\boldsymbol{U_{D-K}}]\;diag\,([\lambda_1, \dots, \lambda_k, 0, \dots, 0])\;[\boldsymbol{U_K}\,|\boldsymbol{U_{D-K}}]^{T}\;\boldsymbol{U_K}diag\,([\lambda_1, \dots, \lambda_k])\boldsymbol{U_K}^{T}$$

(3.23)

Where the K columns of $\boldsymbol{U}_K$ are orthonormal, the vector $[\lambda_1, \dots, \lambda_k]$contains the non-zero eigenvalues of **P** and the operator diag(·) places the entries of its argument in the diagonal of a matrix.

If we define:

$$\boldsymbol{\Lambda} = diag\ ([\lambda_1, \dots, \lambda_k]) \tag{3.24}$$

$$\widetilde{\boldsymbol{Q}} = \boldsymbol{U_K}^T \boldsymbol{Q}\, \boldsymbol{U_K} \tag{3.25}$$

$$\widetilde{\boldsymbol{w}}_{\boldsymbol{m}} = \boldsymbol{U_K}^T\ \boldsymbol{w_m} \tag{3.26}$$

$$\widetilde{\boldsymbol{w}}_{\boldsymbol{t}} = \boldsymbol{U_K}^T\ \boldsymbol{w_t} \tag{3.27}$$

Now, the score can be computed as:

$$score = \widetilde{\boldsymbol{w}}_{\boldsymbol{m}}{}^{\boldsymbol{T}}\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{w}}_{\boldsymbol{m}} + \widetilde{\boldsymbol{w}}_{\boldsymbol{t}}{}^{\boldsymbol{T}}\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{w}}_{\boldsymbol{t}} + \boldsymbol{2}\widetilde{\boldsymbol{w}}_{\boldsymbol{m}}{}^{\boldsymbol{T}}\boldsymbol{\Lambda}\widetilde{\boldsymbol{w}}_{\boldsymbol{t}} \tag{3.28}$$

Note that $\tilde{Q}$ and $\widetilde{\boldsymbol{w}}_{\boldsymbol{m}}$ (the enrolled model) can be precomputed and at verification time, and after projecting the test i-vector $\widetilde{\boldsymbol{w}}_{\boldsymbol{t}}$, all the remaining computations are performed in a lower dimensional space. The computational advantage becomes more significant as the ratio K/D decreases.

3.1.2. Beta Vectors Extraction

When we arrive the experimental results we will see that G-PLDA is a very powerful technique to channel-compensate the i-vectors and perform the scoring. So we found in it a really good scenario to find suitable input data for the DNN. However, G-PLDA does not give explicitly normalized i-vectors as an output, as the transformations are made at the moment of scoring.

Following the analogy of the i-vectors and GMMs we choose the identity latent variable $\boldsymbol{\beta}$ as our channel-compensated i-vector. After computing the G-PLDA matrix with the background i-vectors, we have the values of $\boldsymbol{\Phi}, \boldsymbol{\Sigma}, \boldsymbol{m}.$ Using this equation, the values of the normalized i-vectors of the speaker ith, $\boldsymbol{\beta_i}$, can be extracted (Fig. 3.1).

$$\boldsymbol{\beta_i} = (\boldsymbol{\Phi^T}\ \boldsymbol{\Sigma^{-1}}\ \boldsymbol{\Phi} + \boldsymbol{I})^{-1}\boldsymbol{\Phi^T}\ \boldsymbol{\Sigma^{-1}}\ (\boldsymbol{w_i} - \boldsymbol{m}) \tag{3.29}$$

From now on, we will call these vectors “Beta vectors”. After extracting all the vectors for models and trials we can make an assessment of the performance using the cosine distance metric and the neural network proposed later.

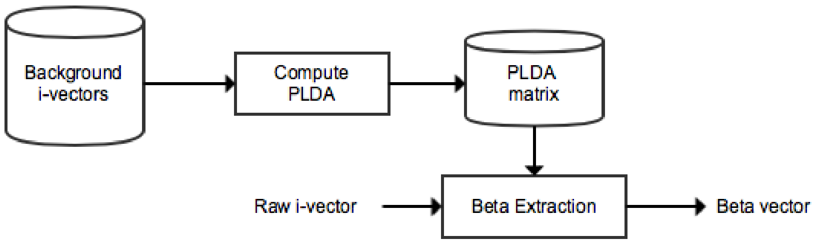


Fig. 3.1: Beta vectors extraction

3.2. Deep Learning for Speaker Verification

The main objective is to model discriminatively the target and impostor i-vectors. We are using the same deep learning architecture proposed in [2]. It consists in a hybrid DBN-DNN structure where first a DBN called Universal DBN is trained unsupervisingly using restricted Boltzmann Machines (RBMs), then is adapted and finally it is used to initialize the DNN. It has been shown that this unsupervised pre-training can set the weights of the network to be closer to a good solution than random initialization and, therefore, avoids local minima when using supervised gradient descent. Once the DNN is initialized we can train it discriminatively using the backpropagation algorithm.

