Doctoral Dissertation

Doctoral Program in Computer and Control Engineering (36thcycle)

Speaker Verification and Language Recognition

By

Salvatore Sarni

******

Supervisor(s):

Prof. S. Cumani

Politecnico di Torino

2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Salvatore Sarni
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

I would like to thank all the people with whom I collaborated during these years at Politecnico di Torino. In particular, a lot of gratitude goes to my PhD tutor Prof. Sandro Cumani for giving me such an opportunity and for his support throughout the development of this work. A big thank goes to my family for their constant support.

Abstract

Automatic systems for identity recognition are ubiquitous. A person's identity can be found and matched exploiting some of its biometrical traits. The trivial operation of unlocking a smartphone hides some form of verification, usually face or fingerprint verification. Personal assistants answer our questions only after verifying our identity through our voices. Speaker verification is the main focus of this work. In speaker verification, given an identity and an audio utterance, the system should be able to assess if the audio was spoken by such identity.

The starting point of the verification pipeline is the audio. Due to their nature, voice recordings come with different durations and from various sources. A fixed dimensional utterance representation that contains speaker-discriminant information is then required. The advances in the field of Deep Learning made Deep Neural Networks (DNN) the state-of-the-art technique to process utterances and extract such representations, namely the embeddings. In this work, we explore the latest and most common architecture employed for the speaker verification task.

The nature of audio and the need to model its long-range temporal dependencies resulted in a variety of solutions, from Time Delay Neural Networks (TDNN) to Residual Networks (ResNet) and finally, the latest experiments with Transformers models. Architectures are typically trained on a set of background speakers using a multi-class classification paradigm. The network processes the acoustic features and aggregates them using a pooling layer, obtaining a fixed-length embedding from which speaker posterior probabilities are then computed, typically with a softmax layer.

The need to identify or verify a language arises for a plethora of applications. Language-dependent systems may be required to automatically adapt their model to the language being spoken. Given the small number of languages compared to that of potential speakers, the number of classes is reduced and while training

data is generally abundant, low-resource languages pose a significant challenge. Nevertheless, speaker verification methods can be successfully adapted. In this work, we focus on extending state-of-the-art DNN-based embedding models to the language recognition task.

To assess whether two embeddings convey the same information, being the speaker or the language, a scoring system is needed. Various backend classifiers can be used for this purpose, ranging from distance metrics to probabilistic models. However, no matter how the score is obtained, it needs to be interpreted and each application may have a specific threshold for accepting or rejecting the same speaker (or language) hypothesis. An optimal system should exhibit consistent performance across different applications, regardless of the chosen threshold.

Different techniques can be employed to address the calibration problem. In this study, we introduce a new generative model that can effectively use additional information from the audio, such as utterance duration. Moreover, logistic regression and score normalization are two distinct state-of-the-art approaches used to improve calibration. In this study, we explore a combination of both methods as well as a possible alternative to the score normalization approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A spoken utterance carries different kinds of information at several degrees. The advancement in computational power and the application of complex solutions to complex problems has facilitated the research and development of tools that extract such information. The increasing need for security or the automation of different operations has led to the rapid growth of several fields related to the identification and verification of a person's characteristics. Speaker recognition is a field of study that involves identifying and verifying the identity of an individual based on their unique vocal characteristics. The primary objective of speaker recognition is to create systems that can differentiate between different voices and match them with their corresponding identities. This technology has several potential use cases, including security, law enforcement, and authentication.

On the other hand, language recognition focuses on identifying the language being spoken in an audio recording or speech signal. The objective of language recognition is to accurately identify and classify spoken languages based on their unique phonetic, prosodic, and spectral characteristics. Some potential applications include speech-to-text transcription or frontend to multilingual automatic systems.

With the advent of deep learning and neural networks, these fields, among others, have made significant progress in recent years and have become an integral part of many everyday devices and services.

This work will present the author's contribution to the speaker verification and language recognition fields while introducing and analyzing the deep neural networks model that represents a part of the state-of-the-art approaches in these fields.

1.1 Speaker Recognition

Speaker recognition systems are designed to recognize speakers by analyzing their voices. The field of speaker recognition is divided into two subtopics: speaker identification and speaker verification.
Speaker identification is a multi-class classification problem where the system is required to identify the speaker among a set of enrolled speakers for a given test utterance.
Speaker verification, instead, is a binary classification problem where the system has to determine whether a test audio and an enrollment utterance belong to the same speaker or not.
Furthermore, speaker identification can be classified into two tasks: closed-set and open-set. In the closed-set scenario, the number of enrolled speakers is fixed and limited, while in the open-set scenario, the set of enrolled speakers is not limited, and the system needs to distinguish between known and unknown speakers. Making open-set speaker identification a more demanding task.

1.2 Language Recognition

Similarly to speaker recognition, language recognition can be viewed as a multi-class classification problem. Its objective is to identify the language spoken in a given utterance. The set of languages can be defined and limited, which results in a closed-set scenario. Conversely, in an open-set scenario, the test utterance could belong to a language that is not specified in the given set.

1.3 Backend Model

In recent years neural networks have become the state-of-the-art for extracting speaker or language information-rich utterance representations. However, to determine if two embeddings encode the same information, an additional step is necessary. The classifier, which is the first component of the backend, may have different characteristics to tackle different requirements and scenarios. Nonetheless, the score it produces still needs to be interpreted. Calibration is the final step where an appropriate threshold is chosen to optimize the performance for specific applications.

Speaker verification and language recognition are the main focus of this work. The two fields share some common concepts and techniques, in the following we will discuss our work on the deep neural networks frontend while introducing the current state-of-the-art. Then, we will introduce and discuss the author's contribution to the state–of–the–art techniques employed in the backend classifier and calibration models.

1.4 Outline

The work is organized as follows:

- Chapter 2 introduce the basic concept and tools in feature extraction while presenting some background works
- In Chapter 3 we present and examine our work on the most recent frontend techniques used both in speaker verification and language recognition, that is deep neural networks
- Chapter 4 will present some of the backend classifier models and the calibration issue
- The experimental results related to the author's contribution are collected in Chapter 5
- Finally in Chapter 6 we drawn conclusions and explore possible future works

Chapter 2

Audio modeling

An audio signal is continuous, meaning it is represented by an infinite number of values. However, computers and other digital systems can only handle discrete signals that have a finite number of values. In this chapter, we will discuss some techniques used to transform an audio sample into something that can be used and interpreted by a machine. These transformations, which are commonly called feature extraction, represent a crucial first step in many different language or speaker recognition systems.

2.1 Feature extraction

In this section, we will explore the first step taken by any automatic system when recognizing a speaker or language. This involves extracting a digital representation of the most relevant aspect of the analog signal.

2.1.1 Sampling, quantization and filtering

In order to process or store an analog signal in a digital format, it must be sampled and quantized.
Sampling is the process of taking discrete samples of an analog signal at regular time intervals. This corresponds, in the time domain, to a multiplication of the input signal with a sequence of impulses $\sum\limits_{k} \delta(t - kt_s)$, where t_s is the sampling time. The

sampled signal $y_s(t)$ is thus defined as

$$y_s(t) = \sum_k [y(kt_s)\delta(t - kt_s)] \tag{2.1}$$

where $y(t)$ is the input signal. In the frequency domain, the Fourier transform of $y_s(t)$ is defined as a sequence of impulses convoluted with the spectrum of the analog signal

$$Y_s(\omega) = \frac{1}{t_s} \sum_k Y\left(\omega + \frac{2\pi k}{t_s}\right) \tag{2.2}$$

The Nyquist-Shannon sampling theorem states that in order to accurately represent an analog signal in a digital format, the sampling frequency must be at least twice the highest frequency component of the signal. However, since frequencies higher than 4 kHz are hardly sensed by the common human hears, it follows that 8 kHz is an acceptable sampling frequency.
Quantization is the process of mapping the continuous range of possible values of an analog signal with a finite set of discrete values, with an inevitable loss. Linear quantization uniformly divides the input range and assigns the amplitude of each sample to the nearest value of the corresponding interval. In this scenario, the absolute value of the quantization error will be constant for any given input. However, the amplitude distribution of an audio signal is highly non-linear. Logarithmic quantization solves this issue by applying a linear quantization to the logarithm of the acoustic signal. Moreover, the μ–law (used in American communication nets) and the A-law (used in European communication nets) [1] are two examples of functions used in practice, due to the fact that the logarithm is not defined in zero. When working with telephone speech, values are commonly represented using 8 bits. However, in practice, a larger number of bits is used to perform an initial linear quantization. Finally, one of the laws shown in Figure 2.1 is utilized to map these values to 8 bits.
To obtain a more flat spectrum of the signal for a given frequency band, a first order pre-emphasis filter is used to filter the discretized samples. The transfer function in the frequency domain is defined as

$$H(z) = 1 - az^{-1} \tag{2.3}$$

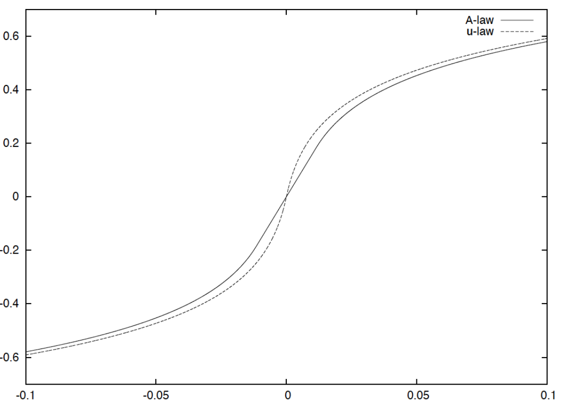


Figure 2.1 The A and μ–law

Equivalent, in time-domain, to

$$\hat{Y}(k) = Y(k) - aY(k-1) \tag{2.4}$$

where a is a constant, typically $a = 0.95$ in real applications.

2.1.2 Mel-scaled filter banks

In audio signals (Fig.2.2a), frequencies usually change over time. Therefore, performing a Fourier transform on the entire signal would result in the loss of information about their changes over time. Since it is reasonable to assume that the frequencies in a signal remain stable over a brief period of time, the signal is split into frames, with a duration of about 10ms. A set of samples is grouped together as:

$$X_t(n) = \hat{Y}(M_a t + n) \quad 0 \leq n \leq N_a - 1, 0 \leq t \leq T - 1 \tag{2.5}$$

where N_a is the grouping window size, M_a the size of the shift, and T is the duration in frames of the signal. Grouping the audio signal into frames can cause the distortion of the spectrum of the samples within each frame (this is known as Gibs phenomenon), a Hamming window is utilized to mitigate the impact of the samples located near the frame's edges

$$\hat{X}_t(n) = X_t(n)W(n) \quad 0 \leq n \leq N_a - 1, 0 \leq t \leq T - 1 \tag{2.6}$$

where

$$W(n) = \begin{cases} c + (1+c)\cos(\frac{\pi n}{N-1} - \frac{\pi}{2}) & if \quad 0 \leq n \leq N-1 \\ 0 & otherwise \end{cases} \tag{2.7}$$

and c = 0.54 in real applications. This method produces a better approximation of the original signal's spectrum (Fig.2.2b), sacrificing the information encoded in the samples at the edges of each frame. To address this issue, the window is only shifted by half its size. As a result, the samples located at the borders of a frame are now situated in the middle of the preceding or succeeding frame. At this point, the data within each frame undergoes the Fourier Transform

$$X_f(j) = F(\hat{X}_t(n))(j) \quad 0 \leq n \leq N_a - 1, 0 \leq t \leq T-1 \tag{2.8}$$

Since the human auditory system cannot perceive phase variations, the focus is on the amplitude spectrum. Filters that emulate the human auditory system are used to define frequency bands on the mel scale [1] for dividing the acoustic signal spectrum (Fig.2.2c). For each band, the energy of the corresponding samples is then evaluated.

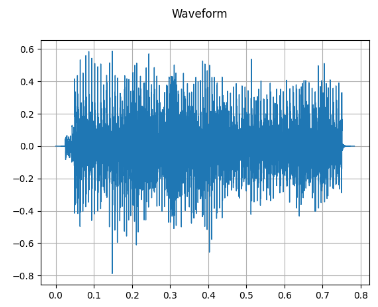
$$E_i(f) = \sum_{j=L_i}^{H_i} |X_f(j)|^2 \quad 1 \leq i \leq N_f \tag{2.9}$$

where $E_i(f)$ is the energy of the i–th band, L_i is the lower bound of the corresponding band, H_i is its higher bound and N_f is the number of bands.

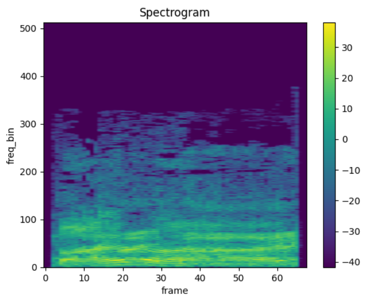
2.1.3 Mel-Frequency Cepstral Coefficients

A Discrete Cosine Transform (DCT) can be applied to decorrelate the filter bank coefficients and create a compressed representation of the filter banks. Mel-Frequency Cepstral Coefficients (MFCC) [1, 2] (Fig. 2.2d) are obtained by computing the DCT of the logarithm of the energy parameters $E_i(f)$. The i–th MFCC for frame k is given by

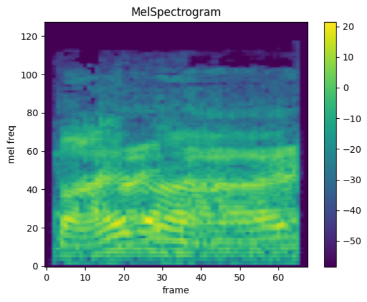
$$C_i(k) = \sum_{j=1}^{N_f} \log(E_j(k)) \cos\left[i(j - \frac{1}{2})\frac{\pi}{N_f}\right] \quad 0 \leq i \leq N_f \tag{2.10}$$



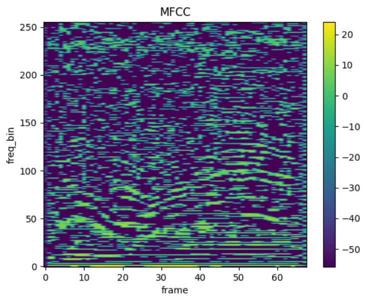
(a) Waveform



(b) Spectrogram



(c) MelSpectrogram



(d) MFCC

Figure 2.2 Audio Feature Extraction

and the total energy of the frame can be evaluated as

$$E(k) = \sum_{j=1}^{N_f} E_j(k) \tag{2.11}$$

The information conveyed by cepstral parameters with higher indices diminishes, and both $E(k)$ and $C_0(k)$ contain the same information. Consequently, we can discard the high index parameters and $C_0(k)$. Usually between 12 and 24 cepstral parameters are selected.

Using the differential counterparts of MFCCs, also known as depstral parameters, can lead to a more accurate acoustic signal model. These parameters are determined by estimating the temporal derivative of MFCCs. To ensure that the variances of both the depstral parameters and MFCCs are comparable, a gain, denoted as G, may be introduced

$$\Delta\bar{C}_i(k) = G \sum_{j=-N}^{N} j\bar{C}_i(k-j) \quad 1 \leq i \leq p \tag{2.12}$$

where N is half the size of the window used to approximate the derivative. Similarly, differential energy can be evaluated as

$$\Delta E(k) = \sum_{j=-N}^{N} jE(k-j) \tag{2.13}$$

Additionally, the second-order derivatives of cepstral coefficients may be calculated in a similar manner. Once cepstral parameters and their derivatives have been obtained, they may be combined to form the feature vector.

$$O_t = \{\bar{C}_1(t), ..., \bar{C}_p(t), \Delta\bar{C}_1(t), ..., \Delta\bar{C}_p(t), \\ \Delta\Delta\bar{C}_1(t), ..., \Delta\Delta\bar{C}_p(t), E(t), \Delta E(t), \Delta\Delta E(t)\} \tag{2.14}$$

2.2 Speaker verification with Latent Variable Models

Speaker recognition systems deal with audio recordings originating from different sources, which may vary in duration and contain varying amounts of speaker-related information. The first challenge faced in building such a system is to obtain a fixed-dimension representation containing only the most relevant features. Gaussian Mixture Models (GMMs) [3, 4] were one of the first successful approaches employed to extract speaker information from audio utterances. Later, the combination of Joint Factor Analysis (JFA) with the GMM model led to the development of the i-vector representation, a long-standing state-of-the-art approach in speaker verification.

2.2.1 i-vector

Speaker-related information can be extracted by using a GMM universal background model (UBM) [5, 6]. The UBM is obtained by training a single GMM over a large set of utterances from different speakers. For each utterance, a GMM can then be trained by adapting its means starting from the UBM ones. Joint Factor Analysis (JFA) can be used to improve this process [7, 8]. JFA represents a speaker's utterances through a supervector M:

$$M = m + Vy + Ux + Dz \tag{2.15}$$

where m is a speaker-independent supervector derived by concatenating the UBM means, and V, U, and D define the subspaces for speaker, session, and residual noises. The vectors y, x, and z characterize the speaker, session, and common factors. Speaker factors are assumed to take the same value for all utterances of the same speakers, whereas the other factors are utterance-dependent. JFA offered a framework for speaker modeling and scoring of test utterances. However, the classical JFA approach is computationally intensive. Consequently, JFA has primarily been employed as a feature extractor. In classical JFA modeling, two distinct spaces are defined: the speaker space, denoted by the matrix V, and the channel space, denoted by the matrix U. This configuration enables JFA to effectively estimate and compensate for channel effects [9, 10]. However, some results obtained in [11] showed that channel factors may contain useful information about the speaker, even if channel factors were assumed to only model channel and noise. Consequently, an adapted version of JFA was proposed, where a single subspace is estimated, assuming that most of the pertinent information lies within it [12]. The supervector M is defined as:

$$M = m + Tw \tag{2.16}$$

here m represents again the speaker-independent supervector, which can be derived from a Universal Background Model (UBM) supervector. The rectangular matrix T encompasses both channel and speaker information, and its columns span the subspace where the GMMs are situated. Finally, w is a random vector that encodes a GMM in the subspace associated with T. The i-vector was a fundamental step in speaker representation and one of the latest state-of-the-art before the advent of approaches derived from deep learning and neural networks. Initially, there were attempts to improve the performance of i-vectors by integrating components trained with deep neural networks [13, 14]. Nevertheless, significant progress has been made by directly extracting speaker representations using neural networks. As a result, neural-networks based speaker embeddings have now become the latest standard in speaker verification.

2.2.2 Embeddings

Given an input with D dimensions, an embedding is a vector with $d \ll D$ dimensions that represents the input in the new space. An embedding extractor has to map similar objects close to each other in this new space. Embeddings offer a fixed and low-dimensional representation of the input, which can be useful in various applications, such as movie recommendation systems or face recognition [15, 16]. In our case, they can be used to extract speaker or language-related information from raw audio data [17, 18]. There are different techniques to extract embeddings, with deep learning being the state-of-the-art approach for many tasks. Deep neural models can build a hidden representation of the input that can be used to optimize a final objective. Networks can be trained to optimize different functions and the embeddings are extracted from one or more of their hidden layers [19]. The number of units defined within each architecture defines the dimension of the embedding space.

In speaker recognition, networks are trained to classify speakers [18, 20]. The network inputs consist of the acoustic features of a speech segment and the corresponding speaker's identity, represented by a label. The network has to learn the distinct characteristics of each speaker in the hidden layers, which results in speakers who share similar traits being projected closer in the embedding space. The same network can then be used to extract embeddings from speakers that were not encountered during the training phase [21]. Similar considerations apply to the task of language recognition [22].

In the upcoming chapter, we will introduce and examine the state-of-the-art techniques utilized for extracting these embeddings, as well as analyze the author's contributions to such techniques.

Chapter 3

Deep Learning for embeddings extraction

3.1 Neural Networks

In this chapter we introduce the fundamental concepts of artificial neural networks (ANN), which have been instrumental in the development of our work on calibration and normalization of speaker verification scores and our novel contributions to the extraction of effective language embeddings [23]. It should be noted that the goal of the following sections is not to provide a comprehensive overview of the history of ANNs. Rather, the focus is on presenting the principles and key concepts that have been utilized to solve the problem of speaker verification and language recognition.

The current concept of a "neural network" was first introduced in [24]. In this work, the primary focus was on constructing a mathematical model to replicate the functions of the human brain, which relies on *networks* of basic processing units called *neurons*, Figure 3.1. The perceptron, introduced in [25], marked an advancement in this model by proposing that each neuron should learn a specific function. This type of architecture is able to learn and realize all elementary logical operations. However, for those functions that require a non-linear separation rule, a single perceptron is not sufficient. Instead, a Multi-Layer perceptron architecture is needed, where a *layer* consists of perceptrons at the same architectural level.

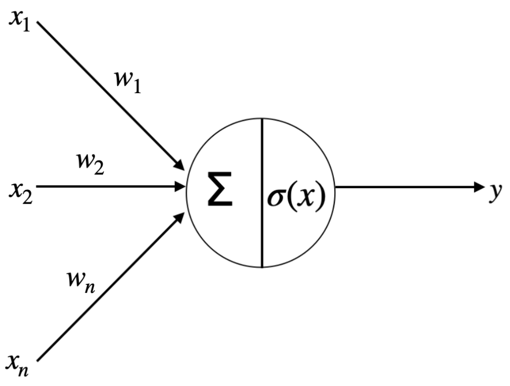


Figure 3.1 Artificial neuron model

Activation Functions The parameters of each node (Fig. 3.1) can be rearranged into a weights vector $w = [w_1, w_2, ..., w_n]^T$ and a biases vector $b = [b_1, b_2, ..., b_n]^T$. Given the activation function $\sigma(\cdot)$, the output is computed through vectors multiplication as

$$h = \sigma(w^T x + b) \tag{3.1}$$

Activation functions can be used to introduce a non-nonlinearity to the node. Some examples of activation functions include:

- logistic: $\sigma(z) = \frac{1}{1+e^{-z}}$
- tanh: $\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Rectified Linear Unit (ReLU): $\sigma(z) = max(0, z)$

In our work, we mainly used the ReLU function.

3.1.1 Feedforward Neural Networks

The multilayer perceptron is an example of a feedforward neural network. In these architectures, depicted in Figure 3.2, information flows unidirectionally from the input layer through one or more hidden layers to the output layer. Each perceptron, or in general a node, is connected to nodes in the preceding and succeeding layers through weighted connections (Fig. 3.1). The weights, in combination with the biases, form the parameters adjusted during training to learn the desired function. Multilayer perceptrons also incorporate fully connected (FC) layers, where each node in one layer connects to every node in the next layer. These FC layers are particularly effective at capturing complex data relationships.

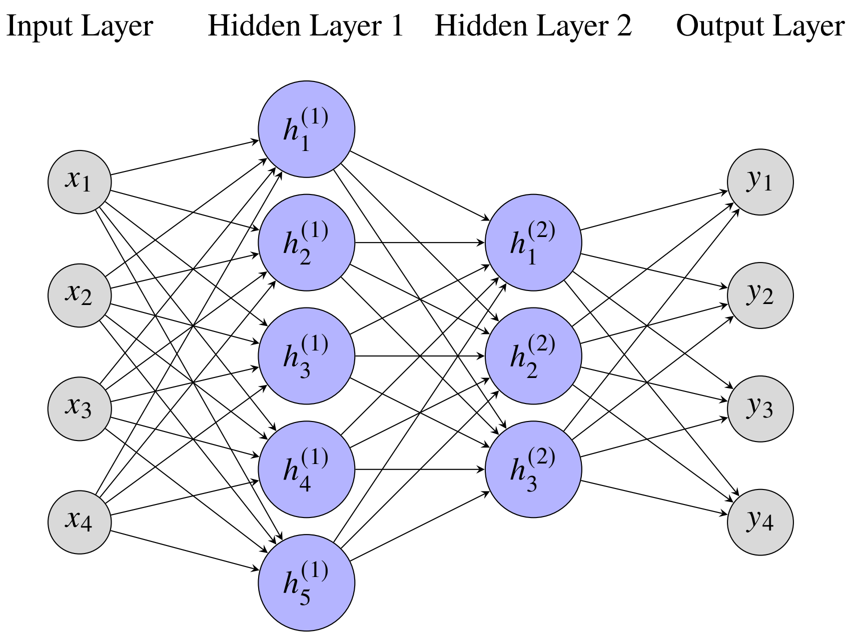


Figure 3.2 Feedforward Neural Network

Advancements in hardware have expanded the capacity to tackle more complex problems by adding layers, nodes, and parameters to neural network architectures. This development has given rise to the field of deep neural networks and deep learning. However, for specific data types like images, these architectures often demand a significant number of parameters, making them inefficient and challenging to train. To address this, various layers have been developed to reduce both parameters and complexity, with the convolutional layer emerging as one of the most successful solutions.

Convolutional Layer The convolutional layer [26, 27], shown in Figure 3.3, is a key component in neural networks. Its design exploits the properties of the input data and is especially suited for processing images. Unlike fully connected layers, they connect neurons to small sections of the previous layer, reducing the overall parameter complexity. Each convolutional layer contains learnable filters that are applied across the input features. Neurons in a convolutional layer are arranged in three dimensions: *width*, *height*, and *depth*. While width and height specify the filter dimensions, depth is determined by the input size. For example, a three-channel

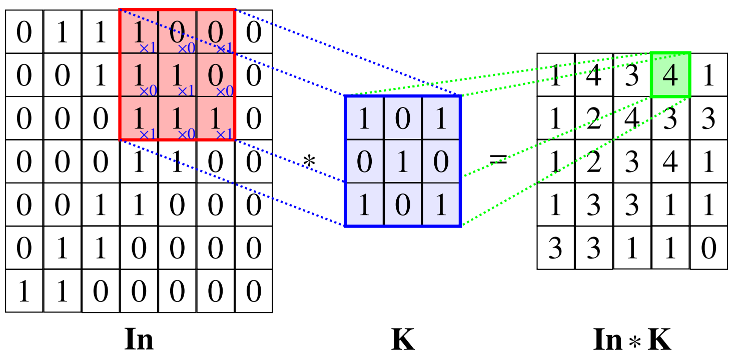


Figure 3.3 Convolutional layer

image typically requires a first convolutional layer with a depth of three. The nature of the input and the way the filter moves across its space defines three kinds of convolution: 1-dimensional (1D), 2D, or 3D. It is important to note that audio can be treated both as a sequence of frames or as an image, depending on which features are extracted (Fig. 2.2). As a result, both 1D and 2D convolution techniques can be effective.

Normalization Normalization is a crucial technique in deep learning, employed to standardize the input of neural network layers. This standardization enhances the efficiency and effectiveness of the learning process by ensuring that input features have consistent scales, thereby addressing issues like vanishing and exploding gradients. Common normalization techniques include batch normalization [28], instance normalization [29], and layer normalization [30]. While instance and layer normalization have gained prominence, particularly in attention and transformer models, our primary focus in this discussion is on batch normalization (often referred to as BN or batchnorm). Batch normalization plays a central role in our work and is widely recognized as a well-established normalization technique. To clarify, in this context, a 'batch' (denoted as $\mathcal{B}$) refers to the set of training samples x considered at any given forward step. For every value x the output is defined as:

$$y = \frac{x - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} * \gamma + \beta \tag{3.2}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ are respectively the mean and the variance of the batch, and ϵ is a small value added to the variance to avoid division by zero. Finally, the normalized values are scaled and shifted using the two learnable parameters γ and β.

Pooling Pooling is a technique extensively used in deep learning to reduce the dimensionality of outputs from convolutional layers in a neural network. Its purpose is to downsample feature maps by summarizing information within small regions.

The most common method of pooling is max pooling [31] (Fig. 3.4), where only the maximum value of a small region of the feature map is forwarded. This operation reduces the size of the feature map while preserving the most relevant information. Alternative pooling methods include average pooling, which computes the average value of the region, and L2-norm pooling, which forwards the L2-norm.

In the context of speaker verification, one of the simplest and most commonly used pooling strategies is to compute the mean and standard deviation along the time dimension. Additionally, attention mechanisms have gained popularity and are frequently integrated at this stage (Section 3.2.3).

Similar to pooling, embedding extraction aims at preserving the most important information contained within an input into a smaller, fixed representation. This similarity can be observed in modern approaches to speaker and language recognition, where embeddings are commonly extracted from a fully connected layer that follows the pooling operation.

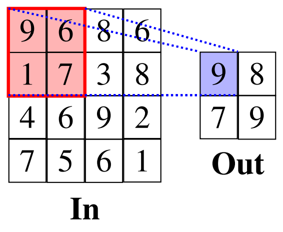


Figure 3.4 Max Pooling layer

3.1.2 Training

The function of a neural network can be expressed as $f(x;\Theta)$, where Θ denotes the network's parameters. The training process of a neural network revolves around the adjustment of these parameters to improve the accuracy of its prediction. Neural networks can be trained using various strategies, broadly divided into supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is a machine learning approach where the parameters of a function are trained using labeled examples. This drives the network to establish associations between input data and the corresponding output labels provided during training. In contrast, unsupervised learning aims at discovering latent patterns, structures, or representations within the data, without relying on predefined labels. When a model can only depend on a small portion of labeled data in relation to a significant amount of unlabeled data, this form of learning is referred to as semi-supervised learning. Conversely, when label information is derived from the input data itself, the training process is categorized as self-supervised learning. In the context of reinforcement learning, the model, referred to as an agent, must independently acquire the optimal strategy or policy through a process of trial-and-error. In this case, there is no need for labeled data, but rather the agent must interact with a simulated environment which provides positive or negative feedback based on its actions. This simulation is necessary for the agent to learn and improve its performance over time. In this work we will concentrate on supervised learning.

The distance between the output of the network function, given its current parameters and a target reference value is measured using a *loss function*. To compute the loss function, let's consider an input vector $x_p = (x_{0_p}, x_{1_p}, ..., x_{N_p})$ and the corresponding reference output vector $y_p = (y_{0_p}, y_{1_p}, ..., y_{M_p})$, where N and M represent the number of input and output units, respectively. The loss function is defined as follows:

$$J(\Theta) = \sum_i loss(f(x_p;\Theta), y_p) \tag{3.3}$$

In a neural network, the weights are updated using the gradient of the parameters. The process is iterative and at each iteration, the parameters are updated following

$$\Theta \leftarrow \Theta - \eta \frac{\delta J(\Theta)}{\delta \Theta} \tag{3.4}$$

where η is the learning rate, until the loss function reaches a minimum value.

Backpropagation Backpropagation [32] is an algorithm employed to minimize the loss function through gradient descent. This algorithm efficiently computes the gradient for every weight by using a recursive equation

$$\frac{\partial J(\Theta)}{\partial w^{l}_{j,k}} = \delta^{l}_{j} y^{(l-1)}_{k} \tag{3.5}$$

with

$$\delta^{(l)}_{i} = \begin{cases} \frac{\partial J(\Theta)}{\partial y^{(l)}_{j}} \cdot \sigma'_{l}(z^{(l)}_{j}) & \text{if } l \text{ is the output layer} \\ \left(\sum_{i=1}^{q} \delta j^{(l+1)} \cdot w^{(l+1)}_{i,j}\right) \cdot \sigma'_{l}(z^{(l)}_{j}) & \text{if } l \text{ is a hidden layer} \end{cases}$$

In this equation, $y^{(l)}_{j}$ represents the output of unit j in layer l, and q is the size of units in layer $l+1$. σ_l is the activation function of layer l and $z^{(l)}_{j} = \sum_{i=0}^{K} w^{(l)}_{i,j} \cdot y^{(l-1)}_{i}$ is the activation value from unit j in layer l. To compute these values, forward propagation needs to be calculated. Once the gradient is obtained, gradient descent can be applied to update the weights and minimize the loss function.

$$w^{(l)}_{j,k} \leftarrow w^{(l)}_{j,k} - \eta \cdot \frac{\partial J(\Theta)}{\partial w^{(l)}_{j,k}} \tag{3.6}$$

Optimizers The strategy adopted to update the weights of a neural network in order to minimize its loss is commonly known as *optimizer*. In gradient descent [33], the weights are updated by following the direction defined by the negative gradient. However, calculating the gradient for all training samples at the same time is not feasible. Instead, the weights are updated iteratively using a subset of the input data. The extreme case, when each iteration is done using only one sample at a time, is known as Stochastic Gradient Descent (SGD). To achieve a balance between the two methods, a generic number of samples higher than one is used, which is known as mini-batch gradient descent. This is the most commonly used approach in practice.

The **Natural Gradient Descent (NGD)** [34] is a second order optimizer defined as

$$\Theta = \Theta - \eta F^{-1} \frac{\delta J(\Theta)}{\delta \Theta} \tag{3.7}$$

where F is the Fisher matrix. However, the necessity of the Fisher matrix (or the Hessian) for second-order optimization makes this method impractical for deep learning models due to the large number of parameters. Nevertheless, these matrices can effectively be approximated and in some of our testing we have used the NGD with the Fisher matrix approximation defined in [35].

To speed up gradient descent and prevent the model from converging to local minima a **momentum** is used in the update rule [36]. At each step, Θ_t is updated keeping track of the previous update Θ_{t-1}, thus introducing inertia to the current update

$$\Theta_t \leftarrow \Theta_{t-1} - \eta \frac{\delta J(\Theta)_t}{\delta \Theta} - \gamma \frac{\delta J(\Theta)_{t-1}}{\delta \Theta} \tag{3.8}$$

where γ is the momentum term.
Some methods, like [37, 38], have introduced an adaptive learning rate strategy, where η is divided by a different quantity related to the previous update for each parameter. For instance, in [39], a running average E_t was recursively defined

$$E_t = \gamma E_{t-1} + (1-\gamma)\left(\frac{\delta J(\Theta)_t}{\delta \Theta}\right)^2 \tag{3.9}$$

with γ playing a similar role to the momentum term. The updated values are now obtained as

$$\Theta_t \leftarrow \Theta_{t-1} - \frac{\eta}{\sqrt{E_t + \epsilon}} \frac{\delta J(\Theta)_t}{\delta \Theta} \tag{3.10}$$

where ϵ is a small constant to avoid zero division. **Adaptive moment estimation (Adam)** [40] combines the adaptive learning rates with the momentum. Given the gradient $g = \frac{\partial J(\Theta)}{\partial \Theta}$, the two moments m_t and E_t are initialized to zero and at each iteration computed as

$$m_t = \frac{\beta_1 * m_{t-1} + (1-\beta_1) * \frac{\delta J(\Theta)_t}{\delta \Theta}}{1-\beta_1^t} \tag{3.11}$$

$$E_t = \frac{\beta_2 * E_{t-1} + (1-\beta_2) * \left(\frac{\delta J(\Theta)_t}{\delta \Theta}\right)^2}{1-\beta_2^t} \tag{3.12}$$

where β_1 and β_2 are fixed values. It is important to note that the second moment E_t functions as an approximation of the Fisher/Hessian matrix, constrained to be diagonal. The update rule is then defined as follows

$$\Theta_t \leftarrow \Theta_{t-1} - \eta \frac{m_t}{\sqrt{E_t + \epsilon}} \tag{3.13}$$

Schedulers Choosing the correct learning rate is a critical factor in the success of an optimizer. If the learning rate η is too large, the training process can become unstable and diverge. On the other hand, if the learning rate is too small, the convergence can be slow or the process can get stuck in a suboptimal local minimum. Generally, the training begins with a learning rate that is gradually decreased when the loss reaches a plateau. Learning rate schedulers are employed to follow certain functions or rules to achieve a specific trend for η, such as a linear or exponential decrease (Section 5.2).

Loss Function In the context of classification, the choice of a loss function depends on the specific problem at hand. For our task, which involves classifying speakers or languages, we employ the cross-entropy (CE) loss function [3]. CE measures the dissimilarity between two distributions, which implies that our network's output needs to represent a distribution. To achieve this, the *softmax* activation function is typically employed in the output layer of neural networks. This function maps the network's output values into a probability distribution, which is then used to compute the cross-entropy loss. In a scenario with K classes and an output vector $z = [z_1, ..., z_K]^T$, the softmax function is defined as follows

$$\sigma_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.14}$$

The loss for a set of N samples can then be expressed as:

$$L_s = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{f_{y_i}}}{\sum_{j=1}^{C} e^{f_j}} \tag{3.15}$$

where y_i is the target label, N and C are the number of training samples and classes and f_j is the output of the last fully-connected layer.

While softmax and cross-entropy loss excel in classification tasks, they do not guarantee well-separated and discriminative embeddings for previously unseen speakers. This limits their ability to generalize effectively. The ability to discriminate unseen speakers can be improved by forcing samples belonging to the same class to be projected closer together and sufficiently far from samples of other classes in the embedding space. This has been addressed by introducing a *margin* into the loss function.

Angular Softmax The loss function, as defined in Equation (3.15), can be reformulated to introduce the concept of separation margin. Given the function of the last fully connected layer $f_{i,j} = W_j^T x_i + b_j$ and $f_{i,y_i} = W_{y_i}^T x_i + b_{y_i}$, where x_i is the i-th training sample and, W_j and W_{y_i} are the j-th and y_i-th column of the weight matrix W, (3.15) can be rewritten as:

$$\begin{aligned} L &= -\frac{1}{N}\sum_i^N \log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_j e^{W_j^T x_i + b_j}}\right) \\ &= -\frac{1}{N}\sum_i^N \log\left(\frac{e^{\|W_{y_i}\|\|x_i\|\cos(\theta_{y_i,i})+b_{y_i}}}{\sum_j e^{\|W_j\|\|x_i\|\cos(\theta_{j,i})+b_j}}\right) \end{aligned} \tag{3.16}$$

in which $0 \leq \theta_{j,i} \leq \pi$ denotes the angle between W_j and x_i. For the binary case, the separation rule defined by (3.16) is $(W_1 - W_2)x + b_1 - b_2 = 0$, assuming $\|W_i\| = 1, b_i = 0$ the derived decision boundary is $\cos(\theta_1) - \cos(\theta_2) = 0$. A sample x belonging to class 1 is correctly classified if $\cos(\theta_1) > \cos(\theta_2)$. In [41] the authors suggest to impose a more stricter boundary: $\cos(m\theta_1) > \cos(\theta_2)$, with $m > 2$, resulting in:

$$L_{ang} = \frac{1}{N}\sum -log\left(\frac{e^{\|x_i\|\cos(m\theta_{y_i,i})}}{e^{\|x_i\|\cos(m\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|x_i\|cos(\theta_{j,i})}}\right) \tag{3.17}$$

In (3.17), $\theta_{y_i,i}$ is constrained to be in the range of $[0, \frac{\pi}{m}]$. In order to make the function optimizable and remove this constrain, $\cos(m\theta_{y_i,i})$ is replaced by a monotonically decreasing angle function $\phi(\theta_{y_i,i})$.

In [42] the authors proposed to use $\phi = \cos(\theta) - m$, defining the Additive Margin or CosFace loss. Thus (3.17) becomes:

$$L_{ams} = -\frac{1}{N}\sum_{i=1}^N \log\left(\frac{e^{s\cdot(\cos\theta_{y_i,i}-m)}}{e^{s\cdot(\cos\theta_{y_i,i}-m)} + \sum_{j\neq y_i} e^{s\cdot\cos\theta_{i,j}}}\right) \tag{3.18}$$

where s is a scaling factor, that can be either learnable or fixed.

Additive Angular Margin ArcFace or Additive Angular Margin has been proposed in [43] and quickly adopted also in speaker verification. In this formulation ϕ is defined as $\phi(\theta_{y_i,i}) = \cos(\theta_{y_i,i} + m)$. A unified loss can then be defined as:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s[\cos(m_1\theta_{y_i,i}+m_2)-m_3]}}{e^{s[\cos(m_1\theta_{y_i,i}+m_2)-m_3]} + \sum_{j\neq y_i} e^{s\cdot\cos\theta_{i,j}}} \tag{3.19}$$

The definition of the function ϕ and the choice of the margin can play a crucial role [44]. Various approaches have been proposed, with some methods, such as *elasticFace* [45] and *Dyn-ArcFace* [46] that relax the fixed definition of the additive margin. Promising results have been achieved with the *subcenter ArcFace* [47] and *InterTop-K* [48]. In this case, for each class, K positive sub-centers are defined. This allows ArcFace to find one dominant sub-class representing the majority of the examples and non-dominant sub-classes that will include hard or noisy examples. The goal is to obtain a more robust network against new examples and noisy conditions. In our experiments, we have evaluated different solutions. Our results and some comparisons will be presented in Chapter 5.

3.2 Speaker and language recognition architectures

In the following sections, we will present the architectures that we have employed for embedding extraction.

In our work, we utilized neural networks trained for speaker or language classification. The general network structure can be broken down into three primary components, as illustrated in Figure 3.5. The first component works at the frame level, analyzing individual utterance frames to capture short-term temporal relationships. The second component, the pooling layer, summarizes this information and generates a fixed-length output. Finally, the third component operates at the segment level, capturing long-term relationships by considering information spanning the entire utterance. Typically, the embeddings are obtained from the segment level.

The architecture of these components is shaped by several factors, including which activation functions are used, the number of neurons in each layer, and the overall number of layers. Although a large variety of neural network architectures is

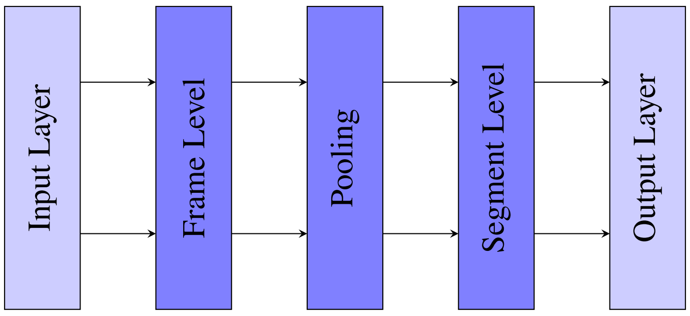


Figure 3.5 Overview of the main components that make up an architecture used in speaker recognition.

available, in this work we focused on a limited set of architectures, mainly due to the temporal nature of the data we are analyzing.

The primary objective of the network is to create a representation of the speaker based on their speech. To achieve this, the network has to effectively handle both short-term and long-term acoustic dependencies, capturing the key characteristics of the speaker's voice.

In the following sections, we will review well-known architectures in speaker verification, as well as our own solutions for language recognition. More details on their implementation will be provided in Chapter 5.

3.2.1 Time Delay Neural Network

To effectively capture the nonlinearity present in the data, a feedforward network needs to be sufficiently complex. Specifically, when dealing with speech data, the network must be able to represent and capture the time-dependent relationship between spectral coefficients and higher-level features. Additionally, it should be able to learn time-invariant abstractions, or features, from the data, while keeping the number of parameters small compared to the training set dimension. To satisfy these requirements, the Time Delay Neural Network (TDNN) was introduced [49]. In the TDNN, a delay (D_i) is applied to each input, and the units at each layer receive input not only from the layer below but also from their time-delayed outputs. Differently from a classic unit, Figure 3.1, the TDNN unit takes in a moving window $[x_{t-m},\dots,x_t,\dots,x_{t+n}]$ centered around the input feature x_t, Figure 3.6. Each neuron processes the same features within different windows using different sets of weights.

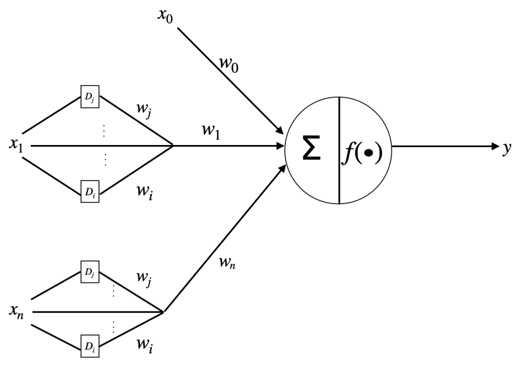


Figure 3.6 TDNNs incorporate temporal context for each input x by additionally analyzing neighboring inputs denoted as $[D_i, ..., D_j]$

.

Variants To improve the performance and quality of the speaker embeddings extracted from the classic TDNN, several alternatives have been developed. Some of these alternatives have been utilized in our work:

- Extended Time Delay Neural Network: the Extended-TDNN (ETDNN) was described in [50]. The *frame level* layers process speech frames with a narrow temporal context that is centered on the current frame t. In ETDNN, the time-delayed layers are interleaved with fully connected layers (FNN). The pooling layer combines the frame level representation computing a set of statistics, such as mean and standard deviation, for each input segment. Speaker embeddings are extracted from the two fully connected layers at the segment level.

- Factorized Time Delay Neural Network: In [51] the weights of the layers at the frame level are forced to be semi-orthogonal. The goal of this approach is the reduction of the number of parameters using Singular Value Decomposition (SVD). Given the parameter matrix M, the condition to enforce is $P = MM^T = I$. The function to minimize is $f = tr(QQT)$, where Q is defined as $Q \equiv P - I$. The update of the weight is then computed as

$$M \leftarrow M - \frac{1}{2\alpha}(MM^T - \alpha^2 I)M \tag{3.20}$$

In the basic case $\alpha = 1$, however, if the objective is to constrain M as a scaled semi-orthogonal matrix, α can assume any value. In our work, we utilized the *floating case* [51] where $\alpha = \sqrt{tr(PP^T)/tr(P)}$ and, again, $P \equiv MM^T$. Another important addition to this architecture is the presence of skip connections, where the input of a layer is concatenated with the output of an early layer.

- Dense Time Delay Neural Network: The need to reduce the number of parameters was explored in [52]. The core unit of the DTNN is an FNN bottleneck layer followed by a TDNN layer, with a skip connection linking the input and the output of the unit. The authors also proposed a multi-branch architecture, where the TDNN layers of each branch have different contexts.

3.2.2 ResNet

In general, increasing the number of layers produces better results. However, this practice also introduces certain challenges that can negatively impact the ability to train such architectures. Deep residual learning [53] was introduced to address this problem. In this approach, residual connections are utilized to establish a direct link between the input and the output of specific layers, effectively bypassing some intermediate layers, (Fig. 3.7). This design allows gradients to flow through the network without being diminished by the activation function, mitigating the vanishing gradient problem [53].

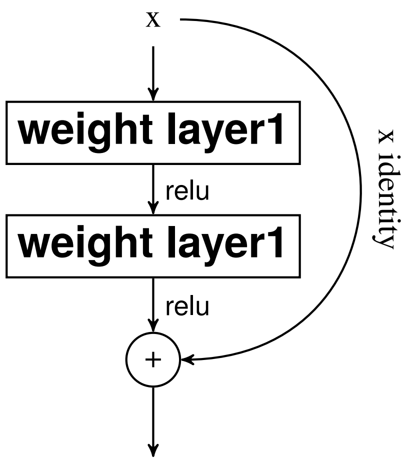


Figure 3.7 Single residual block

TDNN and its derivation were designed to work on audio as a sequence of frames/frequencies. However, in practice, features extracted from audio can also

Table 3.1 Modified ResNet34 for speaker verification from [55]

Layer name	Structure	Output
Input	-	$1\times N\times T$
Conv2D	3×3, stride-1	$128\times N\times T$
ResBlock-1	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times3, stride=1$	$128\times N\times T$
ResBlock-2a	$\begin{bmatrix}3\times3,128\\3\times3,128\\1\times1,128\end{bmatrix}\times1, stride=2$	$128\times N/2\times T/2$
ResBlock-2b	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times3, stride=1$	$128\times N/2\times T/2$
ResBlock-3a	$\begin{bmatrix}3\times3,256\\3\times3,256\\1\times1,256\end{bmatrix}\times1, stride=2$	$256\times N/4\times T/4$
ResBlock-3b	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times5, stride=1$	$256\times N/4\times T/4$
ResBlock-4a	$\begin{bmatrix}3\times3,256\\3\times3,256\\1\times1,256\end{bmatrix}\times1, stride=1$	$256\times N/8\times T/8$
ResBlock-4b	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times2, stride=1$	$256\times N/8\times T/8$
Flatten(C, F)	-	$(256*N/8)\times T/8$
StatsPooling	-	$(256*N/8)*2$
Dense	-	256
FC		#Speakers

be treated as images (Fig. 2.2), and 2D convolution-based ResNets have been successfully used for speaker and language recognition [54–56].

Modified ResNet34 The ResNet34 architecture, initially introduced in [53], consists of 34 layers organized into residual blocks. Originally designed for image processing, the architecture has been adapted to better handle audio features. Specifically, in [55] the authors proposed to enhance its audio processing capabilities by increasing the number of channels in the initial layers while decreasing the expansion rate in deeper layers as shown in Table 3.1. This proved to be more effective than the classic architecture.

Res2Net The Res2Net layer is designed to enhance the representation capacity of deep neural networks by allowing the integration of multi-scale features at different resolutions. The Res2Net [57] layer is constructed by dividing the input feature map into multiple sub-maps. Each submap undergoes independent convolutional operations, and the results are then combined and further processed through additional convolutional layers to produce the final output. This approach allows Res2Net layers to capture multi-scale information without requiring excessive computational resources. While not directly utilized in our work, it remains one of the key components of the ECAPA-TDNN architecture (Section 3.2.4).

3.2.3 Attention

Attention is a mechanism that allows neural networks to focus on specific parts of the input data that are automatically supposed most relevant for a given task. This is achieved by assigning specific weights to different parts of the input. The weights are learned during training and are based on the importance of each feature to the output of the model.

Multi-head self-attention The Multi-head self-attention (MSA) [58] is the core component of the Transformer architecture. While CNNs are able to extract local features, MSA focuses on learning the relationships between different positions in a sequence, even when there is a high temporal distance between them. Given a sequence X with shape $T \times D$, self-attention is computed over H independent heads, each with dimensionality $D_h = D/H$. MSA defines three FC layers, with weight $W_i^{(q)}$, $W_i^{(k)}$ and $W_i^{(v)}$ with shape $D \times D_h$. These weights are used to obtain three different representations of the input: query Q, key K and value V

$$Q_i = XW_i^{(q)} \qquad K_i = XW_i^{(k)} \qquad V_i = XW_i^{(v)} \quad \forall i \in 1,...,H \tag{3.21}$$

Self-attention of each head is then concatenated and linerly projected to obtain the MSA

$$Y = MSA(X) = [A_1, A_2, ..., A_H]W_y \tag{3.22}$$

where W_y has shape $D \times D$ and

$$A_i = softmax\left(\frac{Q_i K_i^T}{\sqrt{D_h}}\right) V_i \tag{3.23}$$

Attentive Pooling Attention has found a prominent application in speaker recognition, particularly in the pooling layer. Researchers have utilized a soft self-attention technique to derive weighted statistics from the temporal pooling layer [59]. The statistics pooling layer, which takes frame-level features $o_t(t = 1, ..., t = T)$ as input, calculates the mean vector μ and the second-order standard deviation σ as:

$$\mu = \frac{1}{T}\sum_{t}^{T} o_t \tag{3.24}$$

$$\sigma = \sqrt{\frac{1}{T}\sum_{t}^{T} o_t \odot o_t - \mu \odot \mu_t} \tag{3.25}$$

It is reasonable to assume that some frames are naturally more informative than others. For example, frames containing silence or noise may not provide valuable speaker characteristics. Attention automatically evaluates the significance of each frame by generating a scalar score $e_t = v^T f(Wo_t + b) + k$ for each frame-level feature. Here, $f(\cdot)$ denotes a non-linear activation function. The score is then normalized using the softmax function.

$$\alpha_t = \frac{\exp(e_t)}{\sum_{\tau}^{T} \exp(e_\tau)} \tag{3.26}$$

The weighted mean and standard deviation are obtained:

$$\tilde{\mu} = \frac{1}{T}\sum_{t}^{T} \alpha_t o_t \tag{3.27}$$

$$\tilde{\sigma} = \sqrt{\sum_{t}^{T} \alpha_t o_t \odot o_t - \mu \odot \mu_t} \tag{3.28}$$

Numerous extensions have been introduced for attentive pooling, with the primary objective of training distinct sets of weights for different subsets of features. A more comprehensive formulation was shown in [48]. In this approach, the frame-level features $H = [h_1, h_2, ..., h_T], h_t \in \mathbb{R}^d$ can be splitted in H heads. Each head, $o_t = [o_t^1, o_t^2, ..., o_t^H], o_t^h \in \mathbb{R}^{d/H}$, has Q trainable queries. The scalar score e_t^h can then be calculated as:

$$e_t^h = \begin{cases} (o_t^h)^T w_a & \text{n=1} \\ f((o_t^h)^T w_b) w_c & \text{n=2} \end{cases} \tag{3.29}$$

where w_a, w_b and w_c are the weight matrixes of size $d_h \times d_s$, $d_h \times d_k$ and $d_k \times d_s$ respectively. The scores are then normalized:

$$\alpha_{t,h}^{q} = \frac{\exp(e_t^h)}{\sum_{\tau}^{T} \exp(e_\tau^h)} \tag{3.30}$$

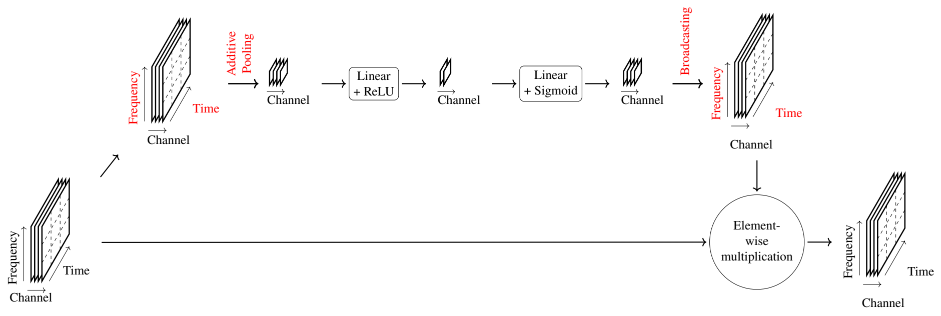
The weighted mean and standard deviation are now obtained as concatenation $E_\mu = [\hat{\mu}_1, \hat{\mu}_2, ..., \hat{\mu}_H]$ where $\hat{\mu}_h = [\mu_h^1, \mu_h^2, ..., \mu_h^Q]$, with:

$$\mu_h^q = \frac{1}{T} \sum_{i}^{T} \alpha_{t,h}^{q} \cdot (o_i^t)^T \tag{3.31}$$

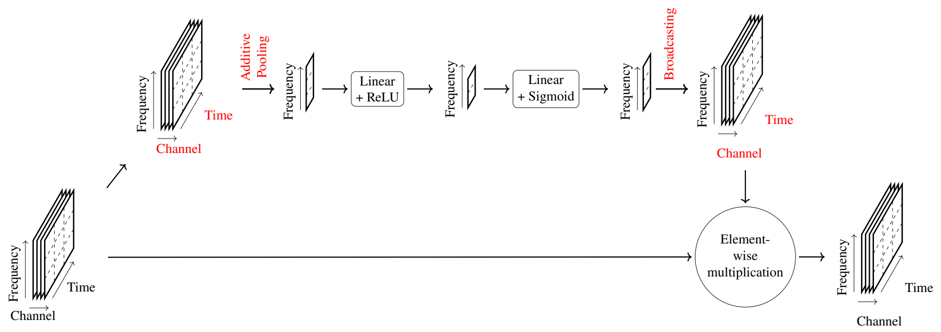
$$\sigma_h^q = \sqrt{\sum_{i}^{T} \alpha_{t,h}^{q} \cdot (o_i^h)^T \cdot (o_i^h)^T - \mu_h^q \cdot \mu_h^q} \tag{3.32}$$

Standard deviation E_σ can be obtained in the same way with σ_h^q. We can notice that (3.27) and (3.28) can be obtained from (3.31) and (3.32) with $(H = 1, Q = 1, n = 2, d_s = 1)$, while the case with $(H > 1, Q > 1, n = 2, d_s = d_h)$ is called Multi-Query Multi-Head Attention (MQMHA) [48].

Squeeze&Excitation A similar approach has been taken with the Squeeze & Excitation (SE) layer [60], which adaptively recalibrates channel-wise feature responses. The SE layer can be added to existing neural network architectures, enhancing their performance. The SE layer (Fig. 3.8a) operates in two steps: it starts with a squeeze operation that compresses one (or more) dimensions of the feature map into a channel descriptor. Then, it performs an excitation operation, which learns a set of weights to selectively amplify or suppress individual channels within the feature map based on their importance. This design allows the SE layer to capture interdependencies between the channels of a feature map. Moreover, the SE layer has been modified to better suit the characteristics of audio data. In [61, 62] the squeeze operation was applied either along the frequency or the time axis, resulting in the frequency-wise SE (Fig. 3.8b) or temporal-wise SE, respectively. In [63] a composite approach is introduced, involving two separate squeeze operations along both the frequency and time axes, which generate weighted maps that are then combined, as illustrated in the figure.



(a) Squeeze & Excitation block

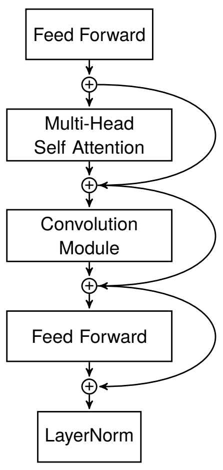


(b) Frequency Wise Squeeze & Excitation block

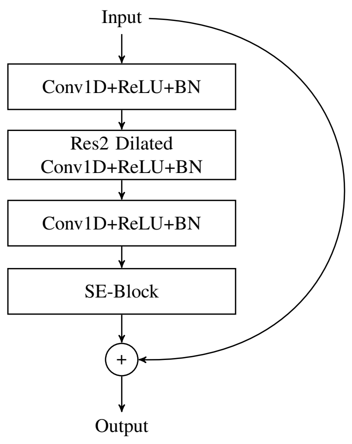
Conformer Convolutional Neural Networks (CNNs) are proficient in local spatial modeling but struggle when dealing with long-range dependencies. This limitation prompted the widespread adoption of Transformer-based architectures for sequence processing tasks. The Conformer is a powerful architecture that combines CNNs for local features and transformer layers for global context, which can be used to obtain a more robust speaker representation. The Conformer [64] block relies on multi-head self-attention (MHSA) and convolution modules as its key components. MHSA incorporates a relative positional encoding scheme from Transformer-XL [65], making it adaptable to inputs of varying lengths. The convolutional operations are implemented by pointwise and 1D depthwise convolution layer [66], and Batch-Norm. Figure 3.9a shows the structure of a Conformer block.

3.2.4 ECAPA-TDNN

In [67], the Emphasized Channel Attention, Propagation and Aggregation in TDNN (ECAPA-TDNN) neural network was introduced. It combines some of the solutions



(a) Conformer Block



(b) ECAPA Res2Net block [67]

Figure 3.9 Building blocks of the Conformer and ECAPA architectures.

presented in the previous paragraphs. The SE-Res2Net block (Fig. 3.9b) is the fundamental component of the architecture. The network structure includes a TDNN block, followed by multiple SE-Res2Net blocks. These blocks merge the 1-dimensional convolution, the residual architecture of a Res2Net layer, and the time-wise Squeeze&Excitation operation. The output of the SE-Res2Net is then concatenated. The statistical pooling is replaced by channel- and context-dependent pooling, extending the concept introduced in equations (3.27) and (3.28) for attentive pooling.

DepthFirst architectures The advent of transformer-based architectures, such as Vision Transformer (ViT), has paved the way for a new state-of-the-art in image classification. However, these architectures have only achieved state-of-the-art performance when they are combined with some of the convolution-based architecture priors in the more general vision field. This inspired the authors of [68] to review the ConvNet architecture and incorporate some of the solutions and concepts developed for the new transformer-based architecture. The same approach was applied to speaker verification-oriented architectures like ResNet and ECAPA-TDNN, [69]. In various tests, it was found that the depth of a network was the most discriminant feature for a network producing good results. Therefore, a depth-first approach was proposed, which involves modifying the usual components to significantly increase the network's depth while keeping the complexity and number of parameters tractable.

- DF-ResNet: in the depth-first ResNet, the basic block is replaced with a bottleneck block [70], which increases the number of parameters. However, this does not improve the performance. To reduce the number of parameters, the 2D-convolutional layer is replaced with a depthwise 2D-convolution, in which each input channel is convolved with a different kernel. A further reduction is achieved by inverting the dimension of the layers in the block, [71]. Following the approach in [68], a separate subsampling is applied. The number of parameters is greatly reduced at this stage, which allows for an increase in the number of layers in each stage, resulting in a deeper architecture with similar complexity as the starting one and better performance.
- DF-ECAPA: a similar approach is suggested for the ECAPA-TDNN architecture. The dilated convolutions utilized in the Res2Net block are replaced with 2D convolution with a large kernel set to 5. The number of channels at each stage is then halved, making the design closer to a ResNet and greatly reducing the number of parameters. At this point, the number of layers in each block can be increased.

3.2.5 Neural Networks for Language Recognition

In this section, we will discuss the architectural approaches for language recognition. Building upon the success of speaker verification architectures in language-related tasks [72, 73]. From the models introduced earlier, we outline the solutions developed to enhance the performance of such systems. We introduce the CNN-ECAPA [61] for language recognition. Inspired by the results we achieved with it, we extended similar modifications to other architectures. The effectiveness of these solutions was assessed in the NIST Language Recognition Evaluation 2022 (LRE22). We present the detailed results in Section 5.3.1.

CNN-ECAPA

We initially explored the ECAPA architecture to tackle the language detection problem. This architecture has been shown to be highly effective in speaker verification and has quickly become one of the state-of-the-art architectures. We obtained acceptable results also for the language recognition task. In order to improve its performance,

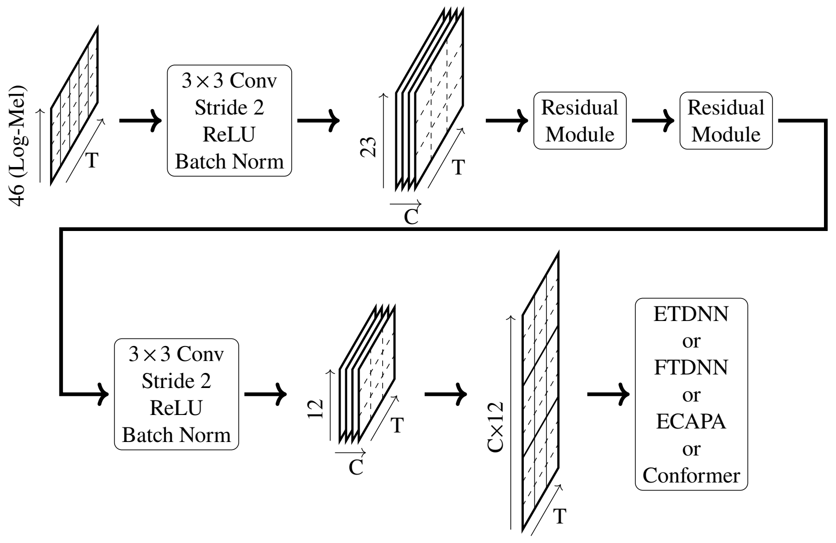


Figure 3.10 CNN block

we experimented with the modification proposed in [61]. This change involves the addition of 2D CNN layers before the input layer of the original architecture, which has been shown to enhance ECAPA's performance in speaker verification. The CNN block illustrated in Figure 3.10 involves several layers, including a 2D convolutional layer, two 2D convolutional residual blocks, and a final 2D convolutional layer. The block's output dimensions are then reduced to one dimension by collapsing the channel and frequency dimensions. It is worth noting that the ECAPA architecture predominantly employs 1D convolution, following the TDNN strategy. The inclusion of 2D convolution to process the input helps the network to reveal more complex relationships in the succeeding stages.

CNN-TDNN

We also investigated other TDNN-based designs to tackle the language evaluation challenge, specifically ETDNN and FTDNN. Both ETDNN and FTDNN, like ECAPA, have been widely utilized in the domain of speaker verification. We extended these architectures by introducing the same 2D CNN block (Fig. 3.10) to the front of both architectures, which resulted in significant improvements in the language evaluation task.