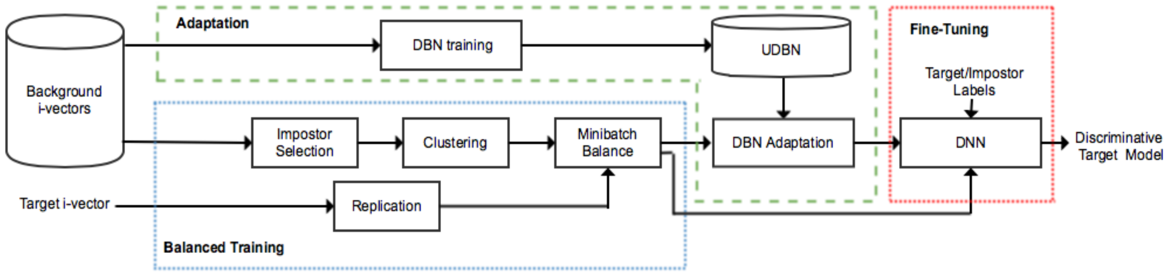


Fig. 3.2: Architecture of the DBN-DNN system

As can be seen in Fig 3.2 we can divide the structure in three steps: balanced training, adaptation and fine-tuning.

3.2.1. Balanced Training

Like other discriminative methods, DNNs need also balanced positive and negative input data to achieve their best results. However, the problem is that the amount of positive and negative data is not balanced in this case. There are a few i-vectors (in our case we have 8 per speaker in multi-session) as the positive sample and there are many impostor i-vectors as the negative ones. Training a network with such highly unbalanced data will yield overfitting.

The balanced training part tries to use the information of all available impostors and decrease their population in a reasonable way. The decreasing is carried out in two steps, selecting the most informative ones and clustering. We use the impostor selection method proposed in [1].

Firstly, we select the most informative one among all. It can be observed in the next pseudocode:

```
1. For each client i-vector $\boldsymbol{s_t} \in \boldsymbol{S}$
    1.1. Compute $score(\boldsymbol{s}_t, \boldsymbol{b}_m|_{m=1}^{M})$
    1.2. Choose the first $n$ highest scores and add their corresponding impostor indexes to a set named $H$
2. Compute the histogram of $H$ and sort it descendingly,
3. Choose the first k impostors as the selected ones.
```

where $score(\boldsymbol{s}_t, \boldsymbol{b}_m|_{m=1}^{M})$ is the cosine score between $\boldsymbol{s}_t$ and all impostors in the large dataset *B*. The parameters *n* and *k* represent, respectively, the number of the closest impostors to each target and the statistically closest ones to all available targets. They will be determined experimentally in section 4.

Secondly, as the number of selected impostors is still high in comparison to the number of target i-vectors, they are clustered by the k-means algorithm using the cosine distance criterion. The centroids of the clusters are used as the final negative samples.

On the other hand, the target i-vector is replicated as many as the number of impostor centroids. The replicated target i-vectors will not act exactly the same as each other due to the sampling noise created in the pre-training process of the network [20]. Moreover, in both adaptation and supervised learning stages, the replicated versions make the target and impostor classes having the same weights when the network parameters are being updated. Once the number of positive and negative samples is balanced, they are divided equally among minibatches. The optimum numbers of impostor clusters and minibatches will be determined experimentally.

3.2.2. Adaptation

DBNs are originally probabilistic generative models with multiple layers of stochastic hidden units above a layer of visible variables (Fig 3.3a). There is an efficient greedy layer-wise algorithm for training DBNs [21]. The algorithm treats every two adjacent layers as an RBM (Fig. 3.3b). The output of each RBM is considered as the input to its above RBM. RBMs are constructed from a layer of binary stochastic hidden units and a layer of stochastic visible units (Figs. 3.4a, 3.4b).