ResNet and Conformer

Finally, we explored the ResNet34 (Section 3.2.2) and Conformer (Section 3.2.3) architectures. In the case of the conformer, we also experimented with the CNN block, which led to better results compared to the standard subsampling architecture. However, for the final system, even better results were achieved by utilizing a 1-dimensional subsampling layer. This subsampling layer draws inspiration from TDNN and is implemented using 1D convolution, as opposed to the 2D implementation depicted in Figure 3.10.

These modules and architectures have been utilized as speaker embedding extractors for our backend classifier (Section 5.1) and have played a pivotal role in our submission to the language recognition evaluation organized by NIST [74]. In Chapter 5 we provide details on the implementation and training strategies we followed, and in Section 5.3.4 we will analyze our language recognition model.

Chapter 4

Backends

Neural networks are trained to classify a closed and known set of speakers. The ability to extract meaningful embeddings for new and unseen speakers is the main goal of the speaker verification task. To obtain a model that can generalize speaker characteristics, networks are trained with a large number of speakers. Nevertheless, there is often a discrepancy between the data used for training and the new testing data. This difference can lead to suboptimal performance in the subsequent steps. In this chapter, we present the theory behind the currently utilized scoring methods and explore solutions to address the issue of data mismatch.

4.1 Scoring

In speaker verification, the objective is to determine whether two spoken utterances originate from the same identity. This process involves employing some sort of scoring function to assess the similarity between two or more embeddings. This can be accomplished through simple similarity measures, often using distance-based methods like cosine similarity. Alternatively, more advanced techniques can be employed, such as Probabilistic LDA and Pairwise Support Vector Machines. Some of these methods try to build a probabilistic model for each speaker, enabling the computation of log-likelihood ratios (LLR). However, to address the mismatch between training and testing conditions, further steps are often necessary before and after scoring. Furthermore, certain scoring functions rely on specific assumptions about the properties of the embeddings. To improve their performance, preprocessing and normalization techniques become necessary to align the embeddings with such assumptions.

4.1.1 PLDA

Historically, dimensionality reduction techniques such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) were commonly used as feature extraction methods for object recognition. However, these methods would only extract the features and require additional steps to perform the actual recognition task. By removing dimensions associated with high intra-speaker variability, LDA is a long-standing method used to perform dimensionality reduction. Defining the within–class, S_W, and between-class, S_B, covariance matrices as:

$$\begin{aligned} S_W &= \sum_{k=1}^{K} \sum_{i|x_i \in C_k} (x_i - \mu_k)(x_i - \mu_k)^T \\ S_B &= \sum_{k=1}^{K} N_k(\mu_k - \mu)(\mu_k - \mu)^T \end{aligned} \tag{4.1}$$

where x_i is a D-dimensional vector that belong to K classes C_k, μ is total mean, μ_k and N_k are the mean and number of samples for class k, LDA search for a linear transformation

$$y_i = W^T x_i \tag{4.2}$$

where W is a $D' \times D$ matrix, with $D' < K - 1$, which maximize the Fisher criterion:

$$J(W) = tr[(WS_w^{-1}W^T)(WS_BW^T)] \tag{4.3}$$

The right eigenvectors of $S_W^{-1}\ S_B$ corresponding to the D' largest eigenvalues provide the solution for W. LDA projects the speaker vector into a smaller space trying to minimize the intra-class variability and maximize the between-class variability. However, the reduced data still needs to be classified. Probabilistic LDA (PLDA) was introduced in [75] as an effective technique for solving this classification problem. Currently, PLDA and the models derived from it are considered the state-of-the-art approach in speaker verification.

PLDA assumes that it is possible to break down an embedding as:

$$x_{ij} = \mu + Uh_i + Vw_{ij} + \epsilon_{ij} \tag{4.4}$$

where μ is the global mean of the dataset, U contains the basis for the between-speaker subspace and h_i is a hidden variable that is assumed to be shared by embeddings belonging to the same speaker. The channel effects of a specific audio segment j, from speaker i, are described by w_{ij}. The matrix V contains the basis for the within-speaker subspace. Finally, ϵ_{ij} describes the residual information and noise. The first and most widely adopted model is the Gaussian PLDA (GPLDA), which assumes a Gaussian distribution for the latent variables

$$\begin{aligned} h_i &\sim \mathcal{N}(0, I) \\ w_{ij} &\sim \mathcal{N}(0, I) \\ \epsilon_{ij} &\sim \mathcal{N}(0, \Lambda^{-1}) \end{aligned}$$

with Λ being diagonal. A more complex model was introduced in [76], known as the Heavy-Tailed PLDA (HTPLDA). In this formulation, the priors for the latent variables are assumed to follow a Student's t distribution

$$\begin{aligned} h_i &\sim \mathcal{N}(0, u_i^{-1} I) & u_i &\sim \mathcal{G}(n_i/2, n_i/2) \\ w_{ij} &\sim \mathcal{N}(0, u_{ij}^{-1} I) & u_{ij} &\sim \mathcal{G}(n_{ij}/2, n_{ij}/2) \\ \epsilon_{ij} &\sim \mathcal{N}(0, v_{ij}^{-1} \Lambda^{-1}) & v_{ij} &\sim \mathcal{G}(\nu/2, \nu/2) \end{aligned} \tag{4.5}$$

HTPLDA consistently outperforms GPLDA when applied to non-processed embeddings. However, GPLDA can achieve similar performance when appropriate preprocessing and normalization techniques are applied. Finally, given the significant computational complexity associated with HTPLDA, our primary focus in this work is on GPLDA.

Two Covariance Model

Assuming that both the speaker and channel subspaces are full rank simplifies the model, leading to the two-covariance model [77]. This simplified model is fundamental to our work on score calibration and normalization. We assume that an embedding is generated by a random variable $\mathbf{\Phi}$

$$\mathbf{\Phi} = \overline{\mathbf{Y}} + \overline{\mathbf{E}} \tag{4.6}$$

where $\bar{Y}$ represent the speaker identity and $\bar{E}$ embed the residual noise. The embedding ϕ can be seen as the sum between the speaker model y_s and some Gaussian noise z_r

$$\phi = y_s + z_r \tag{4.7}$$

The prior distribution is assumed to be Gaussian:

$$P(y) \sim \mathcal{N}(y|\mu, B^{-1}) \qquad\qquad P(\phi|y) \sim \mathcal{N}(\phi|y, W^{-1}) \tag{4.8}$$

where B^{-1} and W^{-1} denotes the between and within class coveariance matrices, while μ is the global mean of the dataset. It can be noted that the left prior in (4.8) is conjugate to the likelihood on the right. Given a set of embeddings $\phi_1, ..., \phi_n$ extracted from audio of the same speaker, we can compute the posterior for y in closed form. The result is a normal posterior

$$P(y|\phi_1, ..., \phi_n) \sim \mathcal{N}(y|L^{-1}\gamma, L^{-1}) \tag{4.9}$$

with

$$\begin{aligned} \gamma &= B\mu + W\sum_{i=1}^{n}\phi_i \\ L &= B + nW \end{aligned} \tag{4.10}$$

Speaker Verification Log-Likelihood Ratio

We can use these models to infer the speaker's identity. Given an enrolled speaker s_e and a test speaker s_t, a set of their embeddings $\phi_{e1}, ..., \phi_{en}$ and $\phi_{t1}, ..., \phi_{tm}$ respectively, we can compute the likelihood of the observed embeddings under the same speaker, H_s, and different speaker, H_d, hypothesis [76, 78] to find if the test and enrollment utterances belong to the same speaker. This is represented by the log-likelihood ratio (LLR)

$$l = \log \frac{P(\phi_{e_1}, ..., \phi_{e_n}, \phi_{t_1}, ..., \phi_{t_m}|H_s)}{P(\phi_{e_1}, ..., \phi_{e_n}, \phi_{t_1}, ..., \phi_{t_m}|H_d)} \tag{4.11}$$

Expressing $P(\phi_1,...,\phi_k|H_s)$ as:

$$P(\phi_1,...,\phi_k|H_s) = \frac{P(\phi_1,...,\phi_k|\mathbf{y}_0)P(\mathbf{y}_0)}{P(\mathbf{y}_0|\phi_1,...,\phi_k)} \tag{4.12}$$

and defining

$$Q(\phi_1,...,\phi_k) = \frac{P(\mathbf{y}_0)}{P(\mathbf{y}_0|\phi_1,...,\phi_k)} \tag{4.13}$$

where $\mathbf{y}_0$ is *any* value of the hidden variable that prevents the denominator from being zero, allows to reformulate the log-likelihood ratio as:

$$l = \log \frac{Q(\phi_{e_1},...,\phi_{e_n},\phi_{t_1},...,\phi_{t_m})}{Q(\phi_{e_1},...,\phi_{e_n})Q(\phi_{t_1},...,\phi_{t_m})} \tag{4.14}$$

A closed form for the LLR can be derived once we fix a value for $\mathbf{y}_0$

$$\log Q(\phi_1,...,\phi_k) = \frac{1}{2}(\log|B| - \mu^T B\mu - \log|L_S| + \gamma_S^T L_S^{-1}\gamma_S) \tag{4.15}$$

here the set of embeddings $\phi_1,...,\phi_k$ is denoted by S, and L_S and γ_S are defined as in (4.10) for the posterior distribution of $\mathbf{y}$ give the embeddings in S [78].

4.1.2 PSVM

In the case of one embedding per speaker, an alternative to PLDA is the Pairwise Support Vector Machine (PSVM) [79, 80], which belongs to the class of discriminative methods. Discriminative models search for a decision rule that optimizes a criterion for a given set of training patterns, given an enrollment embedding ϕ_e and a test embedding ϕ_t. From the two-covariance model, the log-likelihood ratio can be written as:

$$\begin{aligned}\log l = &\frac{1}{2}(\log|B| - \mu^T B\mu - \log|\tilde{\Lambda}| + \gamma_{e,t}^T \tilde{\Lambda}\gamma_{e,t}) \\ &-\frac{1}{2}(2\log|B| - 2\mu^T B\mu - 2\log|\tilde{\Gamma}| + \gamma_e^T \tilde{\Gamma}\gamma_e \\ &\qquad + \gamma_t^T \tilde{\Gamma}\gamma_t)\end{aligned} \tag{4.16}$$

where

$$\begin{aligned}\tilde{\Lambda} &= (B+2W)^{-1} & \Gamma &= (B+W)^{-1} \\ \gamma_{e,t} &= B\mu + W(\phi_e+\phi_t) & \gamma_i &= B\mu + W\phi_i\end{aligned} \tag{4.17}$$

Collecting all the terms that do not depend on the embeddings in the term $\tilde{k}$ and replacing (4.17) in (4.16), the LLR can be rewritten as

$$\log l = \phi_e^T \Lambda \phi_e + \phi_t^T \Lambda \phi_t + \phi_e^T \Gamma \phi_e + \phi_t^T \Gamma \phi_t + (\phi_e + \phi_t)^T c + k \tag{4.18}$$

where

$$\begin{aligned} \Gamma &= \frac{1}{2} W^T (\tilde{\Lambda} - \tilde{\Gamma}) W & \Lambda &= \frac{1}{2} W^T \tilde{\Lambda} W \\ c &= W^T (\tilde{\Lambda} - \tilde{\Gamma}) B\mu \quad k = \frac{1}{2}\tilde{k} + \frac{1}{2}(B\mu)^T \left(\Lambda \tilde{-} 2\tilde{\Gamma}\right)(B\mu) \end{aligned} \tag{4.19}$$

The LLR written as (4.18) is quadratic for the embeddings and PSVM can be discriminatively trained to find the parameters Γ, Λ, c and k. However, SVM aims to find a linear separation, but in the feature space $\Phi_{e,t} = [\phi_e \phi_t]^T$ the score function (4.18) is not linear. Recalling that we can rewrite $x^T A y$ as the Frobenius inner product $\langle A, xy^T \rangle = vec(A) vec(xy^T)$, where the $vec(\cdot)$ operation stacks the columns of a matrix into a vector, we can write (4.18) as:

$$\begin{aligned} \log l = &\left\langle \Lambda, \phi_e \phi_t^T + \phi_t^T \phi_e \right\rangle + \left\langle \Gamma, \phi_e \phi_e^T + \phi_t^T \phi_t \right\rangle \\ &+ c^T (\phi_e + \phi_t) + k \end{aligned} \tag{4.20}$$

Expanding the feature space as:

$$\varphi(\Phi_{e,t}) = \begin{bmatrix} vec(\phi_e \phi_t^T + \phi_t^T \phi_e) \\ vec(\phi_e \phi_e^T + \phi_t^T \phi_t) \\ \phi_e + \phi_t \\ 1 \end{bmatrix} = \begin{bmatrix} \varphi_\Lambda(\phi_e + \phi_t) \\ \varphi_\Gamma(\phi_e + \phi_t) \\ \varphi_c(\phi_e + \phi_t) \\ \varphi_k(\phi_e + \phi_t) \end{bmatrix} \tag{4.21}$$

and stacking the parameters as:

$$\varphi(\Phi_{e,t}) = \begin{bmatrix} vec(\Lambda) \\ vec(\Gamma) \\ c \\ k \end{bmatrix} = \begin{bmatrix} w_\Lambda \\ w_\Gamma \\ w_c \\ w_k \end{bmatrix} \tag{4.22}$$

we can write (4.18) as an explicit dot-product

$$S(\Phi_{e,t}) = S(\phi_e, \phi_t) = \log l(\phi_e, \phi_t) = w^T \varphi(\Phi_{e,t}) \tag{4.23}$$

4.1.3 Bayes Decision

Once the backend classifier is trained it is used to score two embeddings. At this stage, a decision still has to be made [81]: accept the true hypothesis $\mathcal{H}_T$, acting as both embedding belong to the same speaker, or reject it, $\mathcal{H}_F$. In order to make this decision, a threshold t must be defined, so that:

$$\begin{aligned} s \geq t &\rightarrow \mathcal{H}_T \\ s < t &\rightarrow \mathcal{H}_F \end{aligned} \tag{4.24}$$

where the score s could be a LLR. The choice of the threshold needs to account for the performance metric that we aim at optimizing. In this work, we consider Bayes risk, which measures the expected cost of using a classifier over a given dataset [82]. To introduce the risk, we start defining a cost $C(a|k)$ that quantifies the price of the specific action a, for example accept $\mathcal{H}_T$ or reject it, when the sample belongs to class k. Given a classifier $\mathcal{R}$ that provides class posterior probabilities $P(C = k|x, \mathcal{R})$ for a given sample x, we can define the expected cost of action a according to the classifier knowledge as:

$$C_{x,\mathcal{R}}(a) = \sum_{k=1}^{K} C(a|k) P(C = k|x, \mathcal{R}) \tag{4.25}$$

For an easier analysis, we can match the set of actions with the set of labels. In the binary case, the action a can assume two values: $\mathcal{H}_T$, accept the true hypothesis, or $\mathcal{H}_F$, reject it. Their combination defines four costs. In the following, we also assume that choosing the right action a has no cost, while the cost of false negative errors $C_{fn} = C(\mathcal{H}_F|\mathcal{H}_T)$ and the cost of false positive errors $C_{fp} = C(\mathcal{H}_T|\mathcal{H}_F)$ are assumed to be ≥ 0. The expected Bayes cost (4.25) for the two actions becomes:

$$\begin{aligned} C_{x,\mathcal{R}}(\mathcal{H}_T) &= C_{fp} P(\mathcal{H}_\mathcal{F}|x, \mathcal{R}) \\ C_{x,\mathcal{R}}(\mathcal{H}_F) &= C_{fn} P(\mathcal{H}_\mathcal{T}|x, \mathcal{R}) \end{aligned} \tag{4.26}$$

It follows that the optimal decision is defined as:

$$a^*(x, \mathcal{R}) = \begin{cases} \mathcal{H}_T & \text{if } r(x) > 0 \\ \mathcal{H}_F & \text{if } r(x) < 0 \end{cases} \tag{4.27}$$

with

$$r(x) = \log \frac{C_{fn} P(\mathcal{H}_T | x, \mathcal{R})}{C_{fp} P(\mathcal{H}_F | x, \mathcal{R})} \tag{4.28}$$

When $\mathcal{R}$ is a generative model, r can be expressed as a function of costs, prior probabilities, and conditional likelihood

$$r(x) = \log \frac{\pi_T C_{fn}}{(1 - \pi_T) C_{fp}} \cdot \frac{f_{X|\mathcal{H},\mathcal{R}}(x|\mathcal{H}_T)}{f_{X|\mathcal{H},\mathcal{R}}(x|\mathcal{H}_F)} \tag{4.29}$$

where π_T is the prior probability for class $\mathcal{H}_T$. The decision rule can be expressed as:

$$r(x) \lesseqgtr 0 \Leftrightarrow \log \frac{f_{X|\mathcal{H},\mathcal{R}}(x|\mathcal{H}_T)}{f_{X|\mathcal{H},\mathcal{R}}(x|\mathcal{H}_F)} \lesseqgtr -\log \frac{\pi_T C_{fn}}{(1 - \pi_T) C_{fp}} \tag{4.30}$$

This implies that the threshold is directly defined by the triplet (π_T, C_{fn}, C_{fp}), defining the working point for an application. An effective prior can be defined as:

$$\tilde{\pi} = \frac{\pi_T C_{fn}}{\pi_T C_{fn} + (1 - \pi_T) C_{fp}} \tag{4.31}$$

so that the triplet $(\tilde{\pi}, 1, 1)$ is equivalent to (π_T, C_{fn}, C_{fp}) in terms of optimal decisions [82].
The score reflects the confidence of the classifier in accepting the target speaker hypothesis, $\mathcal{H}_T$. Analyzing a typical plot of these scores, as in Figure 4.1, some common characteristics appear, such as the average higher values for the target scores, the different variance of the distributions, and the overlap between them. Moreover, it becomes evident that the choice of a threshold has a significant impact on determining the number of false negative errors P_{fn} and false positive errors P_{fp}. Adjusting the threshold results in different error counts.

The Equal Error Rate (EER) is the point where $P_{fp} = P_{fn}$. Although EER is a widely used metric for evaluating system performance, it does not assess the quality of the threshold. To address this, the two errors are combined taking into account the prior probability of a target π_T and the expected cost of these two types of errors, C_{fp} and C_{fn}. The weighted combination of P_{fp} and P_{fn} defines the un-normalized Detection Cost Function (C_{det}) [83, 84]:

$$DCF_u(\pi_T, C_{fn}, C_{fp}) = \pi_T C_{fn} P_{fn} + (1 - \pi_t) C_{fp} P_{fp} \tag{4.32}$$

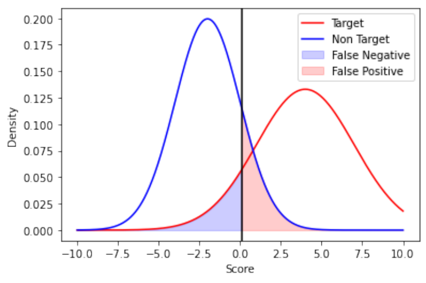


Figure 4.1 Score distributions for target and non target scores

and its normalized version:

$$DCF(\pi_T, C_{fn}, C_{fp}) = \frac{DCF_u(\pi_T, C_{fn}, C_{fp})}{\min(\pi_T C_{fn}, (1-\pi_T)C_{fp})} \tag{4.33}$$

The best possible value of the detection cost function and its normalized version achievable by adjusting the threshold t, are denoted as C_{det}^{min} and minDCF, respectively. The performance of a system can be evaluated using the DCF and C_{det}^{min} for a specific working point (i.e. a triplet, or an effective prior $\tilde{\pi}$). A possible way to summarize the performance over a wide range of working points is the Cost of log-likelihood ratio C_{llr} [84, 82, 85], which combines C_{det} values computed with bayes optimal decisions for different application priors.

4.2 Calibration

As discussed in the previous section, effective decision-making requires selecting a threshold tailored to the specific application. Well-calibrated speaker verification systems compute the log-likelihood ratio

$$x = LLR(e) = \log \frac{P(e|\mathcal{S}, \mathcal{M})}{P(e|\mathcal{D}, \mathcal{M})} \tag{4.34}$$

where the trial e is the trial pair of speaker embeddings, $\mathcal{M}$ is a statistical model for e and $\mathcal{S}$ and $\mathcal{D}$ represent the same and different speaker hypothesis, respectively. From (4.30) the optimal threshold is

$$t = -log \frac{\tilde{\pi}}{1-\tilde{\pi}} \tag{4.35}$$

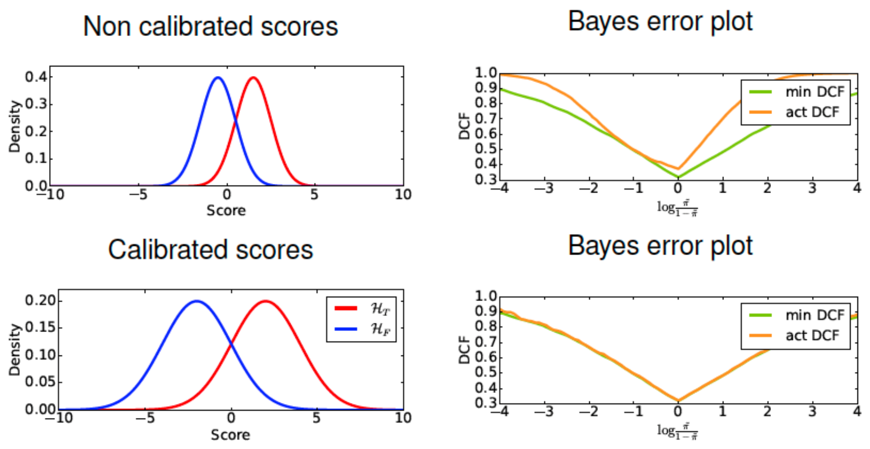


Figure 4.2 On the left, distributions of calibrated and non-calibrated scores obtained by a gaussian classifier on artificial data. On the right, the corresponding Bayes error plot

Thus, the classifier does not depend on the application while the threshold only depends on the prior probabilities and the cost of false negative and false positive errors.
In practice, many systems struggle with generating well-calibrated scores. For example, PSVM provides non-probabilistic scores, and even in cases, such as PDLA, where scores are represented as LLR, miscalibration remains a challenge. As already anticipated this miscalibration arises from disparities between the training and testing populations or inaccuracies in the model assumptions. The distance between the minimum cost, DCF_{min}, obtained with the optimal threshold for the evaluation set, and the cost, DCF_{act}, obtained with the theoretical threshold, defines the additional cost due to miscalibration. DCF can be calculated over a set of working point (π_T, C_{fn}, C_{fp}) to compare different application points. The Bayes plot (Fig. 4.2) is utilized to visualize the difference between the actual and minimum DCF for a given set of scores.

In case of significant miscalibration, score calibration is necessary to map the scores to well-calibrated LLR. To map scores s to well-calibrated scores, we can assume a linear mapping function f_{cal}

$$f_{cal}(s) = \alpha s + \gamma \tag{4.36}$$

From the definition of well-calibrate scores (4.34) and (4.30), it follows

$$log\frac{P(C=\mathcal{H}_T|s)}{P(C=\mathcal{H}_F|s)} = \alpha s + \gamma + log\frac{\tilde{\pi}}{1-\tilde{\pi}} = \alpha s + \beta \tag{4.37}$$

The most common model used to find α and β is the Prior-weighted Logistic Regression (Log-Reg) [86, 87]. As a discriminative and supervised method, Log-Reg requires labeled data (calibration samples) sharing some degree of similarity with the test population.
Generative models, instead, estimate the calibration transformation using a statistical model $\mathcal{M}'$, which describes the distribution of the observed scores. The score s is modeled as a sample of a Random Variable (R.V.) S, with $f_{S|\mathcal{S}}$ and $f_{S|\mathcal{D}}$ representing the conditional densities given the target and non-target hypotheses

$$x' = f_{cal}(s) = \log\frac{f_{S|\mathcal{S}}(s)}{f_{S|\mathcal{D}}(s)} \tag{4.38}$$

There are two ways to define the statistical model. One way is to use an explicit expression for target and non-target densities, which implicitly defines the calibration transformation. Alternatively, we define an explicit model for the calibration f_{cal} along with a model for the distribution of calibrated scores, which are constrained to satisfy the Log-Likelihood Ratio (LLR) property, as discussed in [88]. The authors of [88] show that for a well-calibrated system, the LLR of the LLR must be the LLR. This property limits the types of distributions that can represent well-calibrated LLR scores and highlights a strong relationship between the target and non-target distribution, i.e. if the non-target distribution is Gaussian, then also the target distribution must be. From this, they proposed constrained Gaussian densities with a linear calibration model (CMLG). The work in [89] demonstrated that Variance-Gamma ($V\Gamma$) distributions are effective models for well-calibrated LLR obtained through PLDA and introduced the Constrained Maximum Likelihood Variance-Gamma (CMLVG) model. In the CMLVG model, both target and non-target scores can be seen as the result of an affine transformation of well-calibrated scores, which are modeled using random variables following $V\Gamma$ distributions

$$X|\mathcal{D} \sim V\Gamma(\lambda,\alpha,\beta,\mu) \qquad\qquad X|\mathcal{S} \sim V\Gamma(\lambda,\alpha,\beta+1,\mu) \tag{4.39}$$

where the $V\Gamma$ is defined as

$$f_{V\Gamma}(x|\lambda,\alpha,\beta,\mu) = \frac{\gamma^{2\lambda}|x-\mu|^{\lambda-\frac{1}{2}} K_{\lambda-\frac{1}{2}}(\alpha|x-\mu|)}{\sqrt{\pi}\Gamma(\lambda)(2\alpha)^{\lambda-\frac{1}{2}}} e^{\beta(x-\mu)} \tag{4.40}$$

with $\lambda > 0, \alpha > |\beta|$ and $\gamma^2 = \alpha^2 - \beta^2$.

A benefit of generative models like the $V\Gamma$ in (4.39) and (4.40) is that the distribution parameters can be estimated both in a supervised and in an unsupervised way from calibration data that closely mimics evaluation data. As for LR and the CMLG model, this approach is effective when the calibration transformation is close to a linear function. While generative methods are able to manage global sources of miscalibration (global calibration), they do not address trial-dependent miscalibration, such as utterance duration. In fact, standard Log-Reg and generative models perform global calibration, the transformation that maps scores to LLRs only depends on the score itself. Furthermore, speaker vector distributions do not directly align with the underlying model assumptions. When distinct calibration sources affect each trial, unique transformation methods become necessary. For example, calibration models designed for long utterances may not be suitable for shorter ones. To address trial-level miscalibration, side-informations, like utterance duration, can be incorporated into the discriminative calibration transformation, as discussed in [90], [91], [92], and [93]. We therefore extended the analysis of well-calibrated PLDA scores [89] to explicitly address distribution mismatches between training and evaluation populations. We introduced a non-linear, generative score model that characterizes target and non-target score distributions in terms of $V\Gamma$ densities. Furthermore, we extended the model to account for trial-level miscalibration sources. Our investigation also revealed that speaker-embedding statistics, such as the squared norm of an utterance embedding, can serve as a measure of utterance-level uncertainty and improve the calibration process. The first model to handle duration was presented in [94]. The full model was later presented in our work [95]

4.2.1 Distribution of PLDA scores

Recalling the Two Covariance Model (4.6), a speaker vector ϕ, whether i-vector or a speaker embedding, is generated by the M-dimensional R.V. Φ. This model assumes that the speaker identity represented by $\bar{Y}$ and the residual noise $\bar{E}$ follow a priori normal distribution

$$\overline{\mathbf{Y}} \sim \mathcal{N}(\mathbf{m}_{\mathcal{M}}, \mathbf{B}_{\mathcal{M}}) \qquad \overline{\mathbf{E}} \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_{\mathcal{M}}) \tag{4.41}$$

where $\mathbf{m}_{\mathcal{M}}$ is the global dataset mean, while $\mathbf{B}_{\mathcal{M}}$ and $\mathbf{W}_{\mathcal{M}}$ can be viewed as between-class and within-class covariance matrices. Since PLDA is invariant to affine transformations, we can assume $\mathbf{m}_{\mathcal{M}} = 0$ and $\mathbf{B}_{\mathcal{M}}$ and $\mathbf{W}_{\mathcal{M}}$ to be diagonal. Given two speaker vectors $\mathbf{z} = \left[\phi_{\mathcal{E}}^T, \phi_{\mathcal{T}}^T\right]^T$, the PLDA will compute the score as

$$\ell(\mathbf{z}) = K_{\mathcal{M}} - \frac{1}{2}\mathbf{z}^T\left(\Sigma_{\mathcal{M},\mathfrak{S}}^{-1} - \Sigma_{\mathcal{M},\mathfrak{D}}^{-1}\right)\mathbf{z} \tag{4.42}$$

where

$$\Sigma_{\mathcal{M},\mathfrak{S}} = \begin{bmatrix} \mathbf{T}_{\mathcal{M}} & \mathbf{B}_{\mathcal{M}} \\ \mathbf{B}_{\mathcal{M}} & \mathbf{T}_{\mathcal{M}} \end{bmatrix} \qquad \Sigma_{\mathcal{M},\mathfrak{D}} = \begin{bmatrix} \mathbf{T}_{\mathcal{M}} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{\mathcal{M}} \end{bmatrix}$$

with $\mathbf{T}_{\mathcal{M}} = \mathbf{B}_{\mathcal{M}} + \mathbf{W}_{\mathcal{M}}$. Assuming $\mathbf{B}_{\mathcal{M}}, \mathbf{W}_{\mathcal{M}}$ are diagonal, (4.42) can be rewritten as a sum of M terms

$$\ell(\mathbf{z}) = \sum_{i=1}^{M} \frac{1}{2}\mathbf{z}_i^T \mathbf{A}_i \mathbf{z}_i + k_i \tag{4.43}$$

where $\mathbf{z}_i = \left[\phi_{\mathcal{E},i}\, , \phi_{\mathcal{T},i}\right]^T$ stacks the i-th components of the speaker vectors $\phi_{\mathcal{E}}, \phi_{\mathcal{T}}$, and

$$\mathbf{A}_i = \frac{\mathbf{B}_{\mathcal{M},i}}{\mathbf{T}_{\mathcal{M},i}^2 - \mathbf{B}_{\mathcal{M},i}^2} \begin{bmatrix} -\frac{\mathbf{B}_{\mathcal{M},i}}{\mathbf{T}_{\mathcal{M},i}} & 1 \\ 1 & -\frac{\mathbf{B}_{\mathcal{M},i}}{\mathbf{T}_{\mathcal{M},i}} \end{bmatrix} \tag{4.44}$$

$$\mathbf{T}_{\mathcal{M}_i} = \mathbf{B}_{\mathcal{M},i} + \mathbf{W}_{\mathcal{M},i} \tag{4.45}$$

$$k_i = \frac{1}{2}\log \mathbf{T}_{\mathcal{M},i}^2 - \frac{1}{2}\log\left(\mathbf{T}_{\mathcal{M},i}^2 - \mathbf{B}_{\mathcal{M},i}^2\right) \tag{4.46}$$