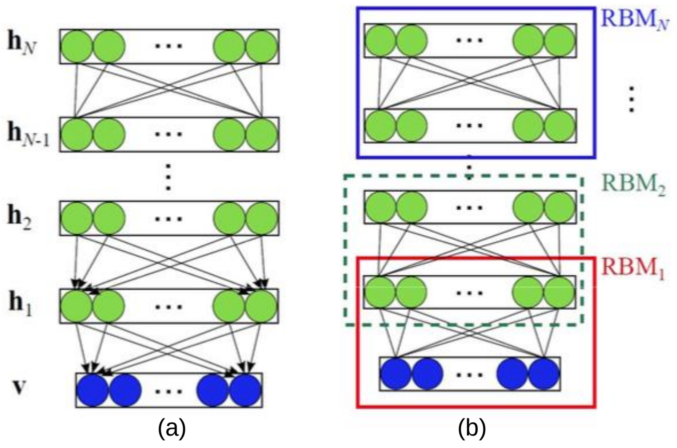


Fig. 3.3: DBN structure (a) and the DBN training (b) [2]

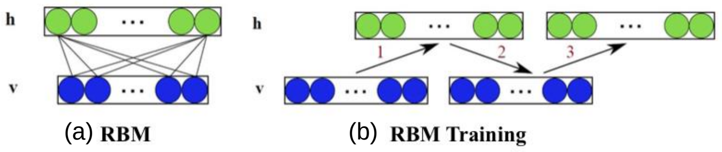


Fig. 3.4: RBM (a) and RBM training (b) [2]

Training an RBM is based on an approximated version of the Contrastive Divergence (CD) algorithm [21] [22] which consists of three steps (Fig. 3.4b). At first, hidden states (**h**) are computed given visible states (**v**), then given **h**, **v** is reconstructed, and in the third step **h** is updated given the reconstructed **v**. Finally, the change of connection weights is given as follows,

$$w_{i,j} \approx -\alpha(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon})$$

where α is the learning rate, $w_{i,j}$ represents the weight between the visible unit i and the hidden unit j, $\langle . \rangle$*data* and $\langle . \rangle$*recon* denote the expectations when the hidden state values are driven respectively from the input visible data and the reconstructed data. Actually, the training process tries to minimize the reconstruction error between the actual input data and the reconstructed one. The parameter updating process is iterated until the algorithm converges. Each iteration is called an epoch. It is possible to perform the above parameter update after processing each training example, but it is often more efficient to divide the whole input data (batch) into smaller size batches (minibatch) and to do the parameter update by an average over each minibatch.

Our global model UDBN is trained layer by layer using RBMs as explained above using all the background vectors as feeding data. As we have said before, in general, neural network parameters are initialized randomly but it has been shown [21] that the pre-trained parameters can be a better initialization for training a network. However, when a few numbers of input samples are available, just pre-training will not be enough to achieve a good model. In this case we have to adapt the UDBN parameters to each speaker's new data including both target and impostor samples. The adaptation is carried out by pre-training each network initialized by the UDBN parameters. To pre-train, only a few numbers of epochs are used, otherwise the network will be led to overfitting.

3.2.3. Fine-Tuning

Once the adaptation process is completed, a label layer is added on the top of the network (Fig 3.5) and the stochastic gradient descent backpropagation is carried out on each minibatch as the fine-tuning process. The softmax and the logistic sigmoid will be the activation functions of the top label layer and the rest hidden layer units, respectively.

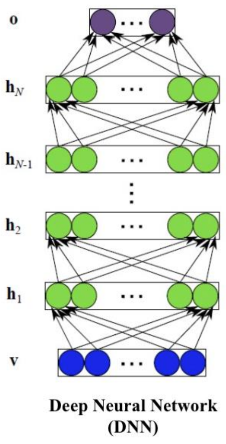


Fig. 3.5: DNN structure [2]

If the input labels in the training phase are chosen as $(l_1 = 1, l_2 = 0)$ and $(l_1 = 0, l_2 = 1)$ for target and impostor i-vectors respectively, the final output score in the testing phase will be computed in a Log Likelihood Ratio (LLR) form as follows,

$$LLR = \log(o_1) - \log(o_2)$$

where o_1 and o_2 represent the output of the first and the second units of the top layer. LLR computation helps to gaussianize the true and false score distributions which can be useful for score fusion. In addition, to make the fine-tuning process more efficient a momentum factor is used to smooth out the updates, and the weight decay method is used to penalize large weights.