The values of $\mathbf{B}_{\mathcal{M}}$ and $\mathbf{W}_{\mathcal{M}}$ are estimated from the training population. PLDA still relies on the assumption that both the training and the evaluation population are similarly distributed (4.6). Unfortunately, the domain of the two populations could be different, and a different model may be better suited to describe the evaluation population. This discrepancy in models impacts the distribution of PLDA scores, resulting in scores that are poorly calibrated. Assuming that the PLDA model used for scoring is different from the (unknown) one modeling the evaluation trials, we consider an evaluation trial as a sample of a Random Variable (R.V.) $\mathbf{Z}$ with conditional distributions defined as:

$$\begin{bmatrix}\mathbf{\Phi}_{\mathcal{E}}\\ \mathbf{\Phi}_{\mathcal{T}}\end{bmatrix}|\mathfrak{S} \sim \mathcal{N}(\mathbf{m},\mathbf{\Sigma}_{\mathfrak{S}}) \qquad \Sigma_{\mathfrak{S}} = \begin{bmatrix}\mathbf{T}_{\mathcal{E}} & \mathbf{B}_{C}\\ \mathbf{B}_{C} & \mathbf{T}_{\mathcal{T}}\end{bmatrix} \tag{4.47}$$
$$\begin{bmatrix}\mathbf{\Phi}_{\mathcal{E}}\\ \mathbf{\Phi}_{\mathcal{T}}\end{bmatrix}|\mathfrak{D} \sim \mathcal{N}(\mathbf{m},\mathbf{\Sigma}_{\mathfrak{D}}) \qquad \Sigma_{\mathfrak{D}} = \begin{bmatrix}\mathbf{T}_{\mathcal{E}} & \mathbf{0}\\ \mathbf{0} & \mathbf{T}_{\mathcal{T}}\end{bmatrix}$$

where $\mathbf{T}_{\mathcal{E}} = \mathbf{B}_C + \mathbf{W}_{\mathcal{E}}$ and $\mathbf{T}_{\mathcal{T}} = \mathbf{B}_C + \mathbf{W}_{\mathcal{T}}$. This correspond to the PLDA model Φ defined for enrollment $\mathcal{E}$ and test $\mathcal{T}$

$$\begin{aligned} \Phi_{\mathcal{E}} &= \mathbf{Y}_C + \mathbf{E}_{\mathcal{E}} & \Phi_{\mathcal{T}} &= \mathbf{Y}_C + \mathbf{E}_{\mathcal{T}} \\ Y_C &\sim \mathcal{N}(\mathbf{m},\mathbf{B}_C) & Y_C &\sim \mathcal{N}(\mathbf{m},\mathbf{B}_C) \\ E_{\mathcal{E}} &\sim \mathcal{N}(\mathbf{0},\mathbf{W}_{\mathcal{E}}) & E_{\mathcal{T}} &\sim \mathcal{N}(\mathbf{0},\mathbf{W}_{\mathcal{T}}) \end{aligned} \tag{4.48}$$

In this model, $\mathbf{W}_{\mathcal{E}}$ and $\mathbf{W}_{\mathcal{T}}$ represent the within-class variability within the enrollment and test segments. These may vary from one segment to another, due to the different levels of information carried by spoken segments and the potential uncertainty in extracting speaker vectors. The common between-class variability of the evaluation population is captured by the matrix $\mathbf{B}_C$. To ensure the tractability of the model, we make the assumption that $\mathbf{B}_C$, $\mathbf{W}_{\mathcal{E}}$, and $\mathbf{W}_{\mathcal{T}}$ are all diagonal. While this may appear as a strong assumption, it is justified by the fact that the model that we derive outperforms current calibration state-of-the-art models. Furthermore, we adopt the assumption that the mean $\mathbf{m}$ is $\mathbf{m} = \mathbf{0}$, although it is worth highlighting that this can be estimated using a small set of unlabelled calibration samples and compensated using unlabeled evaluation samples. As demonstrated in [89], a trial $\mathbf{z}$ is a realization of R.V. $\mathbf{Z}$, we can use a R.V. $\mathcal{L} = l(\mathbf{Z})$ to describe the score.
It is worth noting that, given a R.V. $X(\omega)$ defined over the probability space $(\Omega, \mathcal{A}, \mathcal{P})$, and an event H, then $X|H$ is not a R.V. over $(\Omega, \mathcal{A}, \mathcal{P}(\cdot))$. However, from [96] we can interpret conditioning as acting on the probability rather than just on the probability distribution. Meaning that $(X|H)(\omega)$ can be defined as a R.V. which is functionally identical to $X(\omega)$, $(X|H)(\omega) := X(\omega)$, but is defined on the probability space $(\Omega \cap H, \{H \cap A\} : A \in \mathcal{A}, \mathcal{P}(\cdot,\mathcal{A})/\mathcal{P}(\mathcal{A}))$. Not mixing R.V.s defined over different spaces allows us to work with the R.V.s $\mathbf{Z}_{\mathfrak{S}} \sim \mathbf{Z}|\mathfrak{S}$ and $\mathbf{Z}_{\mathfrak{D}} \sim \mathbf{Z}|\mathfrak{D}$, each defined on its own probability space, whose distributions corresponds to the conditional distributions of R.V. $\mathbf{Z}$, given the $\mathfrak{S}$ or $\mathfrak{D}$ hypotheses. Finally, with an abuse of

notation, we will use $\mathbf{Z}|\mathfrak{S}$ and $\mathbf{Z}|\mathfrak{D}$ to denote both the conditional distribution $\mathbf{Z}|\mathfrak{S}$ and $\mathbf{Z}|\mathfrak{D}$, and the distribution of R.V. $\mathbf{Z}_{\mathfrak{S}}$ and $\mathbf{Z}_{\mathfrak{D}}$. The same considerations apply to score distributions $\mathcal{L}|\mathfrak{S}$ and $\mathcal{L}|\mathfrak{D}$. It follows that the distribution of target and non-target scores can be obtained considering R.V.s

$$\mathbf{Z}_{\mathfrak{S}} \sim \mathbf{Z}|\mathfrak{S} \qquad \mathbf{Z}_{\mathfrak{D}} \sim \mathbf{Z}|\mathfrak{D}$$

This makes the scores realizations of two R.V.s

$$\mathcal{L}|\mathfrak{S} = \ell(\mathbf{Z}|\mathfrak{S}) \qquad \mathcal{L}|\mathfrak{D} = \ell(\mathbf{Z}|\mathfrak{D}) \tag{4.49}$$

Under the same and different speaker hypotheses, given $\mathbf{B}_{\mathcal{E}}$,$\mathbf{W}_{\mathcal{E}}$ and $\mathbf{W}_{\mathcal{T}}$ diagonal, the R.V.s of the different components of the speaker vectors $\mathbf{Z}_i = \left[\mathbf{\Phi}_{\mathcal{E},i}\ , \mathbf{\Phi}_{\mathcal{T},i}\right]$ are independent

$$\mathbf{Z}_i|\mathfrak{S} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}_{\mathfrak{S},i}\right) \qquad \mathbf{Z}_i|\mathfrak{D} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}_{\mathfrak{D},i}\right) \tag{4.50}$$

where

$$\Sigma_{\mathfrak{S},i} = \begin{bmatrix} \mathbf{T}_{\mathcal{E},i} & \mathbf{B}_{C,i} \\ \mathbf{B}_{C,i} & \mathbf{T}_{\mathcal{T},i} \end{bmatrix} \qquad \Sigma_{\mathfrak{D},i} = \begin{bmatrix} \mathbf{T}_{\mathcal{E},i} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{\mathcal{T},i} \end{bmatrix} \tag{4.51}$$

and $\mathbf{B}_{C,i}$, $\mathbf{T}_{\mathcal{E},i}$ and $\mathbf{T}_{\mathcal{T},i}$ are the i-th elements of the diagonals of $\mathbf{B}_C$, $\mathbf{T}_{\mathcal{E}}$ and $\mathbf{T}_{\mathcal{T}}$, respectively. From (4.43), we can use a sum of M independent R.V. to express the distribution of PLDA scores for target and non-target cases

$$\mathcal{L}|\mathfrak{S} = \ell(\mathbf{Z}|\mathfrak{S}) = \sum_{i=1}^{M} \mathcal{L}_i|\mathfrak{S} \qquad \mathcal{L}|\mathfrak{D} = \ell(\mathbf{Z}|\mathfrak{D}) = \sum_{i=1}^{M} \mathcal{L}_i|\mathfrak{D} \tag{4.52}$$

$$\mathcal{L}_i|\mathfrak{S} = \ell(\mathbf{Z}_i|\mathfrak{S}) = \frac{1}{2}\mathbf{Z}_{\mathfrak{S},i}^T\mathbf{A}_i\mathbf{Z}_{\mathfrak{S},i} + k_i \qquad \mathcal{L}_i|\mathfrak{D} = \ell(\mathbf{Z}_i|\mathfrak{D}) = \frac{1}{2}\mathbf{Z}_{\mathfrak{D},i}^T\mathbf{A}_i\mathbf{Z}_{\mathfrak{D},i} + k_i \tag{4.53}$$

$$Z_{\mathfrak{S},i} \sim \mathbf{Z}_i|\mathfrak{S} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathfrak{S},i}) \qquad Z_{\mathfrak{D},i} \sim \mathbf{Z}_i|\mathfrak{D} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathfrak{D},i}) \tag{4.54}$$

We define a standard normal distributed R.V. $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the Cholesky decomposition $\mathbf{C}_{\mathfrak{S}}$ and $\mathbf{C}_{\mathfrak{D}}$ as

$$\mathbf{\Sigma}_{\mathfrak{S},i} = \mathbf{C}_{\mathfrak{S},i}\mathbf{C}_{\mathfrak{S},i}^T \qquad \mathbf{\Sigma}_{\mathfrak{D},i} = \mathbf{C}_{\mathfrak{D},i}\mathbf{C}_{\mathfrak{D},i}^T$$

and the eigendecomposition

$$\mathbf{M}_{\mathfrak{S},i} \triangleq \mathbf{C}_{\mathfrak{S},i}^T \mathbf{A}_i \mathbf{C}_{\mathfrak{S},i} = \mathbf{U}_{\mathfrak{S},i} \mathbf{D}_{\mathfrak{S},i} \mathbf{U}_{\mathfrak{S},i}^T \qquad \mathbf{M}_{\mathfrak{D},i} \triangleq \mathbf{C}_{\mathfrak{D},i}^T \mathbf{A}_i \mathbf{C}_{\mathfrak{D},i} = \mathbf{U}_{\mathfrak{D},i} \mathbf{D}_{\mathfrak{D},i} \mathbf{U}_{\mathfrak{D},i}^T \tag{4.55}$$

Then the conditional distributions of $\mathcal{L}_i$ can be expressed as:

$$\begin{aligned} \mathcal{L}_i|\mathfrak{S} &\sim \frac{1}{2}\mathbf{X}^T\mathbf{C}_{\mathfrak{S},i}^T\mathbf{A}_i\mathbf{C}_{\mathfrak{S},i}\mathbf{X}+k_i & \qquad \mathcal{L}_i|\mathfrak{D} &\sim \frac{1}{2}\mathbf{X}^T\mathbf{C}_{\mathfrak{D},i}^T\mathbf{A}_i\mathbf{C}_{\mathfrak{D},i}\mathbf{X}+k_i \\ &\sim \frac{1}{2}\mathbf{Y}_{\mathfrak{S}}^T\mathbf{D}_{\mathfrak{S},i}\mathbf{Y}_{\mathfrak{S}}+k_i & &\sim \frac{1}{2}\mathbf{Y}_{\mathfrak{D}}^T\mathbf{D}_{\mathfrak{D},i}\mathbf{Y}_{\mathfrak{D}}+k_i \end{aligned} \tag{4.56}$$

where, since $\mathbf{U}_{\mathfrak{S},i}$ and $\mathbf{U}_{\mathfrak{D},i}$ are orthogonal,

$$\mathbf{Y}_{\mathfrak{S}} = \mathbf{U}_{\mathfrak{S},i}\mathbf{X} \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \qquad \mathbf{Y}_{\mathfrak{D}} = \mathbf{U}_{\mathfrak{D},i}\mathbf{X} \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \tag{4.57}$$

We can see that the determinant of $\mathbf{A}_i$ (Eq (4.44)) is negative, it follows that also the determinant of $\mathbf{M}_{\mathfrak{S},i}$ and $\mathbf{M}_{\mathfrak{D},i}$ are negative. As a result, we can infer that the two eigenvalues of $\mathbf{M}_{\mathfrak{S},i}$ and $\mathbf{M}_{\mathfrak{D},i}$ have different signs. By examining the distribution of quadratic, indefinite forms in (4.56), we can derive an expression for $\mathcal{L}_i$. We omit all suffixes to improve readability and consider quadratic forms

$$\mathcal{L} = \frac{1}{2}\mathbf{Y}^T\mathbf{D}\mathbf{Y} + k = \frac{1}{2}d_+Y_+^2 - \frac{1}{2}|d_-|Y_-^2 + k \tag{4.58}$$

where $\mathbf{D}$ is a 2×2 diagonal matrix with diagonal elements $d_+ > 0$ and $d_- < 0$

$$\mathbf{D} = \begin{bmatrix} d_+ & 0 \\ 0 & d_- \end{bmatrix} \tag{4.59}$$

and $\mathbf{Y}$ follows a standard normal distribution:

$$\mathbf{Y} = \begin{bmatrix} Y_+ \\ Y_- \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right). \tag{4.60}$$

$Y_+$and Y_- are independent, and Y_+^2 and Y_-^2 follow a Gamma-distribution

$$Y_+ \sim \Gamma\left(\frac{1}{2},\frac{1}{2}\right) \quad Y_- \sim \Gamma\left(\frac{1}{2},\frac{1}{2}\right) \tag{4.61}$$

meaning that $\mathcal{L}$ coincides with a difference of independent, Gamma-distributed R.V.s

$$\begin{aligned} &\mathcal{L} = G_+ - G_- + k \quad G_+ \triangleq \frac{1}{2} d_+ Y_+^2 \quad G_- \triangleq \frac{1}{2} |d_-| Y_-^2 \\ &G_+ \sim \Gamma\left(\frac{1}{2}, \frac{1}{d_+}\right) \quad G_- \sim \Gamma\left(\frac{1}{2}, \frac{1}{|d_-|}\right) \end{aligned} \tag{4.62}$$

The Moment Generating Function (MGF) of $\mathcal{L}$ is

$$\begin{aligned} M_{\mathcal{L}}(t) &= e^{kt} M_{G_+}(t) M_{G_-}(-t) \\ &= e^{kt} (1 - d_+ t)^{-\frac{1}{2}} (1 - d_- t)^{-\frac{1}{2}} \\ &= e^{kt} \left(1 - \mathrm{Tr}(\mathbf{D}) t + \det(\mathbf{D}) t^2\right)^{-\frac{1}{2}} \end{aligned} \tag{4.63}$$

where Tr and det denote the trace and determinant operators, respectively. For a random variable X following the $\mathrm{V}\Gamma(\lambda, \alpha, \beta, \mu)$ distribution, the MGF can be expressed as:

$$M_X(t) = e^{\mu t} \left(1 - \frac{2\beta}{\gamma^2} t - \frac{t^2}{\gamma^2}\right)^{-\lambda} \tag{4.64}$$

where $\gamma^2 = \alpha^2 - \beta^2$.

This derivation show that $\mathcal{L}$ is $V\Gamma$-distributed [89], and the parameters can be recovered by inspection:

$$\lambda = \frac{1}{2} \quad \mu = k \quad \gamma^2 = -\frac{1}{\det(\mathbf{D})} \quad \beta = -\frac{1}{2} \frac{\mathrm{Tr}(\mathbf{D})}{\det(\mathbf{D})} \tag{4.65}$$

From (4.65) and (4.56) it can be derived that $\mathcal{L}_i|\mathfrak{S}$ and $\mathcal{L}_i|\mathfrak{D}$ are also $\mathrm{V}\Gamma$-distributed. And from (4.55) we have that

$$\begin{aligned} &\mathrm{Tr}(\mathbf{D}_{\mathfrak{S},i}) = \mathrm{Tr}(\mathbf{M}_{\mathfrak{S},i}) = \mathrm{Tr}(\mathbf{A}_i \mathbf{\Sigma}_{\mathfrak{S},i}) \qquad &\mathrm{Tr}(\mathbf{D}_{\mathfrak{D},i}) = \mathrm{Tr}(\mathbf{M}_{\mathfrak{D},i}) = \mathrm{Tr}(\mathbf{A}_i \mathbf{\Sigma}_{\mathfrak{D},i}) \\ &\det(\mathbf{D}_{\mathfrak{S},i}) = \det(\mathbf{M}_{\mathfrak{S},i}) = \det(\mathbf{A}_i \mathbf{\Sigma}_{\mathfrak{S},i}) \qquad &\det(\mathbf{D}_{\mathfrak{D},i}) = \det(\mathbf{M}_{\mathfrak{D},i}) = \det(\mathbf{A}_i \mathbf{\Sigma}_{\mathfrak{D},i}) \end{aligned} \tag{4.66}$$

so that

$$\mathcal{L}_i|\mathfrak{S} \sim \mathrm{V}\Gamma\left(\frac{1}{2}, \alpha_{\mathfrak{S},i}, \beta_{\mathfrak{S},i}, k_i\right) \qquad \mathcal{L}_i|\mathfrak{D} \sim \mathrm{V}\Gamma\left(\frac{1}{2}, \alpha_{\mathfrak{D},i}, \beta_{\mathfrak{D},i}, k_i\right) \tag{4.67}$$

with parameters

$$\begin{aligned}
\beta_{\mathfrak{S},i} &= -\frac{1}{2}\frac{\operatorname{Tr}(\mathbf{A}_i\boldsymbol{\Sigma}_{\mathfrak{S},i})}{\det(\mathbf{A}_i\boldsymbol{\Sigma}_{\mathfrak{S},i})} & \beta_{\mathfrak{D},i} &= -\frac{1}{2}\frac{\operatorname{Tr}(\mathbf{A}_i\boldsymbol{\Sigma}_{\mathfrak{D},i})}{\det(\mathbf{A}_i\boldsymbol{\Sigma}_{\mathfrak{D},i})} \\
\gamma^2_{\mathfrak{S},i} &= -\frac{1}{\det(\mathbf{A}_i\boldsymbol{\Sigma}_{\mathfrak{S},i})} & \gamma^2_{\mathfrak{D},i} &= -\frac{1}{\det(\mathbf{A}_i\boldsymbol{\Sigma}_{\mathfrak{D},i})} \\
\alpha^2_{\mathfrak{S},i} &= \gamma^2_{\mathfrak{S},i} + \beta^2_{\mathfrak{S},i} & \alpha^2_{\mathfrak{D},i} &= \gamma^2_{\mathfrak{D},i} + \beta^2_{\mathfrak{D},i}
\end{aligned} \tag{4.68}$$

The relationship between the parameters of the speaker vectors distribution $\mathbf{B}_{\mathcal{M},i}$, $\mathbf{W}_{\mathcal{M},i}$, $\mathbf{B}_{\mathcal{C},i}$, $\mathbf{W}_{\mathcal{E},i}$, $\mathbf{W}_{\mathcal{T},i}$ and the parameters in (4.68) can be expressed through the ratios

$$\eta_{\mathcal{E},i} = \frac{\mathbf{T}_{\mathcal{M},i}}{\mathbf{T}_{\mathcal{E},i}} \qquad \eta_{\mathcal{T},i} = \frac{\mathbf{T}_{\mathcal{M},i}}{\mathbf{T}_{\mathcal{T},i}} \tag{4.69}$$

$$\rho_{\mathcal{E},i} = \frac{\mathbf{B}_{\mathcal{C},i}}{\mathbf{W}_{\mathcal{E},i}} \qquad \rho_{\mathcal{T},i} = \frac{\mathbf{B}_{\mathcal{C},i}}{\mathbf{W}_{\mathcal{T},i}} \qquad \rho_{\mathcal{M},i} = \frac{\mathbf{B}_{\mathcal{M},i}}{\mathbf{W}_{\mathcal{M},i}} \tag{4.70}$$

The scaling differences between the training population and the enrollment and test population are characterized by the ratios $\eta_{\mathcal{E},i}$ and $\eta_{\mathcal{T},i}$. While the ratios $\rho_{\mathcal{M},i}, \rho_{\mathcal{E},i}$ and $\rho_{\mathcal{T},i}$ indicate the between-to-within class variability for the training, enrollment, and test populations. Moreover, we can express $\eta_{\mathcal{E},i}$ and $\eta_{\mathcal{T},i}$ as functions of $\rho_{\mathcal{E},i}$ and $\rho_{\mathcal{T},i}$ and a ratio between the scale of the training population and the average scale of enrollment and test population, $\eta_{\mathcal{C},i} = \frac{2\mathbf{T}_{\mathcal{M},i}}{\mathbf{T}_{\mathcal{E},i}+\mathbf{T}_{\mathcal{T},i}}$. It is often reasonable to assume that the enrollment and test populations are homogeneous, denoted as $\mathbf{W}_{\mathcal{E}} = \mathbf{W}_{\mathcal{T}} \triangleq \mathbf{W}_{\mathcal{C}}$. Under this circumstance, $\rho_{\mathcal{E},i} = \rho_{\mathcal{T},i} \triangleq \rho_{\mathcal{C},i}$, $\eta_{\mathcal{E},i} = \eta_{\mathcal{T},i} = \eta_{\mathcal{C},i}$, which leads to

$$\begin{aligned}
\gamma^2_{\mathfrak{S},i} &= \gamma^2_{\mathfrak{D},i}\frac{(1+\rho_{\mathcal{C},i})^2}{1+2\rho_{\mathcal{C},i}} & \gamma^2_{\mathfrak{D},i} &= \eta^2_{\mathcal{C},i}\frac{1+2\rho_{\mathcal{M},i}}{\rho^2_{\mathcal{M},i}} \\
\beta_{\mathfrak{S},i} &= \eta_{\mathcal{C},i}\frac{1+\rho_{\mathcal{C},i}}{1+2\rho_{\mathcal{C},i}}\left(\frac{\rho_{\mathcal{C},i}}{\rho_{\mathcal{M},i}} - 1\right) & \beta_{\mathfrak{D},i} &= -\eta_{\mathcal{C},i}
\end{aligned} \tag{4.71}$$

The skewness of the target distribution depends on the ratios $\rho_{\mathcal{C},i}$ and $\rho_{\mathcal{M},i}$. The distribution will be right-skewed if $\rho_{\mathcal{C},i}$ is greater than $\rho_{\mathcal{M},i}$, meaning that along direction i the evaluation vectors are easier to discriminate than the training samples. We can also notice that the between-over-within variability ratio $\rho_{\mathcal{C},i}$ of the evaluation population does not play any role in shaping the non-target distribution. If we set that $\rho_{\mathcal{E},i} = \rho_{\mathcal{T},i} = \rho_{\mathcal{M},i}$ and $\eta_{\mathcal{E},i} = \eta_{\mathcal{T},i} = 1$, which correspond to the assumption that the evaluation population follows the same distribution as the training population, we can go back to the results showed in [89]. We lack a closed-form solution for the distribution $\mathcal{L}|\mathfrak{D}$ and $\mathcal{L}|\mathfrak{S}$ for the general case. Nevertheless, assuming that the ratios $\eta_{C,i}$, $\rho_{\mathcal{M},i}$, $\rho_{\mathcal{E},i}$ and $\rho_{\mathcal{T},i}$ are the same for along any directions i

$$\eta_{C,i} = \eta_C \quad \rho_{\mathcal{M},i} = \rho_{\mathcal{M}} \quad \rho_{\mathcal{E},i} = \rho_{\mathcal{E}} \quad \rho_{\mathcal{T},i} = \rho_{\mathcal{T}} \quad \forall i \tag{4.72}$$

then $\mathcal{L}|\mathfrak{D}$ and $\mathcal{L}|\mathfrak{S}$ are again VΓ-distributed, with

$$\mathcal{L}|\mathfrak{S} \sim \mathrm{V}\Gamma\left(\frac{M}{2}, \alpha_{\mathfrak{S}}, \beta_{\mathfrak{S}}, \sum_{i=1}^{M} k_i\right) \qquad \mathcal{L}|\mathfrak{D} \sim \mathrm{V}\Gamma\left(\frac{M}{2}, \alpha_{\mathfrak{D}}, \beta_{\mathfrak{D}}, \sum_{i=1}^{M} k_i\right) \tag{4.73}$$

and the parameters $\alpha_{\mathfrak{D}}, \beta_{\mathfrak{D}}, \alpha_{\mathfrak{S}}, \beta_{\mathfrak{S}}$ can be computed from (4.68) using any index i.

4.2.2 The $V\Gamma$ model, a generative model for mismatched data

Assuming that the covariance matrices, for any of the speaker vector directions, are linked to a common set of normalized variances as

$$\begin{aligned} &\mathbf{B}_{\mathcal{M},i} = \xi_i b_{\mathcal{M}} \quad \mathbf{W}_{\mathcal{M},i} = \xi_i w_{\mathcal{M}} \\ &\mathbf{B}_{C,i} = \xi_i b_C \quad\;\; \mathbf{W}_{\mathcal{E},i} = \xi_i w_{\mathcal{E}} \quad \mathbf{W}_{\mathcal{T},i} = \xi_i w_{\mathcal{T}} \;, \end{aligned} \tag{4.74}$$

for scalars $\xi_i \neq 0$, and that, the distribution of the speaker vectors, with M-dimension, is described by (4.47), then equation (4.73) can be used to describe the distribution of target and non-target scores. The work in [89] has shown that assuming that only a small set of "effective" speaker vector dimensions significantly influences the scores and that the assumption made in (4.74) is a valid approximation. In fact, most speaker vector dimensions exhibit low between-over-within variability ratios, minimizing their contribution to the PLDA scores. Thus, we can assume that the scores are generated by a speaker vector with the effective dimensions approximately following the assumptions in (4.74). To incorporate this behavior into the distributions of target and non-target scores, we can utilize the λ parameter of the $V\Gamma$ distributions [89]. Our model can be derived starting from the score model

$$\mathcal{L}|\mathfrak{S} \sim \mathrm{V}\Gamma(\lambda, \mu, \alpha_{\mathfrak{S}}, \beta_{\mathfrak{S}}) \qquad \mathcal{L}|\mathfrak{D} \sim \mathrm{V}\Gamma(\lambda, \mu, \alpha_{\mathfrak{D}}, \beta_{\mathfrak{D}}) \tag{4.75}$$

where λ and μ are shared parameters modeling the shape and the location, while $\alpha_{\mathfrak{D}}, \alpha_{\mathfrak{S}}, \beta_{\mathfrak{D}}, \beta_{\mathfrak{S}}$ depend on the parameters $b_{\mathcal{M}}, w_{\mathcal{M}}, b_{\mathcal{C}}, w_{\mathcal{E}}, w_{\mathcal{T}}$ through

$$
\begin{aligned}
&\alpha_{\mathfrak{S}}^2 = \gamma_{\mathfrak{S}}^2 + \beta_{\mathfrak{S}}^2 && \alpha_{\mathfrak{D}}^2 = \gamma_{\mathfrak{D}}^2 + \beta_{\mathfrak{D}}^2 \\
&\beta_{\mathfrak{S}} = -\frac{1}{2}\frac{\operatorname{Tr}(\mathbf{A}\boldsymbol{\Sigma}_{\mathfrak{S}})}{\det(\mathbf{A}\boldsymbol{\Sigma}_{\mathfrak{S}})} && \beta_{\mathfrak{D}} = -\frac{1}{2}\frac{\operatorname{Tr}(\mathbf{A}\boldsymbol{\Sigma}_{\mathfrak{D}})}{\det(\mathbf{A}\boldsymbol{\Sigma}_{\mathfrak{D}})} \\
&\gamma_{\mathfrak{S}}^2 = -\frac{1}{\det(\mathbf{A}\boldsymbol{\Sigma}_{\mathfrak{S}})} && \gamma_{\mathfrak{D}}^2 = -\frac{1}{\det(\mathbf{A}\boldsymbol{\Sigma}_{\mathfrak{D}})} \\
&\mathbf{A} = \boldsymbol{\Sigma}_{\mathcal{M},\mathfrak{D}}^{-1} - \boldsymbol{\Sigma}_{\mathcal{M},\mathfrak{S}}^{-1} \\
&\boldsymbol{\Sigma}_{\mathfrak{S}} = \begin{bmatrix} t_{\mathcal{E}} & b_{\mathcal{C}} \\ b_{\mathcal{C}} & t_{\mathcal{T}} \end{bmatrix} && \boldsymbol{\Sigma}_{\mathfrak{D}} = \begin{bmatrix} t_{\mathcal{E}} & 0 \\ 0 & t_{\mathcal{T}} \end{bmatrix} \\
&t_{\mathcal{M}} = b_{\mathcal{M}} + w_{\mathcal{M}} \quad t_{\mathcal{E}} = b_{\mathcal{C}} + w_{\mathcal{E}} \quad t_{\mathcal{T}} = b_{\mathcal{C}} + w_{\mathcal{T}} \\
&\boldsymbol{\Sigma}_{\mathcal{M},\mathfrak{S}} = \begin{bmatrix} t_{\mathcal{M}} & b_{\mathcal{M}} \\ b_{\mathcal{M}} & t_{\mathcal{M}} \end{bmatrix} && \boldsymbol{\Sigma}_{\mathcal{M},\mathfrak{D}} = \begin{bmatrix} t_{\mathcal{M}} & 0 \\ 0 & t_{\mathcal{M}} \end{bmatrix}
\end{aligned}
\tag{4.76}
$$

The solution for the $V\Gamma$ parameters is invariant to a scaling factor $\xi > 0$ for the variance terms in (4.76). The model is over-parametrized so in the following analysis we fix $w_{\mathcal{M}} = 1$. For the PLDA LLR, both distributions are expected to share the location parameter μ, which should be linked to the remaining parameters, $b_{\mathcal{M}}, b_{\mathcal{C}}, w_{\mathcal{E}}, w_{\mathcal{T}}$. However, often is better to use a shared location parameter estimated from the data and independent of the other parameters. This allows modeling bias terms of PLDA-derived models, such as Pairwise Support Vector Machines (PSVM) [79, 80, 97] or discriminative PLDA [98]. While these bias terms are optimal for the training set and the specific training loss, they may result in poor calibration for the evaluation data. Moreover, in the previous derivations we did not consider the effects of linear terms introduced in the scoring function of PSVM or discriminative PLDA. These dataset shifts or the equivalent effects of linear terms would result in more complex distributions, and we do not know of a closed-form solution, even with the isotropic assumption. Nonetheless, we can assume that the shift is small enough so that its impact can be modeled through a linear transformation of the $V\Gamma$ distributions. The proposed $V\Gamma$ model becomes

$$
\mathcal{L}|\mathfrak{S} \sim \mathrm{V}\Gamma\left(\lambda, \mu_{\mathfrak{S}}, \frac{\alpha_{\mathfrak{S}}}{a_{\mathfrak{S}}}, \frac{\beta_{\mathfrak{S}}}{a_{\mathfrak{S}}}\right) \qquad \mathcal{L}|\mathfrak{D} \sim \mathrm{V}\Gamma\left(\lambda, \mu_{\mathfrak{D}}, \alpha_{\mathfrak{D}}, \beta_{\mathfrak{D}}\right) \tag{4.77}
$$

where we have added $a_{\mathfrak{S}}$ to handle the scale disparities between the target and non-target distributions, and $\mu_{\mathfrak{S}}$ and $\mu_{\mathfrak{D}}$ to characterize the two independent location parameters. The non-target class does not require a scaling parameter, as the original parameter can be estimated to account for it. The 8 free parameters $\mathbf{\Pi} = (\lambda, \mu_{\mathfrak{D}}, \mu_{\mathfrak{S}}, b_{\mathcal{M}}, b_{\mathcal{C}}, w_{\mathcal{E}}, w_{\mathcal{T}}, a_{\mathfrak{S}})$ fully describe our model. When both enrollment and test data are affected by independent and identically distributed nuisance, we can tie $w_{\mathcal{E}} = w_{\mathcal{T}} = w_{\mathcal{C}}$, decreasing the parameters to 7. We denote this model as $V\Gamma$-Var, as its parameters can be seen as "effective" variances. To train the model, we maximize the weighted likelihood, over a set of calibration scores $(\mathcal{S}_{\mathfrak{S}}, \mathcal{S}_{\mathfrak{D}})$ [89]

$$\arg\max_{\mathbf{\Pi}} \frac{\zeta}{|\mathcal{S}_{\mathfrak{S}}|} \sum_{s \in \mathcal{S}_{\mathfrak{S}}} f_{\mathcal{L}|\mathfrak{S}}(s; \mathbf{\Pi}) + \frac{1-\zeta}{|\mathcal{S}_{\mathfrak{D}}|} \sum_{s \in \mathcal{S}_{\mathfrak{D}}} f_{\mathcal{L}|\mathfrak{D}}(s; \mathbf{\Pi}) \tag{4.78}$$

where $f_{\mathcal{L}|\mathfrak{S}}(s; \mathbf{\Pi})$ and $f_{\mathcal{L}|\mathfrak{D}}(s; \mathbf{\Pi})$ are the VΓ densities for the target and non-target distributions (4.75) and ζ is a weighting factor. The calibration transformation defines a non-linear transformation of the scores as

$$f_{cal}(s; \mathbf{\Pi}) = \frac{\mathrm{V}\Gamma\left(s|\lambda, \mu_{\mathfrak{S}}, \frac{\alpha_{\mathfrak{S}}}{a_{\mathfrak{S}}}, \frac{\beta_{\mathfrak{S}}}{a_{\mathfrak{S}}}\right)}{\mathrm{V}\Gamma\left(s|\lambda, \mu_{\mathfrak{D}}, \alpha_{\mathfrak{D}}, \beta_{\mathfrak{D}}\right)} \tag{4.79}$$

with the $V\Gamma$ densities defined in (4.40). The set of parameters can be obtained in a discriminative fashion with the prior-weighted logistic regression, setting the objective function

$$\begin{aligned} \arg\min_{\mathbf{\Pi}} \quad & \frac{\pi}{|S_{\mathfrak{S}}|} \sum_{s \in S_{\mathfrak{S}}} \log\left(1 + e^{-f_{cal}(s;\mathbf{\Pi}) - \log\frac{\pi}{1-\pi}}\right) \\ & + \frac{1-\pi}{|S_{\mathfrak{D}}|} \sum_{s \in S_{\mathfrak{D}}} \log\left(1 + e^{f_{cal}(s;\mathbf{\Pi}) + \log\frac{\pi}{1-\pi}}\right) \end{aligned} \tag{4.80}$$

where π is the effective prior for the target class. In the experiment and results section we will show that the generative and discriminative models obtain close results (more details in Section 5.4.1), showing that our model effectively characterizes the score distributions.