4. Experimental Results

The experiments have been divided in two parts: single-session and multi-session. Single session means that we only have one utterance per target speaker while multi-session means that we have more than one. The first part will be related to feature normalization and i-vector normalization while the multi-session part will be focused to the deep learning stage. All the experiments explained are stated chronologically.

4.1. Experimental Setup

Databases: All the databases used are provided by the National Institute of Standards and Technology Speaker Recognition Evaluation series (NIST SRE). As a background vectors we use more than 6,000 speech files collected from NIST 2004 and 2005 SRE corpora. It is worth noting that in the case of NIST 2005 only the speech files of those speakers that do not appear in NIST 2006 database are collected. For the Single-session test part we use the whole core test condition of the NIST 2006 SRE. It includes 816 target models and 51,068 trials. For the Multi-session test part we use the NIST 2006 Multi-session task (8 samples per each target speaker) and consists in 699 targets and 31080 trials. All the signals have around two minutes of speech.

Software: All the experiments and computations for the feature normalization part, i-vectors framework and i-vectors normalization have been carried out using the ALIZE Toolkit in combination with the LIA_RAL libraries [23]. All the beta vectors extraction process and the DNN modeling have been done with MATLAB and UPC developed codes.

Hardware: All the experiments have been carried out in the Speech Processing Group servers.

Feature vectors: The features used in the experiments are Frequency Filtering (FF) features extracted every 10 ms using a 30 ms Hamming window. The number of static FF features is 16 and together with delta FF and delta energy, they make 33-dimensional feature vectors. Before feature extraction, speech signals are subjected to an energy-based silence removal process.

I-vectors' framework: All the i-vectors in the experiments are 400-dimensional vectors. The UBM and the T matrix have been computed using all the background vectors. The gender-independent UBM is represented as a diagonal covariance, 512-component GMM.

Assessment: The performance of every system is evaluated using the figures of the EER and the minDCF calculated using C_{fr} = 10, C_{fa} = 1 and P_t = 0.01.

4.2. Single-session Experiments

In our first experiment we want to see the effects of applying feature normalization, we will measure the performance at three points as can be seen in Fig. 4.1. In point A we will measure the contribution of feature normalization after modeling with GMM-UBM, in point B we after modeling with i-vectors and in C after applying i-vector channel compensation techniques.

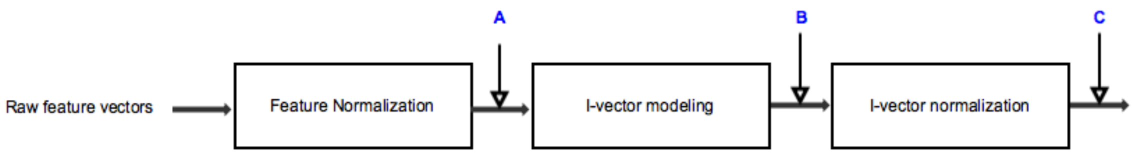


Fig. 4.1: Block scheme of the features normalization experiment

MVN has been computed globally in each utterance. Its implementation follows **eq. 2.1**. To compute the feature warping and according to the general implementation and the experiments carried out in [7] we have decided to use a sliding window of 3 seconds and a Gaussian target distribution N~(0,1).

By looking at the Table 4.1 we can observe that the contribution of feature normalization is very high at GMM-UBM level, comparing with the use of raw features we obtain a maximum relative improvement of a 40.2% and 26.7% in EER and minDCF respectively with the feature warping normalization.

Feature Normalization	EER (%)	minDCF
-	19.26	0.0737
MVN	11.61	0.0539
Warping	11.50	0.0540
Warping + MVN	11.56	0.0541

Table 4.1: Contribution of feature normalization at GMM-UBM level. (A)

Table 4.2 shows that the contribution of feature normalization after the i-vector modeling has decreased but it is still a bit better than the baseline (i-vector modeling of raw features). If we compare with the previous GMM-based results we can observe the power of modelling with i-vectors and the reason it has become a very popular technique in the speaker recognition area.

Feature Normalization	EER (%)	minDCF
-	7.17	0.0324
MVN	7.00	0.0324
Warping	7.00	0.0322
Warping + MVN	6.99	0.0320

Table 4.2: Contribution of feature normalization after i-vector modeling. (B)

After the i-vector modeling, we have applied LDA and WCCN as channel compensation tecniques and we have assessed the performance with the cosine scoring metric. The results with LDA were worse than the ones with WCCN so we are only showing the last ones (Table 4.3). We can see that applying feature normalization before modeling with i-vector decreases the performance after the application of WCCN while the best result is obtained with raw features.