4.2.3 Utterance-dependent calibration

In our model, we assume that the score of a trial $z = \left[\boldsymbol{\phi}_{\mathcal{E}}^T, \boldsymbol{\phi}_{\mathcal{T}}^T\right]^T$ is a sample of the $V\Gamma$ distributions in (4.75). The within-class variability for the enrollment and test speaker vectors, described by $w_{\mathcal{E}}$ and $w_{\mathcal{T}}$, is typically considered constant. However, to accommodate factors specific to individual utterances, such as utterance duration, we suppose that the values of $w_{\mathcal{E}}$ and $w_{\mathcal{T}}$ can vary across utterances. Specifically, the effective within-class variance w_i of each speaker-vector ϕ_i becomes a function of both i.i.d. and utterance-dependent miscalibration sources.

Utterance duration

The accuracy of a speaker verification system is heavily influenced by the duration of an utterance. Inspired by i-vector models [12], our goal is to model the influence of utterance duration on w_i. The i-vector model offers a means to consider uncertainty in i-vectors by utilizing the i-vector posterior covariance matrix (2.16) [99]–[100]. To incorporate i-vector uncertainty at the trial level, we adopt a similar approach to equation (4.47). This involves replacing the covariance matrices $\mathbf{T}_{\mathcal{E}}$ and $\mathbf{T}_{\mathcal{T}}$ with $\mathbf{T}_{\mathcal{E}} = \mathbf{B}_{\mathcal{C}} + \mathbf{W}_{\mathcal{E}} + \mathbf{C}_{\mathcal{E},i}$ and $\mathbf{T}_{\mathcal{T}} = \mathbf{B}_{\mathcal{C}} + \mathbf{W}_{\mathcal{T}} + \mathbf{C}_{\mathcal{T},i}$, where $\mathbf{C}_{\mathcal{E},i}$ and $\mathbf{C}_{\mathcal{T},i}$ are the i-vector posterior covariances for enroll and test i-vectors of trial $\mathbf{z}_i$. The i-vector posterior covariance matrix $\mathbf{C}$ possesses a complex expression that relies on zero-order statistics for an utterance, computed on a Universal Background Model (UBM). Nonetheless, it can be reasonably approximated, as discussed in [101], by a matrix defined as

$$\mathbf{C} \approx (\mathbf{I} + D\mathbf{M})^{-1} \tag{4.81}$$

where $\mathbf{M}$ depends on the UBM and the i-vector model parameters, and D indicates the utterance duration. To further simplify the computation and its efficiency, we assume that $\mathbf{C}$ shares its principal direction with $\mathbf{W}_{\mathcal{M}}$, $\mathbf{W}_{\mathcal{E}}$, and $\mathbf{W}_{\mathcal{T}}$. This assumption, which derives from a similar approximation utilized in [102], allows for their joint diagonalization. Each component $\mathbf{C}_j$ of $\mathbf{C}$ can be expressed as $\mathbf{C}_j = \frac{1}{1+D\eta_j^{-1}} = \frac{\eta_j}{\eta_j+D}$. This means that the effects of utterance duration on the within-class variability of speaker vectors can be encoded by the i-vector posterior covariance matrix. In our score model, we incorporate utterance duration through a similar functional relationship and represent the effective variances $w_{\mathcal{E},i}$ and $w_{\mathcal{T},i}$ for a trial z_i as

$$w_{\mathcal{E},i} = w_{\mathcal{E}} + \frac{\psi}{D_{\mathcal{E},i} + \eta} \qquad\qquad w_{\mathcal{T},i} = w_{\mathcal{T}} + \frac{\psi}{D_{\mathcal{T},i} + \eta} \tag{4.82}$$

where $D_{\mathcal{E},i}$ and $D_{\mathcal{T},i}$ represent the duration of the enrollment and test segments, and ψ and η are additional model parameters shared across all trials.

In the novel $V\Gamma$-Var + Dur, we assume $w_{\mathcal{E}} = w_{\mathcal{T}} = w_{\mathcal{C}}$, as we did for the $V\Gamma$-Var model. It is important to note that in situations where uncertainty estimates are unavailable (e.g., in the case of x-vectors [103]) or cannot be integrated at the classification level (e.g. PSVM), this approach can model speaker vector uncertainty.

Speaker vector norms

Length normalization is a common pre-processing step applied to the embeddings to enhance the classification performance of the standard PLDA. We can demonstrate that the squared norm of a speaker vector can be regarded as a measure of utterance-specific variability, linking the effect of length normalization to the reduction of intra-trial mismatch. The following analysis is derived from the simplified Heavy-Tailed PLDA (HT-PLDA), introduced in [104]:

$$\begin{gathered} \mathbf{\Phi}_i = \mathbf{m} + \overline{\mathbf{U}}\mathbf{Y} + \mathbf{E}_i \\ \mathbf{Y} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \mathbf{E}_i | V_i \sim \mathcal{N}(\mathbf{0}, V_i \mathbf{W}) \quad V_i^{-1} \sim \Gamma\left(\frac{a}{2}, \frac{a}{2}\right) \end{gathered} \tag{4.83}$$

where V_i is a hidden R.V. that represents an utterance-dependent scaling factor. The HT-PLDA model assumes a gamma prior for the inverse of V_i. We can assume that the dimensionality of $\mathbf{Y}$ is $D \ll M$. This assumption is derived from the standard PLDA, where the speaker subspace has a much lower dimension than the speaker vector space. We further assume that the speaker vectors have been withened so that $\mathbf{m} = \mathbf{0}$, $\mathbf{W} = \mathbf{I}$, and the D-dimensional speaker subspace corresponds to the first D principal directions of the speaker vector space. The speaker vector can be partitioned into two components:

$$\overline{\mathbf{U}} = \begin{bmatrix} \mathbf{U} \\ \mathbf{0} \end{bmatrix} \quad \mathbf{\Phi}_i = \begin{bmatrix} \mathbf{\Phi}_i^u \\ \mathbf{\Phi}_i^n \end{bmatrix} = \begin{bmatrix} \mathbf{U}\mathbf{Y} + \mathbf{E}_i^u \\ \mathbf{E}_i^n \end{bmatrix} \tag{4.84}$$

where $\mathbf{U}$ is a $D \times D$ matrix, $\mathbf{\Phi}_i^u$ refers to the first d components of $\mathbf{\Phi}_i$, while $\mathbf{\Phi}_i^n$ represents to the remaining components of $\mathbf{P}hi_i$, that do not depend on the speaker factor $\mathbf{y}$. A similar distinction applies to the noise terms $\mathbf{E}_i^u$, $\mathbf{E}_i^n$. Typically, length normalization is applied to whitened vectors. When the whitening operation is performed on the covariance of speaker vectors, it results in $\mathbf{E}_i^n$ being approximately standard normal distributed, as per our assumption. Given a set of vectors k belonging to a single speaker, the marginal density is

$$\begin{aligned}
&f_{\mathbf{\Phi}_1 \dots \mathbf{\Phi}_k}(\phi_1 \dots \phi_k) \\
&= \int f_{\mathbf{\Phi}_1 \dots \mathbf{\Phi}_k | \mathbf{Y}, V_1 \dots V_k}(\phi_1 \dots \phi_k | \mathbf{y}, v_1 \dots v_k) f_{\mathbf{Y}}(\mathbf{y}) d\mathbf{y} \prod_{i=1}^{k} f_{V_i}(v_i) dv_i \\
&= \int \prod_{i=1}^{k} f_{\mathbf{\Phi}_i^u | \mathbf{Y}, V_i}\left(\phi_i^u | \mathbf{y}, v_i\right) f_{\mathbf{Y}}(\mathbf{y}) d\mathbf{y} \prod_{i=1}^{k} f_{\mathbf{\Phi}_i^n | V_i}\left(\phi_i^n | v_i\right) f_{V_i}(v_i) dv_i \\
&= \prod_{i=1}^{k} f_{\mathbf{\Phi}_i^n}\left(\phi_i^n\right) \int \prod_{i=1}^{k} f_{\mathbf{\Phi}_i^u | \mathbf{Y}, V_i}\left(\phi_i^u | \mathbf{y}, v_i\right) f_{V_i | \mathbf{\Phi}_i^n}\left(v_i | \phi_i^n\right) dv_i f_{\mathbf{Y}}(\mathbf{y}) d\mathbf{y} \qquad (4.85)
\end{aligned}$$

The LLR for a given trial depends only on the integral term of equation (4.85). It is worth highlighting that $V_i | \mathbf{\Phi}_i^n$ acts as a prior for the conditional likelihood of $\mathbf{\Phi}_i^u | Y, V_i$, with

$$\left(V_i^{-1} | \mathbf{\Phi}_i^n = \phi_i^n\right) \sim \Gamma\left(\frac{a + M - D}{2}, \frac{a + \left\|\phi_i^n\right\|^2}{2}\right) \qquad (4.86)$$

Instead of dealing with the posterior distribution directly, we replace the posterior distribution for V_i with a Maximum-a-Posteriori (MAP) point estimate approximation.

$$v_i^{\mathrm{MAP}} = \frac{a + \left\|\phi_i^n\right\|^2}{a + M - D + 1} \qquad (4.87)$$

The posterior distribution of $V_i | \mathbf{\Phi}_i^n = \phi_i^n$ takes the form of an inverse-Gamma distribution with the same parameters of (4.86). The MAP estimate for V_i^{-1} has a similar formal expression, although it is slightly different. By substituting the random variable V_i with the MAP estimate (4.87), we return to a standard PLDA model. In this model, the noise variance for embedding i is directly associated with the estimated value $v_i^{\mathrm{MAP}}\mathbf{I}$.

It is noteworthy that matrix $\overline{\mathbf{U}}$ can be approximated by the subspace discovered through Linear Discriminant Analysis. By doing so, we can avoid estimating the HT-PLDA model. Consequently, the norm of the embedding computed in the complement of the LDA subspace aligns with the MAP solution. Under the assumption that the speaker subspace has a small dimension and the between-to-within variability ratio is generally low across most embedding directions, the norm of the original whitened embedding serves as a further approximation of the MAP solution. Through the application of length normalization, the distribution of the noise term becomes approximately independent of the utterance, while introducing a dependency of the speaker factor distribution on the utterance. Once we have established that utterance-dependent variability can be approximated by $\mathbf{v}_i^{\mathrm{MAP}}$, we can incorporate it in our calibration model. This can be achieved by defining effective enrollment and test variances as

$$w_{\mathcal{E},i} = w_{\mathcal{C}} + \psi \left\| \mathbf{U}_c \phi_{\mathcal{E},i} \right\|^2 \qquad w_{\mathcal{T},i} = w_{\mathcal{C}} + \psi \left\| \mathbf{U}_c \phi_{\mathcal{T},i} \right\|^2 \tag{4.88}$$

where $\mathbf{U}_c$ is a projection matrix corresponding to the complement of an LDA subspace, and ψ is a parameter that can be estimated by Maximum Likelihood over the calibration set.

4.3 Score-Normalization

Utterance-dependent variability measures, such as duration, have been introduced in both discriminative [90–92] and generative models [95, 94] to address trial-level miscalibration sources. An alternative technique for addressing trial-dependent miscalibration is score normalization [6, 105, 76, 106–108]. The score normalization goal is to align the distribution of non-target (impostor) scores from various enrollment speakers and test segments to a fixed, common distribution, typically a standard Gaussian. This normalization process involves using impostor cohorts, which consist of unlabeled impostor scores computed for both the enrollment and test sides of a trial. Trial scores are normalized using the impostor score statistics. Symmetric normalization (S-norm) [76] and Adaptive S-norm (AS-norm) [107, 108] are some of the most established and successful approaches.

S-norm is achieved by averaging Z-normalized [6] and T-normalized [105] scores. For a trial (e,t) and an impostor cohort, $\{x_i\}_{i=1}^N$, with N samples, S-norm is defined as:

$$s_{s-norm}(e,t) = \frac{1}{2}\left(\frac{s(e,t)-\mu(e)}{\sigma(e)} + \frac{s(e,t)-\mu(t)}{\sigma(t)}\right) \tag{4.89}$$

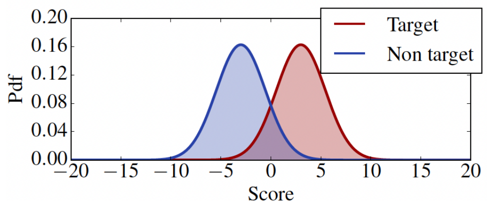


Figure 4.3 Gaussian-distributed score densities for a single speaker.

where $s(e,t)$ corresponds to the unnormalized score for the trial (e,t). Statistics, $\mu(e)$ and $\sigma(e)$ are the mean and standard deviation of the impostor scores $s(e,x_i)_{i=1}^{N}$. Likewise, $\mu(t)$ and $\sigma(t)$ for the set $s(x_i,t)_{i=1}^{N}$. These statistics can also be obtained from an adaptively selected cohort, as in AS-norm [107, 108]. The adaptive strategy selects the subset so that the cohort used for computing the enrollment e statistics behaves similarly to the test utterance t, and vice versa. Using an adaptive score normalization approach can lead to more diverse cohorts, covering a wider range of potential conditions. As a result, adaptive score normalization methods often yield improved accuracy compared to non-adaptive techniques. One advantage of score normalization is its ability to utilize unlabeled datasets with only a single repetition per impostor speaker, without the need to add side information. On the other hand, while calibration approaches can utilize unlabeled datasets to estimate calibration parameters, they still rely on multiple repetitions per speaker and to address trial-level miscalibration effectively they require additional side information. However, due to the lack of information about the distribution of target scores, score normalization is not able to achieve global calibration. Therefore, global calibration remains necessary and an additional dataset, with multiple repetitions, is still required. Additionally, in particular cases, score normalization suffers a second disadvantage. In fact, although score normalization is generally effective in aligning score distributions and improving trial-level calibration, there are instances where it may amplify trial-level miscalibration. This can be shown by considering a simple scenario in which we fix an enrollment speaker i and assume that both impostor and target scores are generated by Gaussian-distributed random variables, as in Figure 4.3

We also assume that the scores are well-calibrated LLRs. From [88], the same-speaker and different speaker distribution densities $f_{\mathfrak{S}}$ and $f_{\mathfrak{D}}$ are required to be related by:

$$f_{\mathfrak{S}}(s) = \mathcal{N}\left(s \,\middle|\, \frac{1}{2}v_i, v_i\right) \qquad\qquad f_{\mathfrak{D}}(s) = \mathcal{N}\left(s \,\middle|\, -\frac{1}{2}v_i, v_i\right) \tag{4.90}$$

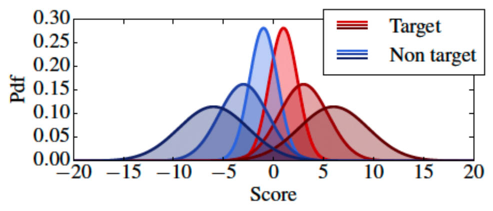


Figure 4.4 Gaussian-distributed score densities for three different speakers. Densities with the same variance correspond to the same speaker.

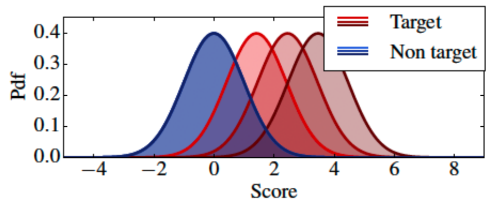


Figure 4.5 Score densities for three different speakers after Z-norm.

where v_i is a variance term. It is important to notice that a larger variance means that trials are easier to discriminate. The amount of information carried by an utterance is linked to its nature, including factors like different noise levels and durations. This implies that even for well-calibrated verification models, we should expect score distributions with different variances for each distinct enrolled speaker. Figure 4.4 represents a scenario with three different speakers with different variances. The target and non-target score distributions for each speaker share the same variance value.

By construction, the corresponding scores of each speaker are well-calibrated. The pooled scores for the three speakers will also be well-calibrated if we assume that trials are obtained by uniformly sampling over the three enrolled speakers

Since we want to show the possible negative effect of score normalization, we apply Z-normalization to the scores in Figure 4.4. The results are shown in Figure 4.5, the scores of each speaker have been normalized to align the impostor densities with standard Gaussian distributions. The results show that the non-target score densities overlap, while the target ones have different locations but the same scale.

It is clear that after normalization, the pooled scores are no longer well-calibrated and do not adhere to the LLR property anymore. Figure 4.6 shows that Z-normalization has introduced trial-level miscalibration. In fact, even if an optimal threshold can be found for one speaker, it will be sub-optimal for the other two speakers.

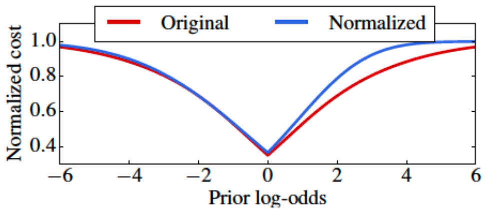


Figure 4.6 Bayes error plots of the minimum cost for the synthetic scores of Figures 4.4 in red and 4.5 in blue.

The Bayes error plots in Figure 4.6 provide additional confirmation that the normalized scores result in higher costs across a wide range of application priors. To address both limitations we have proposed to integrate the impostor score statistics into a global discriminative calibrator [109]. In [110] we also introduced adaptive cohort selection for Data Normalization as a further way to address the second issue.

4.3.1 Impostor score statistics as quality measures

Prior logistic regression is a common discriminative approach used to achieve global calibration and it can be enriched with side information, such as utterance duration, to address trial-level calibration. We propose to utilize the impostor score statistics outlined in score normalization as side information, to enhance trial-dependent miscalibration while obtaining good global calibration [109]. In Z-normalization, the scores of a specific group of enrollment speakers are recalibrated using a speaker-dependent transformation. It is important to notice that we will extend the discussion to S-norm to incorporate both sides of a trial in our model.
Although in the previous chapter we have introduced $V\Gamma$ distributions as better suited to describe scores, for the sake of simplifying the derivation of this model we assume that for a given speaker i, the scores represent samples from Gaussian-distributed random variables $X_{\mathfrak{D},i}$ and $X_{\mathfrak{S},i}$. It is also assumed that these scores are well-calibrated, up to a speaker-dependent affine transformation:

$$X_{\mathfrak{S},i} \sim a_i X^{cal}_{\mathfrak{S},i} + b_i \qquad\qquad X_{\mathfrak{D},i} \sim a_i X^{cal}_{\mathfrak{D},i} + b_i \tag{4.91}$$

where

$$X^{cal}_{\mathfrak{S},i} \sim \mathcal{N}\left(s \middle| \frac{1}{2}v_i, v_i\right) \qquad\qquad X^{cal}_{\mathfrak{D},i} \sim \mathcal{N}\left(s \middle| -\frac{1}{2}v_i, v_i\right) \tag{4.92}$$

The parameters a_i and b_i represent the enrollment-dependent miscalibration, while v_i is the variance of the well-calibrated R.V.s $X^{cal}_{\mathfrak{D},i}$ and $X^{cal}_{\mathfrak{S},i}$. Knowing both a_i and b_i of each enrolled speaker the calibrated score can be obtained as:

$$s_{cal} = \frac{s - b_i}{a_i} \tag{4.93}$$

It should be noted that fixing $a_i = 1$ for all speakers allows us to compute b_i from the mean $m_{\mathfrak{D},i}$ and variance v_i of $X^{cal}_{\mathfrak{D},i}$:

$$b_i = m_{\mathfrak{D},i} + \frac{1}{2} v_i \tag{4.94}$$

Since $m_{\mathfrak{D},i}$ and v_i are usually unknown, they can be replaced using values derived from the set of impostor scores for speaker i. However, in practice, the value of a_i is not given, making it challenging to estimate both a_i and b_i without knowledge of the target score distribution for speaker i. Score normalization circumvents this problem by assuming that, up to a speaker-dependent affine transformation, for each speaker, their impostor scores have been generated by R.V.s with the same distribution

$$X_{\mathfrak{D},i} \sim a_i X^{cal}_{\mathfrak{D}} + b_i \qquad\qquad X^{cal}_{\mathfrak{D}} \sim \mathcal{N}\left(-\frac{1}{2}v, v\right) \tag{4.95}$$

Since a successful score-normalization implies that for each speaker the normalized scores should be well-calibrated:

$$X_{\mathfrak{S},i} \sim a_i X^{cal}_{\mathfrak{S}} + b_i \qquad\qquad X^{cal}_{\mathfrak{S}} \sim \mathcal{N}\left(\frac{1}{2}v, v\right) \tag{4.96}$$

The resulting assumption is that the distributions of the calibrated scores have the same mean and the same variance $v_i = v$ for all speakers. Under these conditions, both a_i and b_i can be estimated up to speaker-independent (global) factors, that can be managed through a global linear calibration. When datasets are close to the same-variance assumption score normalization produces close to optimal results. However, as previously demonstrated, in certain applications this assumption is not accurate enough and leads to worse trial-level calibration. To derive our model, instead of the same-variance assumption, we assume that the scaling factor a_i is shared among the speakers. Assuming $a_i = a$ for each speaker is equivalent to the assumption that most of the trial-dependent miscalibration is the result of a speaker-dependent (trial-dependent generally) shift, while the amount of information carried by different utterances is modeled by different variance values. In this case, miscalibrated scores are assumed to be generated by R.V.s with distributions

$$X^{cal}_{\mathfrak{S},i} \sim aX^{cal}_{\mathfrak{S},i} + b_i \qquad\qquad X^{cal}_{\mathfrak{D},i} \sim aX^{cal}_{\mathfrak{D},i} + b_i \tag{4.97}$$

where $X^{cal}_{\mathfrak{D},i}$ and $X^{cal}_{\mathfrak{S},i}$ are the calibrated R.V.s defined in (4.92). Combining the two equation

$$X^{cal}_{\mathfrak{S},i} \sim \mathcal{N}(m_{\mathfrak{S},i}, v_{\mathfrak{S},i}) \qquad\qquad X^{cal}_{\mathfrak{D},i} \sim \mathcal{N}(m_{\mathfrak{D},i}, v_{\mathfrak{D},i}) \tag{4.98}$$

where the parameters $m_{\mathfrak{D},i}$, $v_{\mathfrak{D},i}$, $m_{\mathfrak{S},i}$ and $v_{\mathfrak{S},i}$, of the target and non-target scores are tied as:

$$m_{\mathfrak{S},i} = \frac{1}{2}av_i + b_i \qquad\qquad m_{\mathfrak{D},i} = -\frac{1}{2}av_i + b_i \tag{4.99}$$

$$v_{\mathfrak{S},i} = v_{\mathfrak{D},i} = a^2 v_i \tag{4.100}$$

From (4.97) the calibrated score can then be obtained as

$$s_{cal} = \frac{s - b_i}{a} \tag{4.101}$$

And retrieving $b_i = m_{\mathfrak{D},i} + \frac{1}{2}\frac{v_{\mathfrak{D},i}}{a}$ from (4.99), s_{cal} can be defined as:

$$s_{cal} = \frac{1}{a}s - \frac{1}{a}m_{\mathfrak{D},i} - \frac{1}{2a^2}v_{\mathfrak{D},i} \tag{4.102}$$

Also, in this case, $m_{\mathfrak{D},i}$ and v_i are unknown but can be estimated from a set of impostors. Nonetheless, the scale parameter a has to be estimated, which requires information about the target score distribution. To determine the value of a we can employ Maximum Likelihood, or we can notice that (4.102) is equivalent to a linear calibration model with a speaker-dependent shift. The terms $m_{\mathfrak{D},i}$ and $v_{\mathfrak{D},i}$ can be treated as side-information (quality measures) [91], which is linearly combined with a re-scaled score. Hence, to estimate this calibration transformation we can use a discriminative linear fusion model, such as prior-weighted logistic regression [86]. Additionally, the model can be extended to automatically estimate the weights of different terms from the data. The calibration model, enriched with impostor side information, is defined as:

$$s_{cal} = \alpha s + \beta m_{\mathfrak{D},i} + \gamma v_{\mathfrak{D},i} + k \tag{4.103}$$

where the values of α, β, γ and k are found by training a prior-weighted logistic regression. This model presents two benefits. Firstly, it combines normalization and calibration to optimize a proper scoring rule. Secondly, unlike conventional score normalization methods, the ability of the model to estimate the weights of different terms allows it to prevent the introduction of trial-dependent miscalibration (estimating $\beta \approx \gamma \approx 0$) in scenarios that deviate from our model assumption. The model in (4.102) is derived from the Z-norm style approach. By adding test statistics (T-norm) as side information, (4.102) can be extended to handle both enrollment and test statistics. Given a trial (e,t), we define the mean m_e and variance v_e of the scores, which are determined by comparing the enrollment utterance against the normalization cohorts, similarly m_t and v_t are obtained by comparing the test utterance. The new calibration transformation becomes

$$s_{cal} = \alpha s(e,t) + \beta m_e + \gamma v_e + \delta m_t + \epsilon v_t + k \tag{4.104}$$

From our experience, we have found that characterizing interactions between the test and the enrollment impostor distributions can improve the accuracy of the model. Thus, in the final model, we add an additional term:

$$s_{cal} = \alpha s(e,t) + \beta m_e + \gamma v_e + \delta m_t \epsilon v_t + \zeta \sqrt{v_e v_t} + k \tag{4.105}$$

Prior-weighted logistic regression can be trained to estimate all the model parameters $(\alpha, \beta, \gamma, \delta, \epsilon, \zeta, k)$. It should be noted, that other kinds of side-information, such as utterance duration [91], can be easily introduced in our model. Also, our impostor statistics can be used as side information in other calibration approaches [93]. Finally, we can extend our model utilizing adaptive cohort selection to replace the statistics in (4.105) with values computed from the selected subsets for each trial. In fact, adaptive score normalization has demonstrated greater efficacy in scenarios where the impostor cohort is more heterogeneous.

4.3.2 From adaptive score normalization to adaptive data normalization

As an alternative to score-level compensation, the reduction of the mismatch between evaluation and training population at embedding level may lead to improved performance, reducing also miscalibration issues. Additionally, in the following, we show that score normalization can also be interpreted as a sub-optimal approach to data-level normalization. We propose a way to integrate the most beneficial aspect of score normalization, namely the trial-level mismatch compensation deriving from adaptive cohort selection, with mismatch compensation at embedding level.

In the Adaptive S-Norm [108, 107] the cohort is formed by selecting utterances that are close to the enrollment and test segment. The statistics in (4.89) are computed over these utterance-dependent cohorts, $C(e)$ and $C(t)$, and used to compute the normalized score of a given trial (e,t). Using the scores between the enrollment segment e and the normalization segments x_i with respect to the entire impostor set, we build the vectors

$$v_t = [s(e,x_1) \quad s(e,x_2) \quad \dots \quad s(e,x_N)] \tag{4.106}$$

$$v_i = [s(x_i,x_1) \quad s(x_i,x_2) \quad \dots \quad s(x_i,x_N)] \tag{4.107}$$

$C(e)$ is obtained by computing the Euclidean distance

$$d_i = ||v_e - v_i||^2 \tag{4.108}$$

and selecting K impostor utterances associated with the lowest values (usually between 200 and 400). The selected impostor utterances are the most similar to the enrollment segment. These utterances are compared with the test segment to obtain the score set $\{s(x_i,t)|x_i \in C(e)\}$ from which the test normalization statistics are obtained, namely the mean $\mu(t|e)$ and the standard deviation $\sigma(t|e)$. To compute the enrollment statistics $\mu(e|t)$ and $\sigma(e|t)$, the method is the same, with the roles of enrollment and test swapped. The normalized score is defined as

$$s_{asnorm}(e,t) = \frac{s(e,t) - \mu(e|t)}{2\sigma(e|t)} + \frac{s(e,t) - \mu(t|e)}{2\sigma(t|e)} \tag{4.109}$$

Adaptive data normalization

As we have shown earlier in Section 4.3, when the original scores are well-calibrated, score normalization has the potential to worsen trial-level miscalibration effects, leading to a reduction of the performance. Conversely, the dataset-level and trial-level mismatch can be lowered by directly aligning the distribution of training and evaluation embeddings on a per-trial basis. In fact, this would further align the evaluation data to the backend model assumption. In the Adaptive Mean (AM-norm) [111] approach, the normalization process is moved from the score level to the speaker vector. A mean vector is computed from a set of impostor utterances to re-center both enrollment and test embeddings. In [110] we propose a similar approach in contrast to score normalization. Unlike [111], however, we employ the same selection strategy of the AS-norm to build the utterance-dependent impostor cohort. As explained earlier, for each enrollment and test segment, we independently compute the cohort sets $C(e)$ and $C(t)$, and each embedding is independently normalized by removing the mean of the corresponding cohort

$$\hat{e} = e - m_e \qquad m_e = \frac{1}{|C(e)|} \sum_{i|x_i \in C(e)} x_i \tag{4.110}$$

$$\hat{t} = t - m_t \qquad m_t = \frac{1}{|C(t)|} \sum_{i|x_i \in C(t)} x_i \tag{4.111}$$

The recognizer scoring function $s(\cdot,\cdot)$ provided by the standard backend can then be used to score the normalized enrollment and test segments

$$s_{adnorm} = s(\hat{e},\hat{t}) = s(e - m_e, t - m_t) \tag{4.112}$$

The name of our approach is Adaptive Data normalization (AD-norm). Length normalization is a common pre-processing step for many backends. In our methodology, we not only re-center the L2-normalized embeddings but also perform an additional L2-normalization to align them more accurately with the classifier assumptions. Differently from [111], our approach eliminates the need for a specialized condition classifier. In fact, our approach is able to automatically select impostor segments that exhibit a certain degree of similarity to the trial segments. Moreover, because the normalization process is independently applied to enrollment and test embeddings, it can be efficiently computed during the extraction of embeddings without incurring into additional costs during scoring. It is important to note that AD-norm can be integrated into the clustering feature framework introduced in [112], for large-scale clustering of speaker vectors, unlike AS-norm. The adoption of the AS-norm selection mechanism can be justified by exposing the analogies between the AD-norm and the AS-norm. In fact, some of the effects of the AS-norm can be explained as an imperfect application of embedding-level normalization. We assume that the training and evaluation data share the same distribution up to a shift of the evaluation mean. Given the scoring function of zero-mean PLDA [79, 80]

$$s(e,t) = e^T Ae + t^T At + e^T Bt \tag{4.113}$$

the optimal score would be defined by re-centering the evaluation data

$$\begin{aligned} s_{opt} =& s(e-m_e, t-m_t) & (4.114)\\ & (e-m_e)^T A(e-m_e) + (t-m_t)^T A(t-m_t) + (e-m_e)^T B(t-m_t) & (4.115)\end{aligned}$$

where m_e and m_t are the mean vectors of the distributions of the enrollment and test segments, respectively. In AD-norm m_e and m_t are replaced with the respective estimation obtained from the impostor sets $C(e)$ and $C(t)$. Ignoring the variance normalization, the AS-norm score is given by

$$s_{asnorm}(e,t) = \frac{1}{2}(s_t(e,t) - \mu(e|t)) + \frac{1}{2}(s_t(e,t) - \mu(t|e)) \tag{4.116}$$

where the mean are computed from the impostor sets $C(e)$ and $C(t)$. The cohort sets of AS-norm and Ad-norm may be different, but we assume that they effectively characterize the utterance. This guarantees that both (4.112) and (4.114) represent the optimal scoring rule for the trial. The statistics of the AS-norm are computed as:

$$\mu(e|t) = \frac{1}{|C(t)|} \sum_{i|x_i \in C(t)} s(e,x_i) = e^T Ae + e^T Bm_t + \epsilon_1 \tag{4.117}$$

$$\mu(t|e) = \frac{1}{|C(e)|} \sum_{i|x_i \in C(e)} s(t,x_i) = t^T At + t^T Bm_e + \epsilon_2 \tag{4.118}$$

where m_e and m_t are defined as in (4.110) and the terms that do not depend on e or t are captured by ϵ_1 and ϵ_2. Although these terms affect the global shift of the score, they do not impact the intra-trial calibration compensation of score normalization.