Feature Normalization	EER (%)	minDCF
-	6.42	0.0321
MVN	6.60	0.0335
Warping	6.66	0.0325
Warping + MVN	6.66	0.0325

Table 4.3: Contribution of feature normalization after applying WCCN. (C)

In Table 4.4 we can see the results after applying PLDA. Before applying that technique, we length normalize all the vectors as seen in [14] to increase the performance. When applying PLDA there are two parameters we can optimize: the rank of the Eigenvoices matrix (N_{Φ}) and the number of iterations (N_I) of the EM algorithm for training the model. The optimum configuration to obtain the minimum EER was set experimentally (N_{Φ}=250 and N_I=20).As with WCCN, the best result is achieved over raw i-vectors. After seeing that results we decided not continue using feature normalization in the next experiments.

Feature Normalization	N_{Φ}	N_{I}	EER (%)	minDCF
-	250	20	4.67	0.0243
Warping + MVN	250	20	4.88	0.0258

Table 4.4: Contribution of feature normalization after applying PLDA

4.3. Multi-session Experiments

For this experiments we have eight speech utterances per speaker model, which is very good because we have more information for training and discriminate. We know that the performance of DNNs increases as the data grows so we decided to use them since we will have better results in comparison with single-session.

The first thing we do is extracting the i-vectors from the speech samples. Once we have them, we train the PLDA matrix (N_{Φ}=250 and N_I=20) and we also perform PLDA scoring. We use the trained PLDA matrix to extract the Beta vectors, which will be the input for the DNN. These vectors have reduced its dimension from 400 (i-vectors) to 250. Before training the DNN, the Beta vectors are mean and variance normalized to achieve better performance when using the network.

The baseline work [2] uses raw i-vectors as input data for a 3 layer DNN. In our implementation we will also use a 3 layer DNN but the parameters of our new network will be different because now the input data has changed. The number of hidden units will be 300 as the dimension of our vectors is smaller. For the balanced training stage, the number of minibatches and the number of impostor clusters are set experimentally to 3 and 24. Each minibatch will include 8 impostor centroids and 8 target samples. The eight samples will be replicated 2 times in order to be used in the different minibatches. The impostor selection is carried out with the method explained in section 3.2.2. By setting n=50 we look for the value of k that minimizes the EER, in Fig 4.2 can be seen that this parameter is k=800.

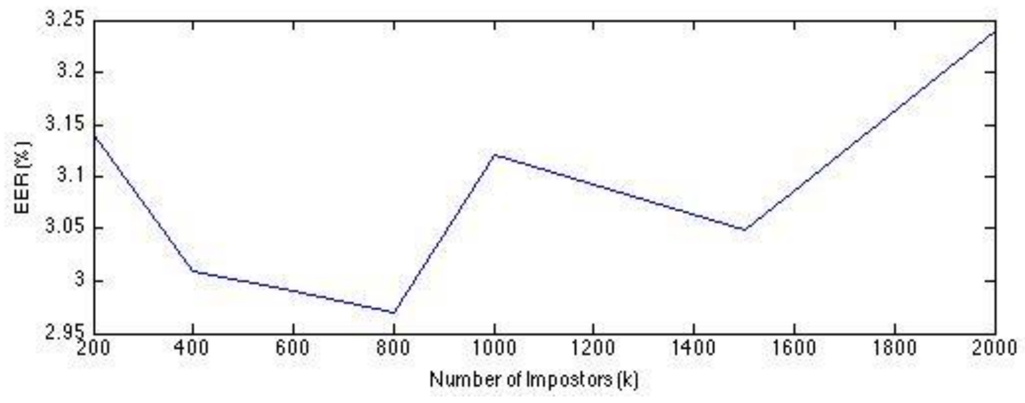


Fig. 4.2: Determination of k for Impostor Selection.