The normalized score is given by

$$s_{asnorm}(e,t) = \frac{1}{2}\left(t^T At + e^t B(t - m_t)\right) + \frac{1}{2}\left(e^T Ae + t^t B(e - m_e)\right) \tag{4.119}$$

Table 4.3.2 shows a comparison between (4.119) with (4.112) and (4.114). Despite the similarities in the expressions, some terms in AD-norm carry different weights or are entirely absent from AS-norm.

s_{adnorm}	s_{asnorm}
$e^T Ae$	$\frac{1}{2}e^T Ae$
$-2e^T Am_e$	
$t^T At$	$\frac{1}{2}t^T At$
$-2etTAm_t$	
$e^T Bt$	$e^T Bt$
$-e^T Bm_e$	$-\frac{1}{2}e^T Bm_e$
$-t^T Bm_t$	$-\frac{1}{2}t^T Bm_t$

We notice that the only term that is shared among AS-norm and AD-norm is $e^T Bt$. This term expresses the similarity of the samples and is reasonably the most important one. If the dataset shift is sufficiently small, the terms that are not present in both expressions can be disregarded. Based on this comparison, it can be concluded that the mean normalization part of AS-norm serves as an approximation of the optimal scoring rule for PLDA-derived models. The effectiveness of AS-norm highlights the value of its cohort selection approach, which accurately captures the characteristics of test and enrollment utterances independently from the backend. This finding provides the rationale for employing the same methodology to compute the cohort sets $C(t)$ and $C(e)$ for AD-norm.

Chapter 5

Experimental Results

In this chapter, we show the experimental outcomes of our work. The first section provides an overview of the development of neural networks used as embedding extractors, highlighting their progress, advancements, and refinements over time. Within this section, we discuss our proposed improvements for embedding extraction, which we first employed in our successful participation in the NIST LRE 2022 evaluation xSarni2023DescriptionAA. In the second section, we present the results of our experiments on calibration and score normalization, which have been published in [94, 95, 109, 110].

5.0.1 Hardware Setup

The majority of our deep learning models were trained on a single NVIDIA Tesla V100 GPU within a high-performance computing cluster (Table 5.1). The computational resources required to perform our experiments were provided by the HPC cluster at Politecnico di Torino[1].

5.1 Speaker Embeddings

A good speaker embedding extractor is the starting point when tackling the speaker verification problem. In Chapter 3 we presented the key idea behind the deep

[1]http://hpc.polito.it

Table 5.1 Hardware and software specifications of the workstation used to run the Model Trainer and the Link Evaluator.

Workstation hardware and software specifications	
CPU model	2x Intel Xeon Scalable Processors Gold 6130 2.10 GHz 16 cores.
GPU model	4x nVidia Tesla V100 SXM2 - 32 GB - 5120 cuda cores (on 6 nodes)
Total RAM memory	21.888 TB DDR4 REGISTERED ECC
OS	Centos 7.6 - OpenHPC 1.3.8.1
SW packages	CUDA toolkit 10.2, Python 3.8.8, PyTorch 1.8.1

neural architectures used for embedding extraction. Different architectures have been proposed to solve this step, each with its benefits and problems. In this section we analyze the baseline performance of some of these solutions for speaker verification, highlighting the architectures most suitable for our task. We will also provide details on the modifications we employed to improve the performance of the baseline models. The resulting architectures will be the basis for our experiments on calibration (Section 5.4).

5.1.1 Datasets

Our networks were trained using various datasets, each containing different speech segments of varying durations and from different speakers. In the following we provide the details of the training datasets.

- **VoxCeleb**, which is comprised of two datasets - VoxCeleb1 and VoxCeleb2, is the largest and primary dataset used for training our classifiers. These datasets were created through speech detection and face recognition techniques and consist of a large set of text-independent speech segments collected from YouTube videos [113].

- **Mixer** is a collection of different datasets, which includes Mixer6 and MixerPhone (with several versions such as Mixer5, Mixer10, etc.). This dataset contains interviews, transcript readings, and phone conversations. Mixer6 has longer speech segments from native speakers compared to other datasets, as shown in Table 5.2.

Table 5.2 Statistics of the datasets used.

	Number of speaker	**Number of file**	**Avg duration**
VoxCeleb1	1,251	22,496	43.59 s
VoxCeleb2	5,994	145,569	46.49 s
Mixer6	594	13,088	388.65 s
MixerPhone	3,269	37,909	106.04 s
SwitchBoard	1,676	24,556	116.68 s

- **Switchboard** is another dataset that contains phone and cellular audio recordings from both sides of the conversation.

A summary of these datasets is shown in Table 5.2. Both VoxCeleb datasets are available on the official website[2], while Mixer and Switchboard are from the Linguistic Data Consortium.

5.1.2 Augmentation

In speaker verification, the evaluation data is usually made up of speakers that were not seen during the training phase. This means that the network must learn to classify the speakers in the training data while also being able to extract the most speaker-related characteristics from new and unseen data. To make the system more robust, augmentation is often used. Augmentation can be used to increase both the number of samples and the number of speakers. In this study, we focused on the former by processing and transforming our data to produce additional distorted data. This helps the networks learn a representation that is invariant to these distortions and thus generalizes better to unseen data. We produced three versions of each segment using one of four speech augmentation techniques:

- **Music** augmentation: a segment of random music is added as a background to the original utterances.

- **Noise** augmentation: different noises are added to the original utterances.

[2]https://www.robots.ox.ac.uk/~vgg/data/voxceleb/

- **Babble** augmentation: a random number of speech segments belonging to different people is added to the background of the original utterance.
- **Reverb** augmentation: reverberation in different kinds of rooms, from small to large, is simulated for each original utterance.

Music, speech and noises are taken from the MUSAN [114] corpus. Reverberation is done using the Room Impulse Response and Noise Database[3].

5.1.3 Extraction

The core of our experiment was conducted by processing and extracting features from the raw audio as explained in Section 2.1. We extract a 46-dimensional log-mel spectrogram feature vector from each utterance. We also remove silence frames from each sample using a neural voice activity detection (VAD), as the one detailed in [115] feature extraction section.

5.1.4 Evaluation

We conducted an evaluation to measure the effectiveness and contribution of different modules and architecture focusing on the NIST Speaker Recognition Evaluation (SRE). In particular, to evaluate our embedding extractor we used SRE18 data [116] to train a PLDA model and the SRE19 Evaluation as evaluation data [117]. To ensure consistency and robustness in our evaluations, we used the same pipeline to assess the performance of our networks. The embeddings are centered, and their dimensionality is reduced to 150 using LDA. We use WCCN to whiten the reduced embeddings and finally apply L2-length normalization. We will report the Equal Error rate and C_{prim}^{min} as defined by NIST for the evaluation [117].

5.2 Speaker Embedding Results

From each utterance, both original and augmented, we extract all possible 3-second segments. The duration of segments was selected based on results, with 3 seconds

[3]https://www.openslr.org/28/

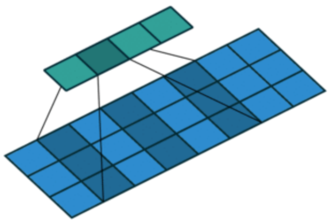


Figure 5.1 1-D convolution with dilation detail. Figure taken from [122]

being the best compromise between effectiveness and computational complexity. In one epoch of training all segments are forwarded through the network.
Although ETDNN (Extended Time Delay Neural Network) showed good results with Natural Gradient Descent, the time required for training didn't allow us to use it consistently. Therefore, we choose to use either Stochastic Gradient Descent (SGD) or AdamW [118]. The majority of our experiment employed the same cyclic scheduler [119], based on the *triangular2* strategy of the CyclicLR implemented in PyTorch.

5.2.1 Neural Networks for Speaker Verification

Time Delay Neural Network (Section 3.2.1), specifically ETDNN outlined in Table 5.3, was used as our starting point and baseline. The network is implemented following [120], using 1-dimensional convolution exploiting the dilation parameter (Fig. 5.1).

Table 5.3 Details of the Extended TDNN as in [120] (left) and Factorized TDNN as in [121] (right) architectures implementation.

	Layer	Context	Size
1	TDNN-ReLU	t-2:t+2	512
2	TDNN-ReLU	t	512
3	TDNN-ReLU	t-2, t, t+2	512
4	TDNN-ReLU	t	512
5	TDNN-ReLU	t-2, t, t+3	512
6	TDNN-ReLU	t	512
7	TDNN-ReLU	t-4, t, t+4	512
8	TDNN-ReLU	t	512
9	TDNN-ReLU	t	1500
10	Pooling	Full-Seq	2x1500

	Layer	Context 1	Context 2	Skip Conn.	Size
1	TDNN-ReLU	t-2:t+2			512
2	FTDNN-ReLU	t-2, t	t, t+2		1024
3	FTDNN-ReLU	t	t		1024
4	FTDNN-ReLU	t-3,t	t,t+3		1024
5	FTDNN-ReLU	t	t	3	2048
6	FTDNN-ReLU	t-3,t	t,t+3		1024
7	FTDNN-ReLU	t-3,t	t,t+3	2, 4	3072
8	FTDNN-ReLU	t-3,t	t,t+3	4, 6	3072
9	FTDNN-ReLU	t	t		1024
10	Dense	Full-Seq			2048
11	Pooling	Full-Seq			2x2048

Table 5.4 Results with TDNN-based architectures trained on data shown in Table 5.2. For each utterance, three copies are obtained as detailed in Section 5.1.2. Preprocessing and backend are reported in Section 5.1.4

	Params	EER	DCF-08	C_{prim}^{SRE}
TDNN (NGD)	12.67M	0.056	0.242	0.452
+ FineTuning	"	0.051	0.215	0.401
CNN-TDNN	17.23M	0.040	0.188	0.375
+ Large Margin	"	0.034	0.164	0.335
fwSE+CNN-TDNN	17.24M	0.036	0.176	0.353
+ Large Margin	"	0.032	0.149	0.309
FTDNN	20.04M	0.038	0.180	0.375
CNN-FTDNN	24.95M	0.037	0.176	0.350
+ Large Margin	"	0.033	0.156	0.311
ECAPA	8.57M	0.043	0.193	0.371
CNN-ECAPA	13.13M	0.036	0.161	0.320
+ Large Margin	"	0.031	0.141	0.300
fwSE+CNN-ECAPA	13.14M	0.034	0.148	0.306
+ Large Margin	"	0.029	0.130	0.273

Table 5.4 reports the best results achieved with TDNN and its derivations. Based on our findings, the most successful approach for ETDNN was to train first with softmax and cross-entropy loss and then finetuning. In the finetuning step we change the last layer of the network and continue training the whole network with an angular margin loss function, like Additive Margin [42], for a few epochs. This approach produced a 10% decrease in both metrics.

We also tested other TDNN-based architectures including the Factorized-TDNN, which was introduced in Section 3.2.1. We implemented this network following the one proposed in [121] and achieved good results by factorizing the weights after each backward step. This architecture has almost double the number of parameters than the ETDNN, being one of the most expensive network we trained, however, it achieves better results than a fine-tuned TDNN and comparable results with the ECAPA-TDNN.
The ECAPA architecture implementation follows the one provided by the authors of [67] in the speechbrain repository [123]. We conducted our tests with the small version with channel dimension set to 512, trained with Additive Angular Margin

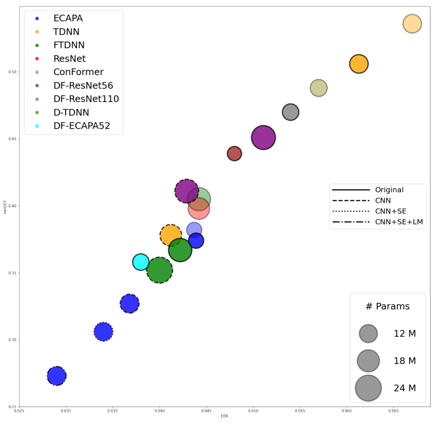


Figure 5.2 Best EER and primary cost for each tested network. The size of the bubble depends on the number of parameters.

[43], with a margin set to 0.2 and scale to 30. Although the ECAPA architecture has fewer parameters than both ETDNN and FTDNN, it performed the best among the base models. Inspired by the results obtained on the language task, we added the 2D-CNN block proposed in [61] to some of our architectures. This block was originally designed for ECAPA but proved to be beneficial also for ETDNN and FTDNN. In Table 5.5, we reported results on Conformer (Section 3.2.3) and DTDNN (Section 3.2.1), and the CNN block seems to improve their performance as well. In the case of ETDNN, the CNN block decreased the EER and primary cost by almost 20% relative. Additionally, the large-margin finetuning approach further improved the results. In this approach, the finetuning step is performed by setting a higher margin in the angular loss and increasing the duration of the training segments. We achieved the best results with 0.5 as margin and 10 seconds segments. Similar improvements were observed for CNN-FTDNN and CNN-ECAPA. The first one didn't improve much with respect to the base architecture, but both finetuned versions achieved better results.

We tested the inclusion of Squeeze & Excitation (Section 3.2.3) components in our models, particularly addressing the CNN block. We added a frequency-wise SE (fwSE) layer in the residual blocks of the CNN module. In this case the AdamW optimizer was needed to obtain better results and more stable training. The addition of the fwSE layer achieved results on par with the finetuned CNN-version of the

Table 5.5 Results with different architectures trained on data shown in Table 5.2. The augmentation strategy in Section 5.1.2 is used to obtain three copies from each utterance. Preprocessing and backend are reported in Section 5.1.4

		EER	DCF-08	C_{prim}^{SRE}
ResNet34 (clean)	17.17M	0.044	0.200	0.398
DF-resnet56	7.46M	0.479	0.219	0.441
DF-ecapa52	9.98M	0.389	0.177	0.358
DTDNN	10.66M	0.066	0.293	0.526
CNN-DTDNN	15.22M	0.054	0.251	0.485
Conformer	21.11M	0.511	0.226	0.450
CNN-Conformer	20.80M	0.042	0.195	0.410

networks. Overall, the best results in our testing were found with a large margin finetuning of the fwSE-CNN-ECAPA.

In our experiments, we also tried other architectures and their results are displayed in Table 5.5. We found that the resnet34-based (Table 3.1) architecture is able to produce competitive results when trained on cleaned data only. However, due to the high number of parameters and computational resources required to train such models, we decided not to pursue this approach. The Conformer model also faced similar issues, and our training didn't lead to comparable results. From the table, we can observe that CNN improves the results of both Conformer and DTDNN, but the final results are still not satisfactory compared to the previous architectures. On the other hand, the depth-first approach showed promising results, with acceptable performance achieved with a relatively low-complexity architecture. In particular, the DF-ECAPA52 architecture implemented according to [69] has 52 layers while maintaining a comparable number of parameters as a normal ECAPA. In Figure 5.2, we plot the best primary cost and EER of all the architectures we tested where the size of each bubble represents their size, in millions of parameters.

Table 5.6 NIST LRE2022 target langauges

Target Languages	North African French
Afrikaans	Ndebele
Tunisian Arabic	Oromo
Algerian Arabic	Tigrinya
Libyan Arabic	Tsonga
South African English	Venda
Indian-accented South African English	Xhosa
Zulu	

5.3 Language Embedding Results

In Section 3.2.5, we presented the architectures and solutions we developed to improve the performance of language detection systems. In this section, we will analyze the results we obtained when dealing with a limited amount of data for some languages. This issue was a significant challenge in the NIST Language Recognition Evaluation 2022, which we used to assess the performance of our models.

5.3.1 NIST Language Recognition Evaluation 2022

The objective of the NIST Language Recognition Evaluation 2022 is to assess the current state of the art in the language detection task. In the LRE 2022 [74] special emphasis was given to the recognition of low-resource languages. The target languages are shown in Table 5.6. Given a speech segment and a specific target language, the system's task is to determine if the target language was spoken in the segment. In the open condition, teams had the freedom to utilize any dataset they deemed necessary. However, our focus was on the fixed condition, which imposed restrictions on the datasets that could be used. The list of datasets approved by NIST for training included the NIST LRE 2017 (LRE17) training and test data [124], as well as VOXlingua-107[4] data. For each of the 14 target languages, the LRE22 Development set provided 30 audio segments, which were the only segments labeled with one of the target languages. Each segment was further divided into 10 sub-segments by NIST, resulting in a development set with 300 audio files per language, with nominal durations ranging from 3 to 93 seconds. Due to the scarcity

[4]http://bark.phon.ioc.ee/voxlingua107/

of data for the target languages, we opted to exclusively employ the development set for training our backend and fusion models.

5.3.2 LRE22 Embedding Extractors

In our experiments, we utilized two distinct subsets of the available dataset for network training. The first subset, indicated as VOX, exclusively relied on VOXlingua-107 as the training data source. In the second, labeled VOX+LRE, we combined the data from LRE17 with data from VOXlingua-107. Specifically, we opted to retain only LRE17 data labeled with languages that appeared both in LRE17 and VOX. This was done because some of the dialects present in LRE17 are present as a single class in VOX.

We employed two types of acoustic features: 46-dimensional Log-Mel and 23-dimensional MFCC. The 46 Log-Mel band parameters were extracted with short time centering (STC) over speech and non-speech audio frames. Similarly, the 23-dimensional MFCCs were obtained using a frame length of 25ms and were mean-normalized over a sliding window. In both cases, we utilized an energy-based voice activity detection (VAD) to eliminate silence. The 23-dimensional MFCCs were used to train a TDNN architecture, while the other architectures were trained using the 46-dimensional Log-Mel. In Section 3.2.5 we introduced the architectures we used as language embedding extractors. These networks, Table 5.7, were trained using 3-second segments and a batch size of 128. Stochastic Gradient Descent (SGD) was our primary optimizer, with a momentum of 0.9 and a weight decay of 10^{-4}, except for the MagNetO, where we opted for a weight decay of 10^{-5}. The management of the learning rate was carried out using various strategies. Cyclic learning rate [119] was the main choice, with a base value of 10^{-5} and a maximum value of 10^{-1}. The maximum learning rate was halved within each epoch, defined as one iteration over the entire training set. An *exponential* schedule was used with a starting learning rate of 0.01 and ending at 0.001, decreasing with a factor of 10 during training. In contrast, a *step* scheduler was used to halve the learning rate at each epoch, starting from 10^{-1}.

Table 5.7 Summary of the proposed architectures for our primary system for LRE 22

Architecture	Params	Loss	Scheduler	Embedding* Size	PCA
CNN-ECAPA	10.7M	ArcFace m=0.2, s=30	Cyclic	192	150
CNN-ETDNN	10.7M	CrossEntropy	Cyclic	512	200
CNN-FTDNN	18.5M	CrossEntropy	Cyclic	512	200
Conformer	18.4M	ArcFace m=0.2, s=30	Cyclic	192	—
TDNN	4.5M	CrossEntropy	Exponential	512	150
MagNetO	28.5M	Arcface m=0.2, s=40	Step	512	100

*All embeddings are taken from the last FC layer, except for CNN-ETDNN where we used the second last FC layer.

5.3.3 LRE22 Backends

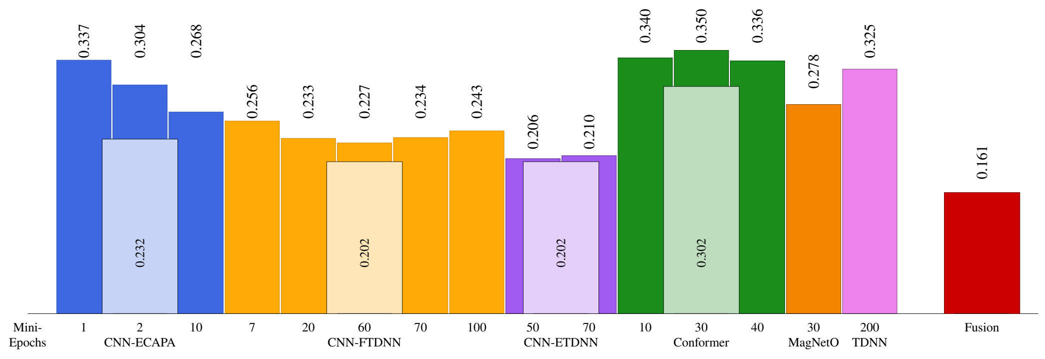
Most of the low-resource target languages are not present in VOXlingua-107 or LRE17 datasets. For this reason, we constructed our validation set by extracting embeddings from the LRE22 Development set. This set served multiple purposes, including testing and selecting our frontend models and training the backend classification, calibration, and fusion models. The dimensionality of the embeddings was reduced using Principal Component Analysis (PCA). The new dimensions were chosen to optimize the overall system and are shown in Table 5.7. The embeddings extracted from each frontend are preprocessed and used to train a Gaussian Linear Classifier (GLC) [125]. We employed multi-class logistic regression [80] for calibration. This included training a single scalar value and a set of language-dependent bias terms to establish a linear mapping of classification scores into class-conditional log-likelihoods. A similar methodology was applied when fusing two or more sub-systems. Our fusion model is based on a linear approach, featuring a scalar value for each sub-system and a bias vector for each language. The LRE22-Dev set is utilized for training the backend classifier and both the calibration and fusion models while also serving as evaluation data for the different systems. To maximize the use of this data, we designed a custom leave-one-out (LOO) approach. Given the structure of the LRE22 development set, the first step consisted of the aggregation of the segments that belong to the same audio file, obtaining 30 groups for each language. At this point:

- For each language, the process begins by shuffling groups and, iteratively, removing one group of segments. During each iteration, 14 groups and 140 segments are collected to train the model $\mathcal{M}_i$ while the removed segments are scored with it.
- The resulting scores of each iteration are pooled together to train fusion models. Additionally, the entire development dataset is utilized to train model $\mathcal{M}_P$. The evaluation segments are scored with this model.
- The first two steps are applied to the pooled scores to train calibration and/or fusion models. A different seed is used for reshuffling the groups. For a given iteration i, we train a calibration model C_i or a fusion model $\mathcal{F}_i$ for each sub-system or combination of sub-systems. These models are then applied to the left-out raw scores.
- The performance over the development set reported in Figure 5.3, is assessed on the pooled calibrated or fused scores. The full set of raw scores is utilized to train the final fusion model $\mathcal{F}_P$, which is subsequently applied to the evaluation scores obtained by the $\mathcal{M}_P$ backend.

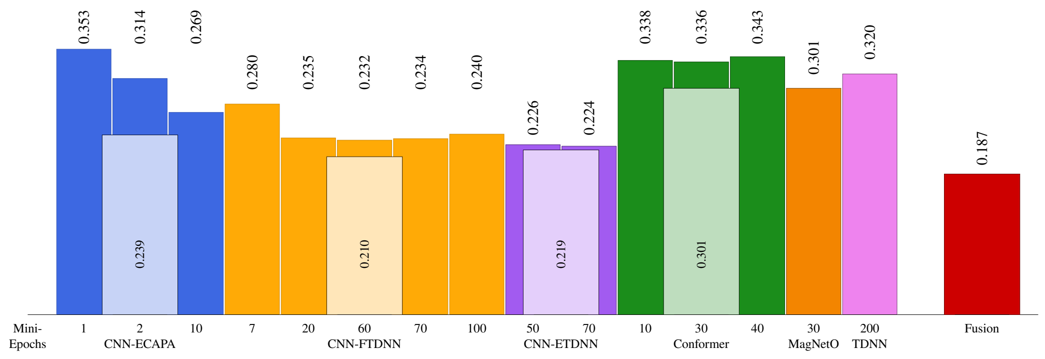
5.3.4 LRE22 Results

At the beginning of our experiment, we observed that some extractors achieved optimal performance very early in the training process, often within just a few epochs. To account for this, we divided each epoch into 10 mini-epochs and evaluated the performance of different networks based on the number of mini-epochs. Our submission relied on the networks introduced in Section 3.2.5 and detailed in Table 5.7.

For each sub-system, we provide the actual primary cost (C_{prim}) [126] in Figure 5.3. These values are derived from both the LRE22 evaluation and development sets using the LOO procedure detailed in Section 5.3.3. The Net Fusion results involve



(a) C_{prim} on the development set results



(b) C_{prim} on the eval set results

Figure 5.3 Comparison of individual sub-system performance. Lighter bars refer to the fusion of the same architecture at various stages (mini-epochs) of training. The red bar shows our primary submission system results.

fusing embeddings extracted from the same architecture at various stages (mini-epochs) of training. These results reveal that, for most architectures, embeddings from different mini-epochs offer complementary information, leading to improved results in both the evaluation and development sets with respect to the single embedding. The right most bar of Figure 5.3b represents the result of our submission, which combines all sub-systems. This submission, referred to as "T2" in [127], demonstrates competitive performance.

We conducted an ablation study on our submission to asses the individual contributions of each sub-system to the primary fusion. In Figure 5.4, we present the results obtained by progressively removing each sub-system from our primary

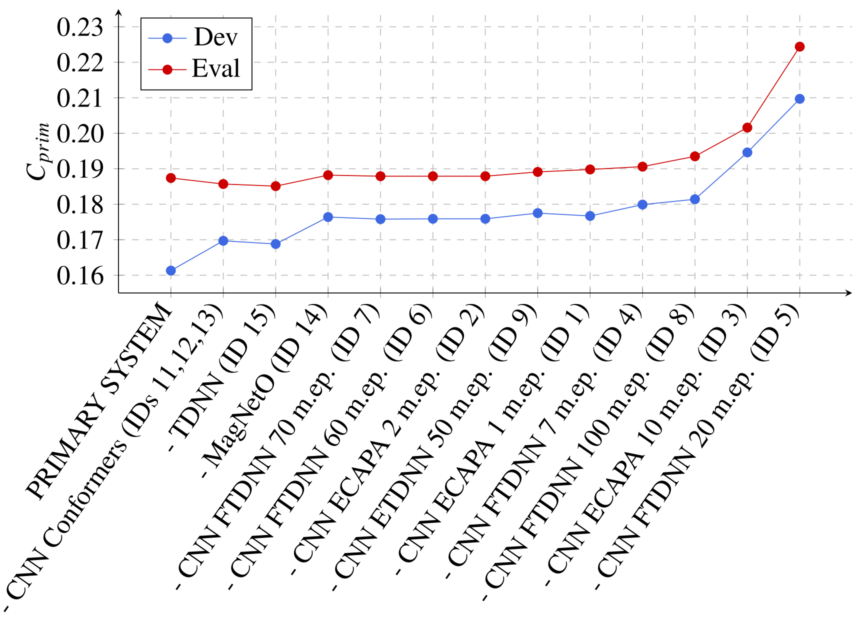


Figure 5.4 Results of our ablation study on the primary submission, which includes all systems. From left to right on the x-axis it shows the *incremental* removal of the least contributing sub-systems based on their evaluation results. The right most system corresponds to the CNN-ETDNN sub-system (ID 10).

fusion. We systematically removed the sub-system that had the least impact on the actual C_{prim} in the evaluation set. The first column on the left represents the fusion of all sub-systems, and moving from left to right, we present the results after incrementally removing the sub-system(s) with the lowest contribution. First of all, this study shows that our primary submission achieves results close to optimal for the eval set. The inclusion of the TDNN (ID 15) and Conformer (ID 11, 12, 13) improved results for the development while slightly degrading the results for the eval set. We can also notice that a smaller set of sub-systems would have achieved similar results. Nonetheless, the inclusion of a large number of systems does not lead to significant overfitting, as the results of development and evaluation data remain largely consistent, highlighting the effectiveness of our training approach.

Finally, we examine the impact of the CNN block described in Section 3.2.5. Figure 5.5 depicts the performance of different frontends across various mini-epochs. Generally, it is evident that the early stages of training yield already good results. Notably, in the cases of ECAPA and conformers, a limited number of mini-epochs suffices, as further training leads to performance degradation due to overfitting. Consistently across all models, the 2D convolution within the CNN block proves to

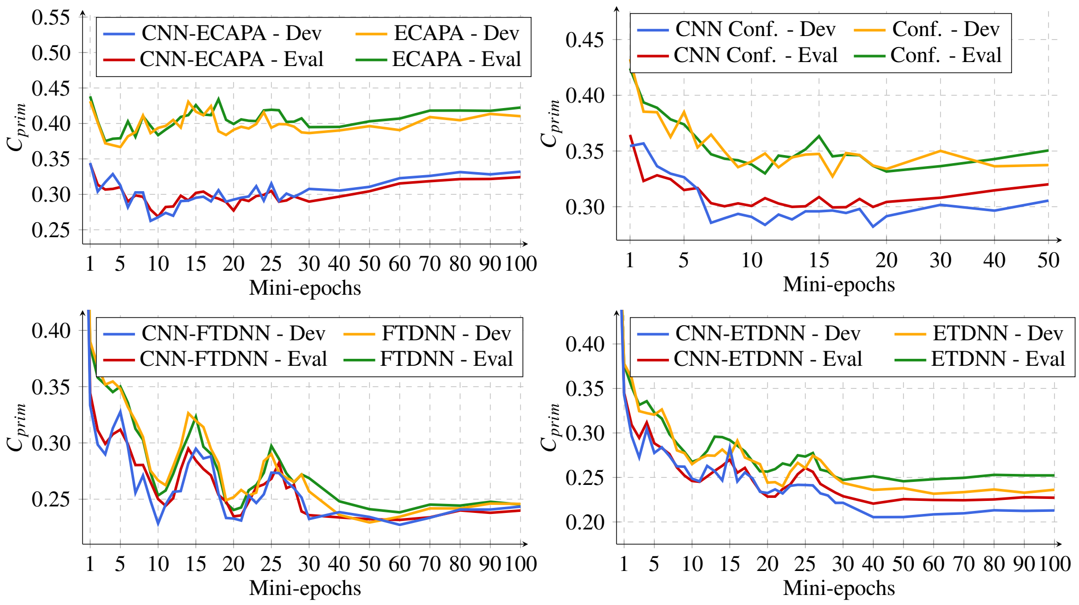


Figure 5.5 Mini-epochs performance for different architectures (Top-Left: ECAPA, Top-Right: Conformer, Bottom-Left: FTDNN, Bottom-Right: ETDNN).

enhance performance. It is important to note, that also in this case, the results on both the development and evaluation sets remain consistent.