UDBN is trained with the same background i-vectors of the impostor database. As the input Beta vectors are real-valued, a Gaussian-Bernoulli RBM [20] [24] is employed. The learning rate (α), number of epochs (NofE), momentum, and weight decay are set respectively to 0.005, 200, 0.9, and 0.0002

The generative parts of the speaker models are initialized by the UDBN parameters and then are adapted with α = 0.0015 and NofE = 10. The momentum and weight decay values are kept the same as in UDBN. The whole backpropagation is carried out with
α = 0.1, NofE = 500, and a fixed momentum of 0.9. The weight decay for both top layer pre-training and the whole backpropagation is set to 0.0012.

In Table 4.5 we can see how the DNN results improve when applying the impostor selection and the adaptation methods proposed before. We can observe a relative improvement of 12% in the EER and 12.8% in the minDCF between the initial and the final implementation.

	EER (%)	**minDCF**
DNN	3.23	0.0148
DNN + Impostor Selection	2.97	0.0131
DNN + Impostor Selection + Adaptation	2.84	0.0129

Table 4.5: Comparison of DNN implementations

In the next figure (Fig 4.3) we can observe the final performance in the form of a DET curve of the baseline systems and the systems implemented in this thesis, which include channel-compensation.

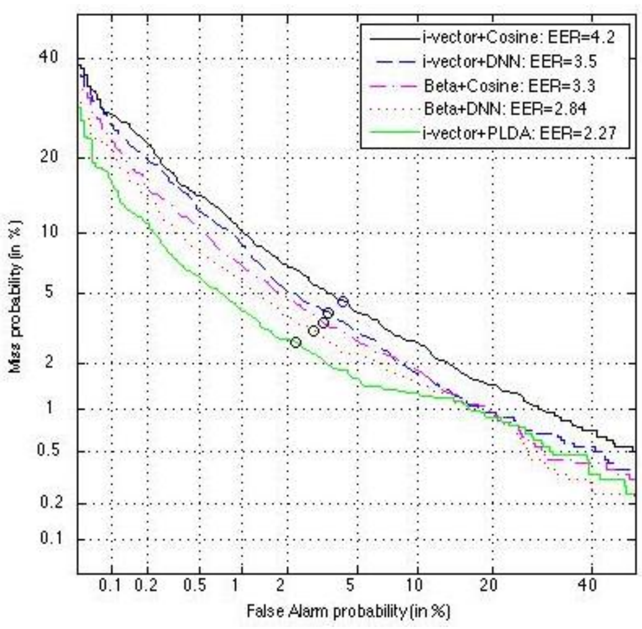


Fig. 4.3: DET Curve of all the implementations

With the Beta vector proposal and scoring with the cosine metric we have obtained a relative improvement of 21.4% and 21% in the EER and minDCF with respect the raw i-vectors. If we change the classifier to the DNN the relative improvement increases to 32.3% and 32.1% respectively. If we compare the two DNNs systems we find that Beta-DNN outperforms the baseline DNN with 18.9% and 25% relative improvement in ERR and minDCF. However, the best result has been obtained modeling and scoring with PLDA directly using raw i-vectors (Section 3.1.1), being the relative improvement of 45.9% in EER and 44.7% in minDCF with respect to the raw i-vectors. These final numerical results can be seen in Table 4.6.

	EER (%)	**minDCF**
i-vectors + Cosine	4.20	0.0190
i-vectors + DNN	3.50	0.0172
Beta vectors + Cosine	3.30	0.0150
Beta vectors + DNN	2.84	0.0129
i-vectors + PLDA	2.27	0.0105

Table 4.6: Comparison of all the implementations

5. Budget

The length of the project has been 12 ECTS, which correspond approximately to 360h. The average salary of a junior engineer is around 15€/h.

$$C_{engineer} = \frac{15€}{h}\, 360\, h = 5400\, €$$

The majority of the project has been carried out at home, using the university servers' resources. The software used has been the ALIZE toolkit and MATLAB. The first one is free, but the second one has a paying license. Considering that the servers and MATLAB are used by a lot of projects and students it is very difficult to compute the real cost and its amortization. We will consider an approximate cost of 1600€ for all the duration of the project.

At the end, the total budget is:

$$C_{project} = C_{engineer} + C_{software} + C_{hardware} = 7000\, €$$

6. Conclusions and future development

In this project we aim to combine channel-compensation techniques with deep learning for speaker recognition and outperform the baseline system given. We have proposed channel-compensated i-vectors called Beta vectors as an input for our DBN-DNN hybrid deep learning system.

In section 4.2 we have seen that the contribution of feature normalization is not very useful once we apply i-vector modeling and i-vector channel-compensation techniques. In section 4.3 we have seen that with the Beta vector proposal and scoring with the cosine metric we obtain a relative improvement of 21.4% and 21% in the EER and minDCF with respect the raw i-vectors. If we change the classifier to the DNN the relative improvement increases to 32.3% and 32.1% respectively. Our Beta-DNN outperforms the i-vector-DNN baseline system with 18.9% and 25% relative improvement in ERR and minDCF. So we can state that with our contribution we have achieved the goals proposed at the start of the thesis.

However, although the final results with the DNN are very good compared with the baseline, there is a gap between DNN and the PLDA performance. Maybe with the use of a database with more data available we could exploit more the DNN strengths and close the gap between them. Anyway, this shows that there is still work to do in the ambit of deep learning for speaker recognition and the door is open for future new implementations and refinements.

This work was successfully published in the form of a poster in the *Red Temática de Tecnologias del Habla* (RTTH) Summer School in July of 2015 held in Barcelona.

Bibliography

[1] O. Ghahabi and J. Hernando, "Deep belief networks for i-vector based speaker recognition," *Acoustics, Speech and Signal Processing (ICASSP) 2014 IEEE International Conference*, pp. 1700,1704, May 2014.

[2] O. Ghahabi and J. Hernando, "Deep Neural Networks for i-Vector Modeling in Speaker Recognition," *Submitted to IEEE Transactions on Audio, Speech, and Language Processing*, 2015.

[3] O. Ghahabi and J. Hernando, "i-Vector modeling with deep belief networks for multi-session speaker recognition," *Odyssey*, pp. 305-310, 2014.

[4] J. F. Bonastre, C. Fredouille, G. Gravier, Magrin I. Chagnolleau, S. Meignier, T. Merlin, Ortega J. Garcia, Petrovska Delacretaz, Reynolds, Bimbot, "A Tutorial on Text-Independent Speaker Verification," *EURASIP Journal on Applied Signal Processing*, vol. 4, pp. 430-451, 2004.

[5] S. Furui, "Comparison of speaker recognition methods using static features and dynamic features," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 29, pp. 342–350, 1981.

[6] R. Duncan, "A description and comparison of the feature sets used in speech processing," 2000.

[7] J., Sridharan, S. Pelecanos, "Feature warping for robust speaker verification.," *Proc. Speaker Odyssey: the Speaker Recognition Workshop*, pp. 213–218, 2001.

[8] H. Hermanksy and N. Morgan, "RASTA processing of speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, pp. 578–589, 1994.

[9] Md J. Alam, P. Ouellet, P. Kenny, and D. O'Shaughnessy, "Comparative Evaluation of Feature Normalization Techniques for Speaker Verification," *NOLISP'11 Proceedings of the 5th international conference on Advances in nonlinear speech processing*, pp. 246-253.

[10] T. F. Quatieri, and R. B. Dunn D. A. Reynolds, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, pp. 19-41, 2000.

[11] G. Boulianne, P. Ouellet, and P. Dumouchel P. Kenny, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 4, pp. 1435–1447, May 2007.

[12] P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet N. Dehak, "Front-End Factor Analysis For Speaker Verification," *IEEE Transactions on Audio, Speech and Language Processing,* vol. 19, no. 4, pp. 788 - 798, May 2010.

[13] S. J. D. Prince, "Probabilistic Linear Discriminant Analysis for Inferences About Identity," *IEEE 11th International Conference on Computer Vision (ICCV)*, pp. 1-8, 2007.

[14] D., Espy-Wilson, C. Garcia-Romero, "Analysis of i-vector length normalization in speaker recognition systems.," *Proceedings of Interspeech*, pp. 249–252, 2011.

[15] P. Matejka, "Full-Covariance UBM and Heavy-Tailed PLDA in I-Vector Speaker Verification," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2011.

[16] P. Kenny, "Bayesian Speaker Verification with Heavy-Tailed Priors," *Odyssey 2010 - The Speaker and Language Recognition Workshop*, 2010.

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

[18] L Deng and Y. Dong, "Deep Learning: Methods and Applications," *Trends Signal Process.*, pp. 197-387, June 2014.

[19] (2006) NIST_SRE. [Online]. http://www.itl.nist.gov/iad/mig/tests/spk/2006/sre-06_evalplan-v9.pdf

[20] G.E. Hinton, "A practical guide to training restricted boltzmann machines," *Neural Networks: Tricks of the Trade in Lecture Notes in Computer Science*, no. 7700, pp. 599–619., January 2012.