5.4 Calibration and Normalization Results

In this section we discuss the results obtained by our calibration and normalization models. We begin with the results of the VΓ-Var models, detailed in Section 4.2.2. Following this, we analyze the results of the C-norm, achieved by integrating score normalization statistics into the Log-Reg calibration model (see Section 4.3.1). Lastly, we present our findings regarding the Adaptive Data Normalization (AD-norm) technique in Section 4.3.2. These results are part of our work published in [94, 95, 109, 110].

Backends classifiers

The backend classifiers we chose to test our models are based on the PLDA and PSVM models described in Section 4.1. The training process for these classifiers relied mainly on embeddings extracted with a DNN frontend. Specifically, we utilized a TDNN architecture, and two derivations namely the Factorized TDNN (FTDNN) and the ECAPA architecture. To evaluate the models, we utilized datasets from NIST speaker recognition evaluations (SRE), including the SRE 2019[5], SRE 2016, SRE 2012[6] and SRE 2010[7] datasets. Additionally, we considered the Speaker in The Wild dataset (SITW) [128].

5.4.1 The $V\Gamma$-Var Model

To show the effectiveness of our approach across a large variety of embeddings and classifiers, we have employed different frontends and analyzed different backends for the different datasets

SRE 2019 embeddings are extracted using TDNN, Factorized TDNN and ECAPA architectures. The TDNN, based on the topology introduced in [50], employs 24-dimensional Perceptual Linear Predictors (PLP) as input features. Details on the FTDNN and ECAPA, as well as the training dataset, can be found in Section 5.2. While TDNN and ECAPA have been trained using augmented data, the FTDNN was trained using only the original audio. TDNN and FTDNN embeddings are 512-dimensional, processed with LDA to reduce dimensions to 400 for PSVM and 200 for PLDA. ECAPA embeddings are 192-dimensional with no dimensionality reduction. Length normalization is tested as an additional preprocessing step for both backends. Furthermore, Within-Class Covariance Normalization (WCCN) normalization is applied for PSVM on both raw and length-normalized embeddings. SRE 2018 Evaluation data, which contains about 13 thousand segments, has been used to train the PLDA model. For PSVM we added a subset of Mixer 4, 5, and 6, Switchborad and VoxCeleb to the training list, increasing the training data to 110 thousand segments. This is necessary since PSVM is more effective when trained with a large amount of data. The selection of the training lists is based on the best results of each baseline model on SRE 2019 Progress data. A subset of the SRE 2019 Progress set was used to train the calibration models, while their performance were evaluated on the SRE 2019 Evaluation set.

[5]https://www.nist.gov/publications/2019-nist-speaker-recognition-evaluation-cts-challenge
[6]https://www.nist.gov/itl/iad/mig/sre12-results
[7]https://www.nist.gov/itl/iad/mig/speaker-recognition-evaluation-2010

SRE2012 An hybrid GMM/DNN model [13, 129] was used for the SRE 2012 system, trained on approximately 42,000 segments from SRE-04 to SRE-10 and Switchboard. Acoustic features consist of 20 PLP coefficients with their delta and delta-delta parameters. The outputs of the DNN are 256-dimensional and an 8-dimensional GMM is trained over each output, as explained in [130]. The resulting UBM has 2048 components. Speaker vectors are obtained from a 400-dimensional e-vector extractor [131], which combines the speaker subspace estimated by the JFA and the i-vector framework. Backends are trained using the same frontend datasets. Tests are performed on the extended tel-tel core condition (condition 5), with calibration parameters estimated on 25% of enrollment segments, while the remaining part is used as evaluation data.

SRE2010 system relies on 400-dimensional i-vectors, estimated from a gender-dependent UBM with 1024 components and diagonal covariance. The input features are 45-dimensional MFCC, incorporating delta and double-delta parameters. PLDA was used as a backend, and whitening and length normalization were applied to pre-process the i-vectors. The frontend is trained with data from Switchboard, Fisher and SRE-04 to SRE-06 datasets. The same data, except the Fisher dataset, was used to train the backend. Tests are conducted on the female extended tel-tel condition (condition 5). Also in this case calibration parameters are estimated on 25% of enrollment segments, while evaluation is performed on the remaining part.

To tackle the issue of non-zero evaluation population means, all test sets calibration and evaluation embeddings have been recentered with respect to the calibration set embedding mean. This allows better coping with the assumption in Section 4.2.1, i.e. that the evaluation population has a zero-mean distribution.

SRE2019 results

In order to evaluate the effectiveness of duration-aware models, it is important to have utterances with variable and short durations. To accomplish this, we randomly selected segments ranging from 3 to 30 seconds in length from both the enrollment and test segments of SRE 2019.

We present the performance of various calibration models in terms of C_{llr} (as defined in [84, 82, 85]) for different frontend and backend combinations in Figure 5.6 and Table 5.8. We report the actual primary cost C_{prim} as defined by NIST for the SRE19 evaluation in Table 5.8. Furthermore, since duration-aware models are generally more discriminant, we also show the Equal Error Rate (EER). We report the *min costs* obtained with the output of the original classifier in the first row of each table. This is followed by the performance of four calibration models that do not use side information. The baseline models include the prior-weighted Logistic Regression model (Log-Reg) and the linear, Constrained ML Variance-Gamma model of [89] (Linear VΓ). The other two models are our own, (4.77), trained with a generative ML criterion (4.78) and a discriminative objective (4.80). The target weight of the generative models is set to $\zeta = 0.1$. The target prior for the discriminative models is set to $\pi = 0.1$.

In the last three rows of the table, we present the results obtained with duration-aware models. In this case, the baseline model is a linear Log-Reg calibration model (4.37) enriched with quality measures (QM),

$$f_{cal}(s, q_1, q_2) = \alpha s + \beta + QM(q_1, q_2) \tag{5.1}$$

where QM accounts for the effects of q_1 and q_2. Following [90], we choose QM_4 defined as

$$QM_4(d_e, d_t) = w_1 (log\frac{d_e}{d_t} + log\frac{d_t}{d_c})^2 - w_2 (log\frac{d_e}{d_t} - log\frac{d_t}{d_c})^2 \tag{5.2}$$

where d_c, w_2 and w_3 are additional parameters, while d_e and d_t are the durations of the enrollment and test segment. This QM achieved the best calibration results in our tests. The yellow background columns show the results obtained after applying length normalization to the embeddings, while the blue background refers to embeddings without length normalization.
The results in Table 5.8 show that for the baseline system based on ECAPA and TDNN, the PSVM backend outperforms the PLDA one, both when length normalization is applied and not. However, for FTDNN embeddings, when length normalization is applied, PSVM produces similar C_{llr} and EER to the PLDA but slightly underperforms in terms of C_{prim}, while better C_{llr} and EER are achieved when no length normalization is applied. Length normalization is beneficial for PLDA in every case, but it does not seem to affect much PSVM, and it produces worse

Table 5.8 Results on the SRE 2019 evaluation dataset with short segments. Calibration models have been trained with target prior $\pi = 0.1$ (discriminative models) or target weight $\zeta = 0.1$ (generative models).

(a) ECAPA embeddings

	PLDA						PSVM					
	with L2-norm			w/o L2-norm			with L2-norm			w/o L2-norm		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs[†]	*0.246*	*0.485*	*6.8%*	*0.296*	*0.486*	*8.7%*	*0.195*	*0.448*	*5.3%*	*0.204*	*0.452*	*5.6%*
Log-Reg	0.264	0.495	6.8%	0.333	0.571	8.7%	0.202	0.457	5.3%	0.213	0.466	5.6%
Linear VΓ	0.270	0.488	6.8%	0.339	0.551	8.7%	0.203	0.456	5.3%	0.218	0.458	5.6%
VΓ-Var (Gen)	0.248	0.486	6.8%	0.301	0.487	8.7%	0.196	0.449	5.3%	0.205	0.454	5.6%
VΓ-Var (Disc)	0.250	0.485	6.8%	0.299	0.487	8.7%	0.197	0.449	5.3%	0.206	0.454	5.6%
Log-Reg + QM_4	0.245	0.493	6.1%	0.282	0.532	6.7%	0.185	0.467	4.8%	0.186	0.484	4.7%
VΓ-Var + Dur (Gen)	0.230	0.485	6.1%	0.251	0.484	6.6%	0.180	0.447	4.8%	0.175	0.454	4.6%
VΓ-Var + Dur (Disc)	0.228	0.478	6.1%	0.243	0.461	6.6%	0.180	0.448	4.8%	0.174	0.454	4.6%

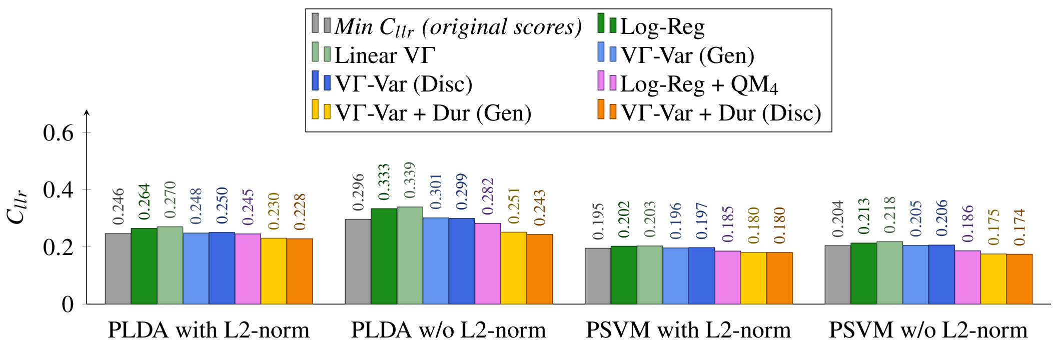
(b) FTDNN embeddings

	PLDA						PSVM					
	with L2-norm			w/o L2-norm			with L2-norm			w/o L2-norm		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs[†]	*0.246*	*0.466*	*7.0%*	*0.307*	*0.477*	*9.0%*	*0.245*	*0.536*	*7.0%*	*0.255*	*0.529*	*7.1%*
Log-Reg	0.259	0.482	7.0%	0.335	0.564	9.0%	0.250	0.550	7.0%	0.270	0.553	7.1%
Linear VΓ	0.263	0.474	7.0%	0.343	0.531	9.0%	0.251	0.544	7.0%	0.276	0.534	7.1%
VΓ-Var (Gen)	0.247	0.467	7.0%	0.308	0.477	9.0%	0.246	0.537	7.0%	0.256	0.531	7.1%
VΓ-Var (Disc)	0.248	0.466	7.0%	0.308	0.477	9.0%	0.246	0.537	7.0%	0.256	0.530	7.1%
Log-Reg + QM_4	0.245	0.486	6.5%	0.250	0.520	6.4%	0.240	0.558	6.6%	0.241	0.591	6.3%
VΓ-Var + Dur (Gen)	0.234	0.468	6.4%	0.225	0.457	6.1%	0.246	0.537	7.0%	0.214	0.518	5.9%
VΓ-Var + Dur (Disc)	0.231	0.465	6.4%	0.225	0.454	6.1%	0.246	0.537	7.0%	0.215	0.520	5.9%

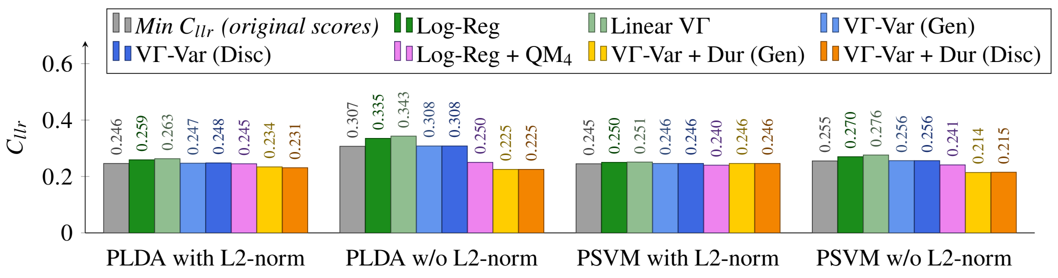
(c) TDNN embeddings

	PLDA						PSVM					
	with L2-norm			w/o L2-norm			with L2-norm			w/o L2-norm		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs[†]	*0.285*	*0.514*	*7.9%*	*0.349*	*0.532*	*10.2%*	*0.226*	*0.480*	*6.1%*	*0.217*	*0.462*	*5.9%*
Log-Reg	0.331	0.528	7.9%	0.443	0.652	10.2%	0.239	0.486	6.1%	0.228	0.476	5.9%
Linear VΓ	0.341	0.517	7.9%	0.451	0.623	10.2%	0.246	0.480	6.1%	0.231	0.464	5.9%
VΓ-Var (Gen)	0.292	0.526	7.9%	0.367	0.536	10.2%	0.227	0.480	6.1%	0.219	0.463	5.9%
VΓ-Var (Disc)	0.288	0.526	7.9%	0.358	0.537	10.2%	0.228	0.481	6.1%	0.219	0.464	5.9%
Log-Reg + QM_4	0.314	0.529	7.2%	0.403	0.613	8.3%	0.226	0.488	5.7%	0.198	0.511	5.0%
VΓ-Var + Dur (Gen)	0.275	0.537	7.2%	0.309	0.536	8.3%	0.216	0.487	5.7%	0.190	0.465	4.9%
VΓ-Var + Dur (Disc)	0.267	0.536	7.1%	0.297	0.534	8.2%	0.213	0.479	5.6%	0.187	0.472	5.0%

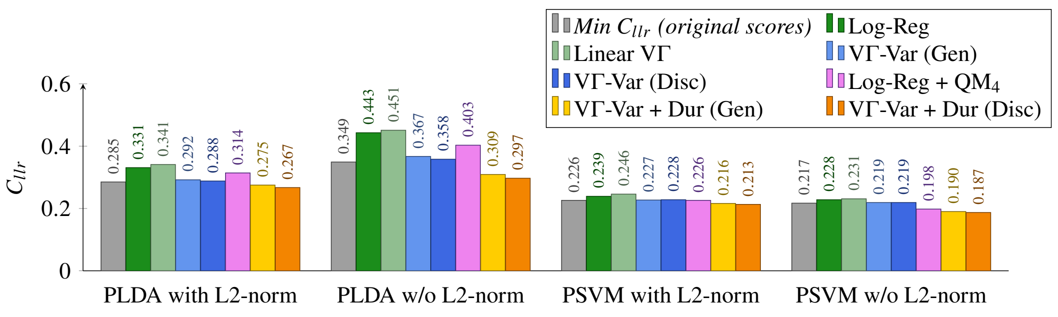
[†] Minimum C_{llr}, minimum C_{prim}, and EER computed on the classifier scores. Models using side-information may provide lower costs.



(a) ECAPA embeddings



(b) FTDNN embeddings

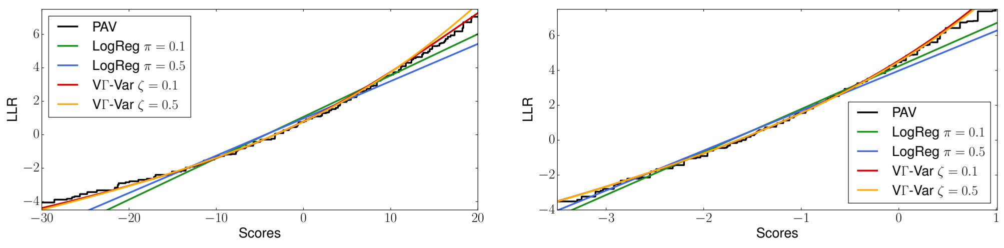


(c) TDNN embeddings

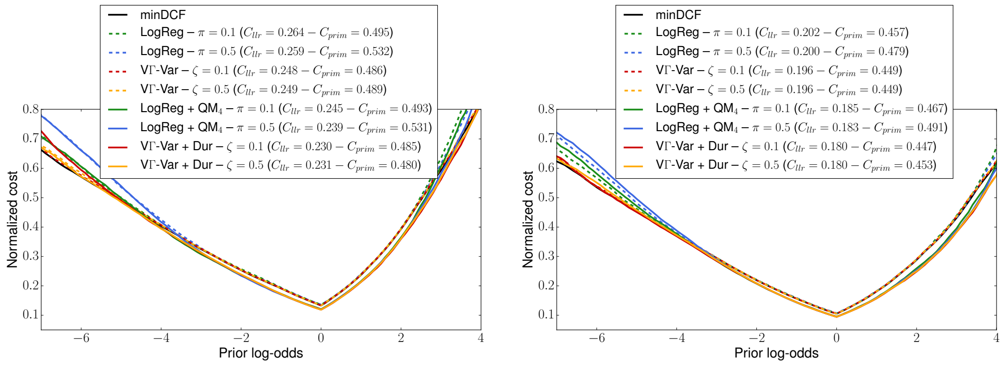
Figure 5.6 C_{llr} for different calibration approaches with different embedding frontends on SRE 2019 short segments. Calibration models have been trained with target prior $\pi = 0.1$ (discriminative models) or target weight $\zeta = 0.1$ (generative models).

results for the TDNN. Both models exhibit some miscalibration error, especially for the TDNN-PLDA one, and it is particularly evident in the PLDA models trained using non-normalized embeddings.

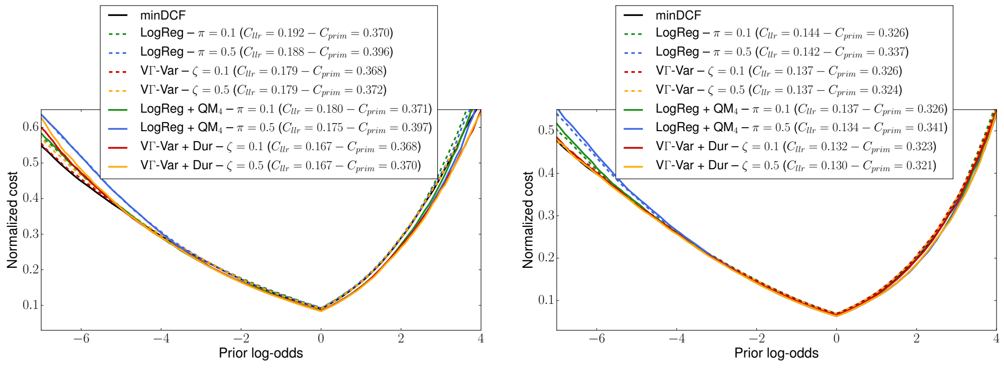
In Figure 5.7a we show different calibration transformations of the score compared to the optimal calibration of the evaluation dataset obtained by isotonic regression computed through the PAV [132] algorithm, applied to the length-normalized embeddings extracted from the ECAPA frontend. This figure shows that an optimal calibration cannot be approximated by a linear transformation over the whole range of scores. Instead, our proposed VΓ-Var model is designed to handle the distribution mismatches by computing a non-linear mapping, which provides a better approximation of the calibration defined by the PAV transformation. This translates to actual costs closer to the minimum ones, both for PLDA and PSVM, with and without length normalization. Furthermore, from the 4th and 5th rows of Table 5.8, it can be seen that the results obtained with the generative model are close to the ones obtained with the discriminative model, meaning that the VΓ-Var can effectively capture the characteristics of target and non-target score distributions. Repeating this observation for the non-normalized embeddings the non-linearity becomes even more evident, leading to a significant calibration loss of the linear model. Figure 5.7b shows the Bayes Error plots for the ECAPA frontend, both for models with and without side-information. From Figure 5.7b we can notice the effects of changing target prior π or target weight ζ. Models without side-information are reported with a dashed line. We can notice that the choice of the prior affects the Logistic Regression performance. Lower C_{llr} can be obtained by setting $\pi = 0.5$, but with the result of having a significantly higher actual primary cost C_{prim}. In contrast, the VΓ-Var models are more robust to the choice of ζ, and similar results are obtained with $\zeta = 0.1$ and $\zeta = 0.5$. As expected the duration aware models outperform the duration agnostic models. Both Log-Reg + QM$_4$ and the proposed VΓ-Var + Dur achieve better calibration and discrimination for almost all frontend and backend combinations, as evidenced by the lower C_{llr} and EER. In particular, looking at the C_{llr}, the VΓ-Var + Dur outperforms the quality measures methods, with relative improvements ranging from 3% to 15% for length-normalized embeddings and 6% to 26% for non-normalized ones, with similar or slightly better EER values. Although in the case of FTDNN with PSVM, Log-Reg+QM$_4$ achieves a 2% improvement, the best results for this frontend are achieved using our VΓ-Var + Dur approach with non-normalized embeddings. For the C_{prim} metric, the Log-Reg+QM$_4$ model produces irregular results. These results are close to or slightly worse than the linear Log-Reg when embeddings are length-normalized. For non-normalized embeddings, the model achieves better performance with PLDA but significantly worse with PSVM



(a) Calibration transformation for short segments - duration-agnostic models - left: PLDA, right: PSVM



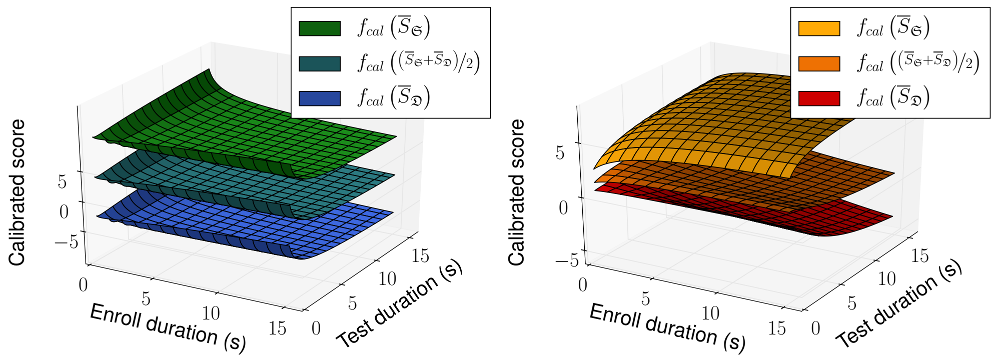
(b) Bayes error plot for short segments - left: PLDA, right: PSVM



(c) Bayes error plot for original segments - left: PLDA, right: PSVM

Figure 5.7 Calibration transformation and Bayes error plots for different calibration models and different backends on SRE 2019 evaluation data, ECAPA frontend.

models. In comparison, our approach provides better C_{prim} than the Log-Reg+QM$_4$, although on average, it is close to the C_{prim} obtained with our VΓ-Var model. Again, the Bayes Error plot 5.7a helps us see that the Log-Reg+QM$_4$ is sensitive to the prior choice, while our model provides consistent results both for $\zeta = 0.1$ and $\zeta = 0.5$.



left: Log-Reg + QM$_4$ - right: VΓ-Var + Dur (Gen)

Figure 5.8 Calibration transformations of duration-aware models for different, fixed scores as a function of segment duration. ECAPA embeddings without length normalization, PLDA backend.

Figure 5.8 shows the profound difference between the transformations defined by Log-Reg+QM$_4$ and the VΓ-Var+Dur models. The transformation is shown as a function of the enrollment and test duration for a given, fixed verification score. The figure displays three surfaces that correspond to the average target score $\overline{S}_\mathfrak{G}$, the average non-target score $\overline{S}_\mathfrak{D}$, and their mean $\frac{\overline{S}_\mathfrak{G}+\overline{S}_\mathfrak{D}}{2}$ for each model. While the ECAPA-PLDA combination with non-normalized embeddings produces the most visible differences, similar outcomes are obtained with other backends. The figure shows that Log-Reg+QM$_4$ applies a transformation that does not consider the score value, since the quality measures act as an additive term to the score. In contrast, the VΓ-Var models apply a transformation that is different for different values of the score. Shorter durations lead to scores mapped to LLR closer to zero, as they inherently possess greater uncertainty, which is not taken into consideration at the classification level [99]. The VΓ-Var model compensates for the relative overconfidence that backends show when treating shorter utterances. We can also see that length-normalization degrades the performance of duration-aware models. This confirms our assumption that length normalization normalizes the within-class variance of the embeddings, which reduces the variability attributed to duration. This decreases the effectiveness of duration-aware models since their objective is to exploit these effects to increase accuracy. We can also observe that accounting for duration variability at the calibration stage, rather than relying on length normalization, produces better results with the PSVM backend.

Table 5.9 and Figure 5.7c show the results of the different calibration approaches for the original SRE 2019 segments. The results align with those obtained from the short segments, although, as expected, the duration-aware models exhibit relatively smaller improvements. Furthermore, we report the results obtained with the VΓ-Var + Norm model (4.88). This set of experiments analyzes the effectiveness of accounting for the embedding norm at calibration, rather than at the classification level. We compare the results obtained with VΓ-Var calibration for length-normalized backends with the ones obtained with VΓ-Var + Norm for non-normalized backends in Table 5.10. In both cases, the embedding norm is computed from the entire vector. The results in yellow are obtained from normalized embeddings using VΓ-Var calibration, while those in blue refer to raw embeddings and VΓ-Var + Norm model. Usually, length normalization improves the results of PLDA, but for this dataset, we can see that encoding the embedding norm at the calibration level rather than at score level produces better results.

The VΓ-Var + Norm model in combination with non-normalized embeddings achieves significantly lower C_{llr} and EER for short segments for the PLDA, maintaining a similar actual C_{prim} cost. Longer segments see no significant improvement over length-normalized backends, but VΓ-Var + Norm shows slightly lower EER. The C_{llr} and EER are on average significantly reduced also for the PSVM when the norm of the embeddings is incorporated at the calibration level. The approach is also effective in reducing the C_{prim} of the FTDNN and TDNN embeddings, while the ECAPA frontend produces similar results. It is important to highlight that using length-normalized embeddings with the VΓ-Var + Norm would not produce any improvement with respect to the VΓ-Var model. This is because the norm that should be applied at the calibration level is the norm of the input vectors of the PLDA model, which corresponds to the norm of the length-normalized vectors. Since this norm is the same for all the embeddings, the VΓ-Var and VΓ-Var + Norm models would produce the same results. We also attempted to extend our model by combining both duration and embedding norm variability as

$$w_{S,i} = w_C + w_{dur,i} + w_{norm,i} \tag{5.3}$$

Table 5.9 Results on the SRE 2019 evaluation dataset with original segments. Calibration models have been trained with target prior $\pi = 0.1$ (discriminative models) or target weight $\zeta = 0.1$ (generative models).

(a) ECAPA embeddings

	PLDA						PSVM					
	with L2-norm			w/o L2-norm			with L2-norm			w/o L2-norm		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs[†]	*0.176*	*0.365*	*4.6%*	*0.204*	*0.358*	*5.6%*	*0.135*	*0.323*	*3.5%*	*0.135*	*0.330*	*3.5%*
Log-Reg	0.192	0.370	4.6%	0.237	0.405	5.6%	0.144	0.326	3.5%	0.146	0.334	3.5%
Linear VΓ	0.200	0.366	4.6%	0.245	0.388	5.6%	0.147	0.326	3.5%	0.151	0.331	3.5%
VΓ-Var (Gen)	0.179	0.368	4.6%	0.210	0.360	5.6%	0.137	0.326	3.5%	0.137	0.334	3.5%
VΓ-Var (Disc)	0.179	0.367	4.6%	0.206	0.360	5.6%	0.139	0.326	3.5%	0.137	0.335	3.5%
Log-Reg + QM_4	0.180	0.371	4.3%	0.217	0.395	5.0%	0.137	0.326	3.2%	0.137	0.340	3.1%
VΓ-Var + Dur (Gen)	0.167	0.368	4.3%	0.190	0.356	5.0%	0.132	0.323	3.3%	0.128	0.340	3.1%
VΓ-Var + Dur (Disc)	0.165	0.362	4.2%	0.186	0.350	5.0%	0.130	0.324	3.2%	0.127	0.334	3.1%

(b) FTDNN embeddings

	PLDA						PSVM					
	with L2-norm			w/o L2-norm			with L2-norm			w/o L2-norm		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs[†]	*0.163*	*0.326*	*4.3%*	*0.185*	*0.329*	*5.0%*	*0.150*	*0.366*	*3.9%*	*0.145*	*0.350*	*3.7%*
Log-Reg	0.176	0.331	4.3%	0.202	0.354	5.0%	0.158	0.370	3.9%	0.155	0.358	3.7%
Linear VΓ	0.182	0.327	4.3%	0.212	0.334	5.0%	0.160	0.367	3.9%	0.161	0.352	3.7%
VΓ-Var (Gen)	0.165	0.330	4.3%	0.186	0.330	5.0%	0.152	0.367	3.9%	0.147	0.351	3.7%
VΓ-Var (Disc)	0.165	0.329	4.3%	0.186	0.331	5.0%	0.152	0.367	3.9%	0.146	0.353	3.7%
Log-Reg + QM_4	0.165	0.330	4.0%	0.172	0.349	4.3%	0.150	0.376	3.6%	0.144	0.374	3.4%
VΓ-Var + Dur (Gen)	0.155	0.332	4.0%	0.160	0.326	4.2%	0.146	0.372	3.7%	0.134	0.354	3.3%
VΓ-Var + Dur (Disc)	0.153	0.329	3.9%	0.157	0.316	4.2%	0.143	0.370	3.6%	0.133	0.353	3.3%

(c) TDNN embeddings

	PLDA						PSVM					
	with L2-norm			w/o L2-norm			with L2-norm			w/o L2-norm		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs[†]	*0.200*	*0.387*	*5.2%*	*0.236*	*0.396*	*6.4%*	*0.164*	*0.366*	*4.2%*	*0.140*	*0.339*	*3.6%*
Log-Reg	0.231	0.388	5.2%	0.287	0.438	6.4%	0.177	0.370	4.2%	0.150	0.340	3.6%
Linear VΓ	0.246	0.392	5.2%	0.301	0.412	6.4%	0.189	0.370	4.2%	0.154	0.341	3.6%
VΓ-Var (Gen)	0.206	0.407	5.2%	0.244	0.403	6.4%	0.165	0.366	4.2%	0.142	0.341	3.6%
VΓ-Var (Disc)	0.204	0.403	5.2%	0.238	0.406	6.4%	0.165	0.367	4.2%	0.141	0.344	3.6%
Log-Reg + QM_4	0.218	0.391	4.8%	0.267	0.435	5.8%	0.169	0.366	4.0%	0.140	0.349	3.2%
VΓ-Var + Dur (Gen)	0.195	0.420	4.9%	0.223	0.409	5.8%	0.160	0.381	4.0%	0.133	0.351	3.2%
VΓ-Var + Dur (Disc)	0.191	0.431	4.8%	0.215	0.398	5.7%	0.155	0.363	3.9%	0.130	0.345	3.2%

[†] Minimum C_{llr}, minimum C_{prim}, and EER computed on the classifier scores.
Models using side-information may provide lower costs.

Table 5.10 Comparison of length-normalization-aware calibration models on the SRE 2019 evaluation dataset.

		Short segments						Original segments					
		PLDA			PSVM			PLDA			PSVM		
L2-norm	Calibration	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
ECAPA													
✓	VΓ-Var (Gen)	0.248	0.486	6.8%	0.196	0.449	5.3%	0.179	0.368	4.6%	0.137	0.326	3.5%
✗	VΓ-Var + Norm (Gen)	0.229	0.483	5.9%	0.188	0.453	4.9%	0.175	0.363	4.3%	0.135	0.329	3.3%
FTDNN													
✓	VΓ-Var (Gen)	0.247	0.467	7.0%	0.246	0.537	7.0%	0.165	0.330	4.3%	0.152	0.367	3.9%
✗	VΓ-Var + Norm (Gen)	0.222	0.457	6.1%	0.215	0.510	5.9%	0.160	0.319	4.1%	0.135	0.345	3.4%
TDNN													
✓	VΓ-Var (Gen)	0.292	0.526	7.9%	0.227	0.480	6.1%	0.206	0.407	5.2%	0.165	0.366	4.2%
✗	VΓ-Var + Norm (Gen)	0.275	0.536	6.9%	0.198	0.464	5.1%	0.211	0.406	5.1%	0.141	0.337	3.4%

where $w_{dur,i}$ and $w_{norm,i}$ are the duration and norm components used in VΓ-Var + Dur and VΓ-Var + Norm models. However, for SRE 2019, the addition of both terms does not provide additional benefits, suggesting that both terms encode similar side information.