[21] S. Osindero, and Y-W. Teh G.E. Hinton, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, May 2006.

[22] G.E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[23] J-F. Bonastre, B. Fauve, K. Lee, C. Lvy, H. Li, J. Mason, and J-Y. Parfait, A. Larcher, "ALIZE 3.0 open source toolkit for state-of-the-art speaker recognition," *Proc. Interspeech*, pp. 2768–2771, 2013.

[24] D. Yu, L. Deng, and A. Acero G.E. Dahl, "Context- dependent pre-trained deep neural networks for large- vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 20, no. 1, pp. 30–42, January 2012.

Glossary

DBN: Deep Belief Network.

DCF: Decision Cost Function.

DET: Detection-Error Trade-off.

DNN: Deep Neural Network.

EER: Equal Error Rate.

FF: Frequency Filtering

GMM: Gaussian Mixture Model.

LDA: Linear Discriminant Analysis.

MVN: Mean and Variance Normalization.

NIST: National Institute of Standards and Technology.

PLDA: Probabilistic Linear Discriminant Analysis.

RBM: Restricted Boltzmann Machine.

SRE: Speaker Recognition Evaluation.

UBM: Universal Background Model.

UDBN: Universal Deep Belief Network.

WCCN: Within Class Covariance Normalization.

Appendices

Work Packages

Project: **Documentation**	WP ref: **DC**	
Major constituent: Documents	Sheet 1 of 6	
Short description: It is all the documentation that state the progress, information and results of the project.	Planned start date: 16/02/15 Planned end date: 10/07/15 Start event: End event:	
Internal task T1: Project Plan Proposal & WorkPlan Internal task T2: Critical Design Review Internal task T3: Final Memory	Deliverables: Every task has his own deliverable	Dates:

Project: **Initial Research and Background Learning**	WP ref: **IRBL**	
Major constituent: Documents	Sheet 2 of 6	
Short description: Obtain the background knowledge needed to reach the goals when realizing the project.	Planned start date: 16/02/15 Planned end date: 06/03/15 Start event: End event:	
Internal task T1: Theory of text-independent speaker verification and feature vectors post-processing techniques Internal task T2: The i-vector methodology inside speaker recognition and i-vector post-processing techniques Internal task T3: Theory about deep learning and about its implementation in speaker verification systems	Deliverables: Every task has his own deliverable	Dates:

Project: **Baseline Experiment**	WP ref: **BE**
Major constituent: Simulation	Sheet 3 of 6
Short description: First experiment to obtain the initial results of the system.	Planned start date: 09/03/15 Planned end date: 27/03/15 Start event: End event:
Internal task T1: Development of the experiment Internal task T2: Experiment set up Internal task T3: Analysis of the results	Deliverables: | Dates:

Project: **Feature Vectors Post-Processing Experiment**	WP ref: **FVPPE**
Major constituent: Simulation	Sheet 4 of 6
Short description: Design the feature vector post-processing module, integrate it in the system, simulate the whole system and check the performance.	Planned start date: 30/03/15 Planned end date: 15/05/15 Start event: End event:
Internal task T1: Design of the features post-processing module Internal task T2: System integration Internal task T3: Development of the experiment Internal task T4: Analysis of the results	Deliverables: | Dates:

Project: **I-Vectors Post-Processing Experiment**	WP ref: **IVPPE**
Major constituent: Simulation	Sheet 5 of 6

Short description: Design the i-vector post-processing module, integrate it in the system, simulate the whole system and check the performance.	Planned start date:08/04/15 Planned end date:05/06/15	
	Start event: End event:	
Internal task T1: Design of the features post-processing module Internal task T2: System integration Internal task T3: Development of the experiment Internal task T4: Analysis of the results	Deliverables:	Dates:

Project: **Integration into Deep Belief Network**	WP ref: **DBN**	
Major constituent: Simulation	Sheet 6 of 6	
Short description: Integrate the post-processing stages in the Deep Belief Network system. Tune the network, test and check the performance.	Planned start date: 08/06/15 Planned end date: 06/07/15	
	Start event: End event:	
Internal task T1: System integration Internal task T2: Development and testing Internal task T3: Analysis of the results	Deliverables:	Dates:

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
DC	T1	Project Proposal & WorkPlan	Document	06/03/15
DC	T2	Critical Design Review	Document	20/04/15
DC	T3	Final Memory	Document	10/07/15