Table 5.11 Results on SRE 2010 with an i-vector frontend.

	Short segments						Original (long) segments		
	PLDA			FPD-PLDA			PLDA		
	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER	C_{llr}	C_{prim}	EER
Min. costs	*0.288*	*0.756*	*8.1%*	*0.251*	*0.764*	*7.3%*	*0.094*	*0.420*	*2.5%*
Log-Reg	0.297	0.818	8.1%	0.260	0.809	7.3%	0.098	0.444	2.5%
Linear VΓ	0.298	0.835	8.1%	0.260	0.818	7.3%	0.099	0.460	2.5%
VΓ-Var (Gen)	0.292	0.791	8.1%	0.256	0.797	7.3%	0.097	0.430	2.5%
VΓ-Var (Disc)	0.292	0.784	8.1%	0.257	0.774	7.3%	0.097	0.425	2.5%
Log-Reg + QM_4	0.261	0.827	7.0%	0.260	0.832	7.2%	0.098	0.451	2.5%
VΓ-Var + Dur (Gen)	0.259	0.755	7.0%	0.256	0.798	7.3%	0.097	0.430	2.5%
VΓ-Var + Dur (Disc)	0.253	0.787	7.0%	0.257	0.774	7.3%	0.097	0.425	2.5%

SRE2010 results

The original utterances of the SRE10 test have mostly long duration. As we did for SRE 2019, to obtain short segments, we cut the utterances randomly with durations between 3 and 60 seconds. The duration of the sample is strongly related to the uncertainty of the i-vector estimate. Additionally, we can obtain a measure of such

Table 5.12 Results on SRE 2010 with an i-vector frontend and length normalization.

PLDA L2-norm	Calibration Model	C_{llr}	C_{prim}	EER
	Original (long) segments			
✓	VΓ-Var (Gen.)	0.097	0.430	2.5%
✗	Log-Reg	0.173	0.612	4.3%
✗	VΓ-Var (Gen.)	0.167	0.491	4.3%
✗	VΓ-Var + Norm (Gen.)	0.102	0.501	2.6%
	Short segments			
✓	VΓ-Var (Gen.)	0.292	0.791	8.1 %
✗	Log-Reg	0.427	0.971	12.7%
✗	VΓ-Var (Gen.)	0.415	0.914	12.7%
✗	VΓ-Var + Norm (Gen.)	0.305	0.789	8.7%
✓	VΓ-Var + Dur (Gen.)	0.259	0.755	7.0%
✗	Log-Reg + QM_4	0.345	0.932	9.5%
✗	VΓ-Var + Dur (Gen.)	0.333	0.874	9.4%
✗	VΓ-Var + Norm + Dur (Gen.)	0.282	0.790	8.0%

uncertainty in the i-vector framework. For this set of experiments, we consider the Full-Posterior-Distribution PLDA (FPD-PLDA) model [99] as an additional baseline for the short segments. In Table 5.11 we report the results obtained on this dataset. As before, the minimum cost for the baseline models, both for short and original segments, is shown in the first row. The two baselines achieve similar C_{prim} for short segments, but the FPD-PLDA produces lower C_{llr} and EER than the PLDA. In the next four rows, the results achieved by the duration-agnostic calibration are shown. The best results in terms of C_{prim} are obtained by the VΓ-Var models, while C_{llr} and EER are similar between the two models, a behavior that we already noticed with SRE19. In terms of duration-aware models, the results for short segments show that Log-Reg + QM_4 achieves lower C_{llr} and EER compared to the standard Log-Reg. We also observe a marginal degradation of the C_{prim} in both cases, however, given the small number of evaluation trials, this may not be statistically significant. In contrast, our models, VΓ-Var + Dur achieve better perfomrance than both Log-Reg + QM_4 and VΓ-Var both in terms of C_{llr} and C_{prim}, while providing similar EER.

Furthermore, it is worth noting that the results achieved by the PLDA and VΓ-Var + Dur and the ones obtained with FPD-PLDA and VΓ-Var are very close. This suggests, once again, that our duration model is able to incorporate embedding uncertainty directly at calibration level obtaining similar results to the combination of uncertainty models and PLDA backends. Furthermore, while both approaches show similar performances, our model eliminates the need for an explicit, and sometimes hard-to-define, model for embedding uncertainty. The VΓ-Var approach can therefore be applied as a substitute to uncertainty-aware backends for modeling the impacts of uncertainty propagation directly at the score level, particularly in scenarios where an uncertainty measure is unavailable, such as in DNNs. Finally, from Table 5.11 we can observe that duration-aware calibration models do not improve the performance of FPD-PLDA models, since the backend is already accounting for duration variability. Also, as expected, duration-aware models do not improve performance for longer utterances. Indeed, the miscalibration effects caused by duration variability tend to disappear for longer utterances. These results align with the findings reported for FPD-PLDA models applied to long segments in [99]. Also, in this case, we tested the effects of length-normalization. These results are shown in Table 5.12. We report the results of PLDA with length-normalized embeddings and VΓ-Var calibration, and PLDA with non-normalized embeddings in conjunction with VΓ-Var + Norm. Given the previous result, for the original utterances, we do not show results with duration-aware models. For these segments, we observe a strong decline in performance when length normalization is not applied, particularly the Log-Reg models exhibit significantly worse values compared to VΓ-Var. This difference is reduced when we introduce the embedding norm at the calibration level, with both C_{llr} and EER approaching the values obtained by length-normalized PLDA. Nonetheless, there is still a significant gap in terms of C_{prim}. The results obtained with short segments are shown in the second part of the table. Once again, length normalization proves to be essential for this dataset. The VΓ-Var + Norm model is able to reduce the difference between non-normalized and normalized embeddings. In this case, duration-aware model results are similar to those obtained for long segments. Interestingly, unlike what we observed in the SRE 2019 tests, there is an improvement when embedding norms and duration are combined. However, VΓ-Var + Dur applied to length normalized embeddings seems to outperform the VΓ-Var + Norm + Dur approach.

SRE2012 results

Table 5.13 Results on SRE 2012 with an e-vector frontend.

	PLDA		PSVM	
	C_{llr}	C_{prim}	C_{llr}	C_{prim}
Min. costs	*0.062*	*0.211*	*0.057*	*0.210*
Log-Reg	0.065	0.244	0.061	0.217
Linear VΓ	0.066	0.269	0.061	0.224
VΓ-Var (Gen)	0.066	0.241	0.062	0.213
VΓ-Var (Disc)	0.065	0.224	0.061	0.212
Log-Reg + QM_4	0.054	0.211	0.051	0.209
VΓ-Var + Dur (Gen)	0.056	0.230	0.054	0.204
VΓ-Var + Dur (Disc)	0.055	0.186	0.053	0.199

The final results related to the VΓ models that we show are the ones obtained with the SRE 2012 evaluation data. In this dataset, the speakers in the enrollment set have multiple replicas of the same utterances, some of which are recorded through different microphones. When utterances are not independent, as in this case, it becomes hard to define an effective duration for the enrollment side. However, we can neglect the uncertainty linked to the enrollment duration, if we assume that enrollment embeddings have been extracted from long utterances, as shown in [133]. The results for different calibration approaches are shown in Table 5.13. We set a large and fixed enrollment duration to train the duration-aware models. For this dataset, good calibration results can be achieved with Logistic Regression. In terms of C_{llr} we do not see improvements with the VΓ-Var models, although we notice lower primary costs, due to a better calibration in the low false alarm region. In the last three rows we report the results obtained with duration-aware models. Our VΓ-Var + Dur achieve slightly better C_{prim} while maintaing similar C_{llr} to the Log-Reg + QM_4.

5.4.2 C-Norm

In this section we analyze the performance of C-norm and its adaptive version, AC-norm (Section 4.3.1). This approach is tested on the SITW [128] and SRE 2019 [117] datasets. Our AC-norm combines a Log-Reg model with score normalization statistics employed as side information. A global calibration model based on prior-weighted logistic regression is used as a first baseline to compare our model. Score normalization techniques, S-norm and AS-norm, implemented as in [134], are also taken into consideration. For our adaptive version of C-norm, we use the same strategy used to select the cohort for the AS-norm.

Evaluation metrics and data

In this analysis we used two frontends. The first extractor is based on a Factorized Time Delay neural network [51], implemented as in [121]. The 512-dimensional embeddings are extracted after 20 epochs of training. No augmentations were applied to the data to train this model. Cross-entropy was employed as training loss. The ECAPA architecture [67] served as the second extractor. This model was trained for 10 epochs with augmented data, Additive Angular Margin softmax [43, 44] and cross-entropy loss. Finally, we extracted 192-dimensional embeddings. Also in this case we considered both PLDA and PSVM as our backends. The NIST SRE 2018 evaluation data and data from the training list (Section 5.1.1) have been used to train both backend classifiers. LDA was used to reduce the FTDNN embedding size to 200 for the PLDA and 400 for the PSVM. Preprocessing for both backends also included withening and length-normalization, WCCN was additionally applied for PSVM embeddings.

As evaluation metrics we used the primary metrics, C_{prim}^{SITW} and C_{prim}^{SRE19}, defined for the SITW and SRE 2019 tasks, and the Cost of Log-Likelihood Ratio C_{llr} [84, 82, 85]. The equal error rate (EER) and actual detection costs as defined for NIST SRE 2008 (DCF08) are also reported. As in the previous section, to show the normalized Detection Cost Function (DCF) corresponding to different target prior log-odds we also provide normalized Bayes error rate [82] plots.

SITW We removed some speakers of SITW that appear in the VoxCeleb1 dataset since it has been used to train our frontends. The development set was split into two non-overlapping parts. 70% of the speakers were used to train the calibration parameters for all models. The remaining part was used as normalization cohort. The results are obtained on the SITW evaluation set.

Table 5.14 Results for different embedding extractors and different classifiers on the SITW Evaluation dataset. Minimum costs are reported for the *unnormalized* scores. Log-Reg rows correspond to global calibration.

(a) FTDNN

	PLDA				PSVM			
	EER	DCF08	C_{prim}^{SITW}	C_{llr}	EER	DCF08	C_{prim}^{SITW}	C_{llr}
Min cost	*3.0%*	*0.159*	*0.312*	*0.110*	*3.0%*	*0.144*	*0.292*	*0.109*
Log-Reg	3.0%	0.161	0.316	0.113	3.0%	0.145	0.295	0.113
S-norm	3.5%	0.194	0.389	0.130	2.8%	0.146	0.301	0.107
AS-norm ($N = 100$)	3.0%	0.161	0.305	0.116	2.8%	0.141	0.290	0.110
C-norm	**2.5%**	**0.145**	**0.298**	**0.098**	**2.5%**	0.130	**0.259**	**0.095**
AC-norm ($N = 100$)	2.7%	0.149	0.299	0.104	**2.5%**	**0.126**	0.262	0.098

(b) ECAPA

	PLDA				PSVM			
	EER	DCF08	C_{prim}^{SITW}	C_{llr}	EER	DCF08	C_{prim}^{SITW}	C_{llr}
Min cost	*2.7%*	*0.132*	*0.269*	*0.105*	*2.1%*	*0.097*	*0.198*	*0.079*
Log-Reg	2.7%	0.135	0.272	0.111	2.1%	0.098	0.202	0.083
S-norm	2.6%	0.182	0.574	0.115	**1.7%**	0.087	0.200	0.072
AS-norm ($N = 100$)	2.4%	0.156	0.433	0.103	**1.7%**	0.090	0.197	0.073
C-norm	**2.3%**	0.131	0.303	**0.096**	**1.7%**	**0.081**	0.176	**0.069**
AC-norm ($N = 100$)	2.5%	**0.128**	**0.285**	0.102	**1.7%**	0.084	**0.173**	0.072

SRE 2019 The SRE 2019 Progress set has been used to train the calibration models. The unlabeled portion of SRE 2018 development data [116] was used as impostor cohort. And results are reported on the SRE 2019 Evaluation dataset.

SITW results

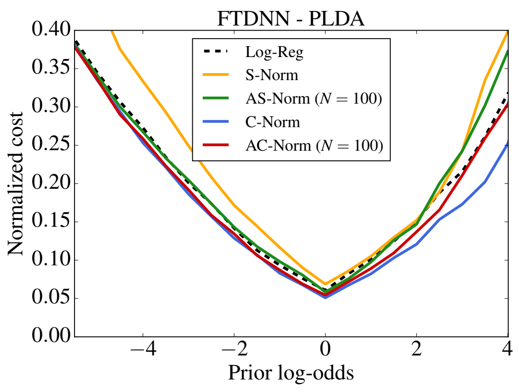
As for SRE datasets, the results of PSVM models on SITW are generally better than those obtained with PLDA as can be seen in Table 5.14. In general, the best results for C-norm were obtained using the complete cohort set, but we also report the performance of AC-norm. The cohort size for AS-norm was selected using the results on the development set, where $N = 100$ was found to be optimal. We used the same cohort value for AC-norm. To recalibrate the normalized scores obtained with S-norm and AS-norm, we trained a prior-weighted logistic regression on the development set. We used the same target prior, set to 0.1, to train each calibration model. The Bayes error plots for various combinations of frontend and backend are shown in Figure 5.9. The minimum costs computed on the original, unnormalized scores, are shown in Table 5.14, along with the Equal Error Rate (EER) and the actual costs associated with different score normalization and calibration methods.

S-norm appears to degrade PLDA-based systems performance (Fig. 5.9a and 5.9c, Tables 5.14a and 5.14b) with respect to other approaches. In some cases, unnormalized scores yield better results. C-norm, however, is able to considerably enhance the results compared to the unnormalized scores. Despite the small full cohort size, a cohort set of size $N = 100$ improves the results for AS-norm. This is particularly true for FTDNN embeddings, while a small degradation persists in terms of C_{prim}^{SITW} for the ECAPA. As expected, the cohort selection strategy proves ineffective in this scenario for C-norm, due to the already small full cohort size. In fact, the value of the C_{llr} obtained with the AC-norm models is slightly worse than the C-norm models, although, in terms of primary cost, we see an improvement for the ECAPA-based system.

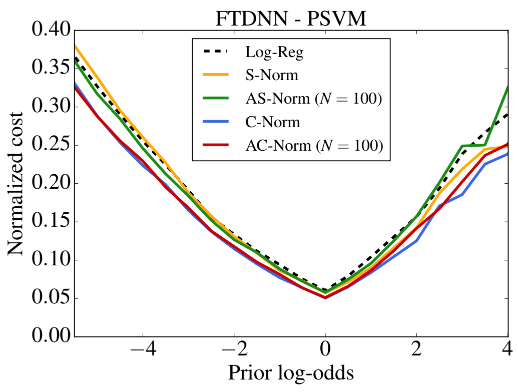
Turning to the PSVM models (Fig. 5.9b and 5.9d, right side of Tables 5.14) we can notice overall better results for all models, particularly when paired with the ECAPA frontend. In this case, S-norm produces much better results, approaching the ones obtained with AS-norm. Consistently with PLDA models, C-norm with the full cohort set delivers the best outcomes, especially in terms of C_{llr}. A significant improvement compared to both S-norm and raw scores in obtained with the FTDNN embeddings. A significant improvement is also shown for the actual primary cost for ECAPA-based models, although the improvement of the other costs is relatively small. Overall, both for PLDA and PSVM, C-norm proves to be effective and outperforms both global calibration and score-normalization approaches.

SRE 2019 results

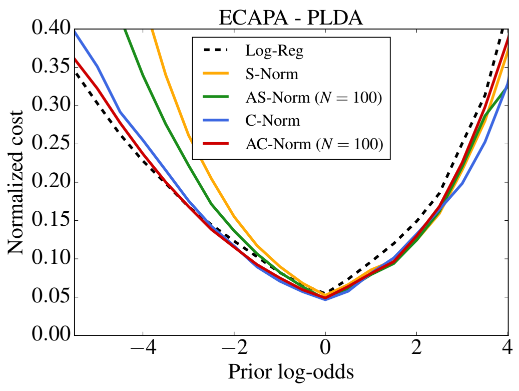
We also compare the same models with the SRE19 dataset, as shown in Figure 5.10 and Table 5.15. Once again, PSVM models outperform PLDA ones. In this case,



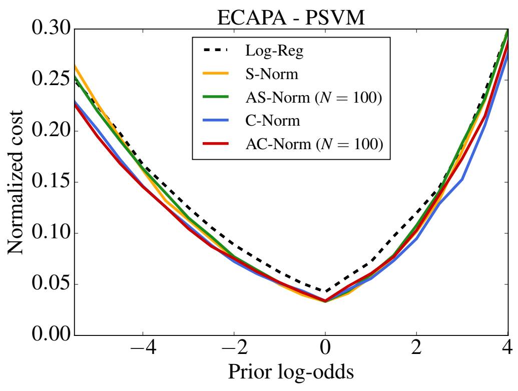
(a) FTDNN - PLDA



(b) FTDNN - PSVM



(c) ECAPA - PLDA



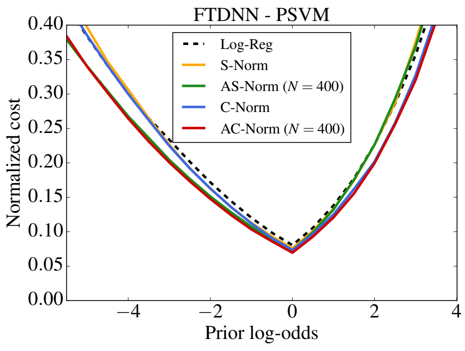
(d) ECAPA - PSVM

Figure 5.9 Results for different embedding extractors and different classifiers on the SITW Evaluation dataset. The AS-norm cohort size ($N = 100$) was selected according to the results on the calibration set. For C-norm the best results are obtained with the full cohort, both on the calibration and evaluation sets. As reference, the results with adaptive selection ($N = 100$) are given also for AC-norm.

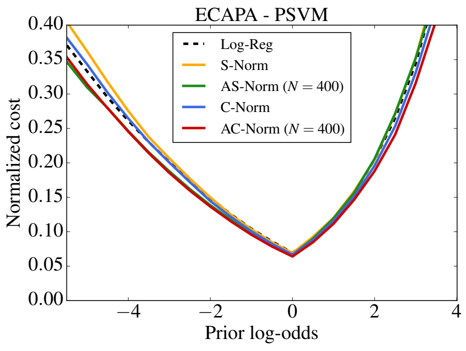
the adaptive models consistently outperform the non-adaptive versions. Contrary to SITW, the full cohort is significantly larger and potentially more heterogeneous, meaning that adaptive methods can leverage the benefits of cohort selection and improve results over non-adaptive methods. As for SITW, C-norm outperforms S-norm, although, in terms of C_{llr}, the improvement is marginal. A more detailed comparison in Figure 5.10 shows that C-norm achieves similar or slightly better performance in the low false alarm region while improving in the low false reject region. The adaptive models have a similar behavior. AC-norm provides a steady improvement in terms of C_{llr} compared to AS-norm, achieving better results in the

Table 5.15 Results for different embedding extractors and the PSVM classifier on the SRE 2019 Evaluation dataset. Minimum costs are reported for the *unnormalized* scores. Log-Reg rows correspond to global calibration.

	FTDNN				ECAPA			
	EER	DCF08	C_{prim}^{SRE19}	C_{llr}	EER	DCF08	C_{prim}^{SRE19}	C_{llr}
Min cost	*4.0%*	*0.188*	*0.374*	*0.153*	*3.5%*	*0.162*	*0.326*	*0.136*
Log-Reg	4.0%	0.188	0.380	0.158	3.5%	0.164	0.329	0.145
S-norm	3.8%	0.178	0.392	0.155	3.5%	0.166	0.356	0.147
AS-norm (400)	3.7%	0.166	0.337	0.148	3.4%	0.152	**0.307**	0.137
C-norm	3.7%	0.180	0.381	0.149	3.3%	0.161	0.337	0.142
AC-norm (400)	**3.5%**	**0.162**	**0.336**	**0.142**	**3.2%**	**0.150**	0.310	**0.134**



(a) FTDNN - PSVM



(b) ECAPA - PSVM

Figure 5.10 Results for different embedding extractors and the PSVM classifier on the SRE 2019 Evaluation dataset. The AS-norm and AC-norm cohort size ($N = 400$) was selected according to the results on the calibration set.

low false reject region while maintaining similar performance in the low false alarm region.

5.4.3 AD-Norm

In this section, we analyze the results produced by the AD-norm approach. We used the SRE datasets, specifically SRE 2016 and the SRE 2019 Evaluation data. We present results for two backends, PLDA and PSVM, trained using data from

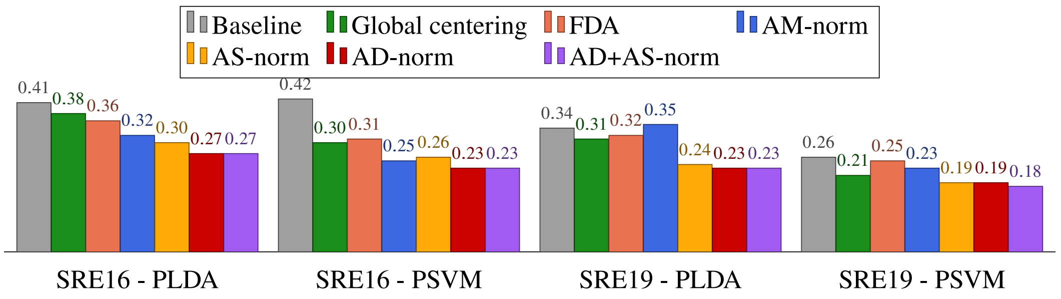


Figure 5.11 Minimum C^*_{llr} for different normalization approaches, out-of-domain training set (Switchboard + Mixer 04, 05, 06)

the frontend training data (5.1.1), for an out-of-domain scenario. Additionally, we conducted a second set of experiments by adding the SRE2018 evaluation set into the backend training data (partially in-domain for SRE19). The unlabeled portion (2472 segments) of SRE16 data was used as normalization data for the SRE16 evaluation. For the SRE19 test we selected 2000 segments from the SRE 2019 Progress test data. The ECAPA architecture is used as the embedding extractor. The dimension of the embeddings is reduced from 192 to 150 with LDA and L2-normalized. In both cases, the embeddings are re-centered with respect to the mean of the cohort set before applying AD-norm. The cohorts are selected with the same AS-norm strategy, as explained in 4.3.2. A further L2-normalization step is necessary after recentering. When testing with the PSVM backend, we also apply WCCN after L2-normalization. In Table 5.16 and 5.16 we report the EER and C^{min}_{prim} as defined by NIST for the two evaluations. The first five rows show the results obtained with (i) unnormalized scores (ii) re-centered with a mean computer over the whole impostor set and L2-normalized (iii) FDA approach [135] (iv) AM-norm [111] (v) AS-norm. The choice of the adaptive cohort set for the last three methods was driven by the best-performing baseline, AS-norm. Considering the average results, we chose a cohort size equal to 200 for the remaining adaptive models. A comparison of the C_{llr} of the different methods against our own can be found in Figure 5.11.

Table 5.16 shows that the adaptive methods outperform the global ones in all conditions on SRE16. The results that we obtained with AM-norm, are very close to the one reported in [111]. Overall, AM-norm and AS-norm achieve similar EER and C_{llr}, but in terms of $C_{\underset{prim}{min}}$ AS-norm performs better than AM-norm. A small degradation of this metric persists also for AD-norm. However, our method is able to outperform both AS-norm and AM-norm in terms of EER and C_{llr}. These findings

Table 5.16 Comparison of different normalization approaches on SRE 2016 datasets. In bold (i) results of the best *single* method for each training list and backend combination, and (ii) fusion results equal to or improving those of the best single method.

		PLDA			PSVM		
Training list	Normalization method	EER	C^{min}_{prim}	C^{*}_{llr}	EER	C^{min}_{prim}	C^{*}_{llr}
Switchboard + Mixer	None	11.3%	0.97	0.41	12.2%	0.99	0.42
	Global	11.0%	0.81	0.38	8.1%	0.74	0.30
	FDA [135]	10.5%	0.62	0.36	9.0%	0.58	0.31
	AM-norm [111]	9.1%	0.67	0.32	6.9%	0.62	0.25
	AS-norm [108]	8.7%	**0.52**	0.30	7.3%	**0.47**	0.26
	AD-norm	**7.6%**	**0.52**	**0.27**	**6.6%**	0.49	**0.23**
	AD+AS-norm	*7.9%*	***0.50***	***0.27***	***6.6%***	***0.46***	***0.23***
Switchboard + Mixer + SRE18	None	10.2%	0.87	0.37	8.5%	0.71	0.30
	Global	9.7%	0.75	0.35	7.2%	0.60	0.26
	FDA [135]	10.3%	0.62	0.36	9.1%	0.58	0.31
	AM-norm [111]	8.0%	0.63	0.29	6.5%	0.53	0.24
	AS-norm [108]	7.8%	**0.50**	0.27	7.0%	**0.47**	0.25
	AD-norm	**6.9%**	0.52	**0.25**	**6.4%**	0.50	**0.23**
	AD+AS-norm	*7.2%*	***0.49***	***0.25***	***6.3%***	***0.46***	***0.23***

highlight that although AS-norm performs slightly better in very low false acceptance regions, AD-norm is more effective over a wider range of applications.

Moving on to SRE19 (Table 5.17), AS-norm and AD-norm perform similarly, while AM-norm performs poorly with respect to these two methods. For the out-of-domain scenario, AD-norm provides a slight improvement, while AS-norm achieves lower values in presence of partially matching training and evaluation data. In perspective, for the in-domain scenario, PLDA trained on SRE18 data only achieve an EER, C^{min}_{prim} and C_{llr} of 4.4%, 0.38 and 0.17 respectively. It is worth noting, that, in the PSVM case for the mixed scenario, the unnormalized score provides the best results.

In both Tables, 5.16 and 5.17, the last row reports results obtained by taking the average of AD-norm and AS-norm scores, after standardizing non-target scores to unit variance. This is done because we think there may be complementary information in the AS-norm variance normalization effect, which is not possible in AD-norm. This assumption proves to be beneficial for all cases, the AD+AS-norm approach

Table 5.17 Comparison of different normalization approaches on SRE 2019 datasets. In bold (i) results of the best *single* method for each training list and backend combination, and (ii) fusion results equal to or improving those of the best single method.

		SRE 2019					
		PLDA			PSVM		
Training list	Normalization method	EER	C_{prim}^{min}	C_{llr}^{*}	EER	C_{prim}^{min}	C_{llr}^{*}
Switchboard + Mixer	None	9.7%	0.59	0.34	7.1%	0.59	0.26
	Global	9.0%	0.56	0.31	5.6%	0.49	0.21
	FDA [135]	9.0%	0.58	0.32	6.9%	0.50	0.25
	AM-norm [111]	10.3%	0.67	0.35	6.2%	0.50	0.23
	AS-norm [108]	6.5%	0.52	0.24	**4.9%**	0.43	**0.19**
	AD-norm	**6.2%**	**0.46**	**0.23**	5.1%	**0.41**	**0.19**
	AD+AS-norm	***6.1%***	***0.46***	***0.23***	***4.7%***	***0.39***	***0.18***
Switchboard + Mixer + SRE18	None	6.4%	0.44	0.23	3.6%	0.36	0.14
	Global	6.3%	0.45	0.23	3.8%	0.34	0.15
	FDA [135]	8.7%	0.57	0.31	6.8%	0.49	0.25
	AM-norm [111]	7.4%	0.51	0.27	4.4%	0.38	0.17
	AS-norm [108]	**4.7%**	0.40	**0.18**	**3.9%**	**0.35**	**0.15**
	AD-norm	5.0%	**0.38**	0.19	4.1%	0.36	0.16
	AD+AS-norm	***4.7%***	***0.37***	***0.18***	***3.8%***	***0.33***	***0.15***

provides similar or slightly better results with respect to the best results found before. In particular, there is a consistent improvement compared to AS-norm.

In Figure 5.12 we further analyze the effects of the cohort size, K, for the different normalization approaches. While SRE19 results are almost not affected by the choice of K, this is not true for SRE 2016. The absence of cross-validation data may complicate the selection of an optimal K value for a given dataset. For some combinations of backend and normalization models smaller cohort sizes may be more effective, although the relative performance of the approaches remains similar. A cohort size of K = 200 provides a good trade-off for the different datasets.

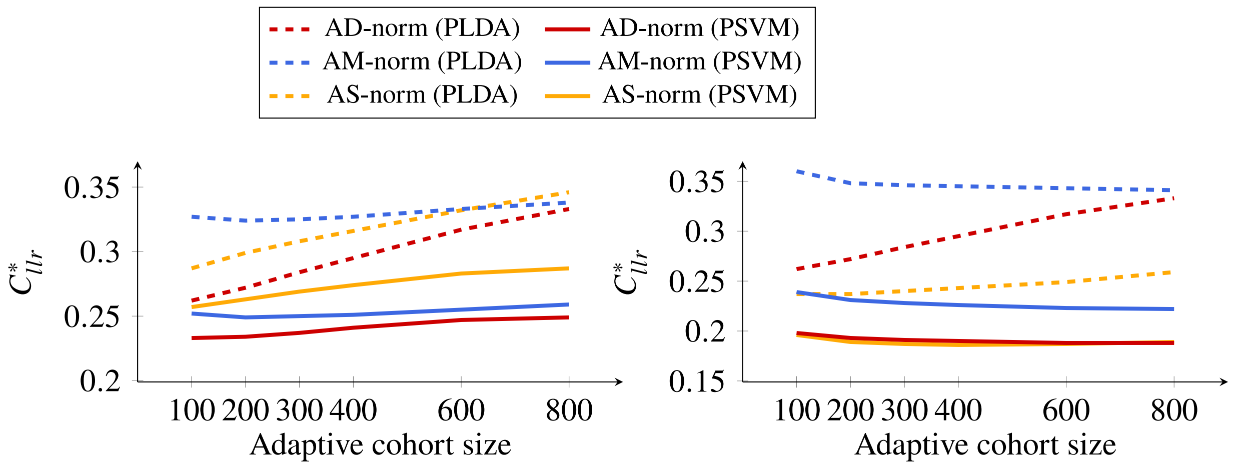


Figure 5.12 C^*_{llr} as a function of the cohort size. Left: SRE 2016. Right: SRE 2019.

Chapter 6

Conclusions

In this work, we outlined our contribution to the language recognition task and to the calibration problem. We have employed state-of-the-art models and techniques well-established in the speaker verification field to address language detection with scarce data scenarios. The models we developed proved successful, achieving highly competitive results in the NIST language recognition evaluation challenge of 2022.

Regarding our contribution to the calibration problem, we have introduced a generative framework that incorporates utterance-dependent mis-calibration sources. This method can directly handle utterance duration at the calibration level and as a generative model, it can be used in scenarios with missing labels. Moreover, it show comparable or superior performance to discriminative methods.

We have also proposed a new discriminative approach for trial-dependent calibration, C-norm. Score normalization statistics are used as side information to enrich the standard logistic regression model. Additionally, we have also introduced an alternative method, the AD-norm adaptive normalization thecnique. We employed the adaptive score normalization (AS-norm) cohort selection mechanism to effectively re-center enrollment and test speaker embeddings. Our results show that adaptive data normalization achieves similar or superior performance with lower scoring time compared to AS-norm.

Bibliography

[1] Lawrence R. Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. In *Prentice Hall signal processing series*, 1993.

[2] S. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken se. 1980.

[3] Christopher M. Bishop. Pattern recognition and machine learning (information science and statistics). 2006.

[4] Douglas A. Reynolds and Richard C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Trans. Speech Audio Process.*, 3:72–83, 1995.

[5] Douglas A. Reynolds. Comparison of background normalization methods for text-independent speaker verification. In *EUROSPEECH*, 1997.

[6] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digit. Signal Process.*, 10:19–41, 2000.

[7] Patrick Kenny. Joint factor analysis of speaker and session variability: Theory and algorithms. 2006.

[8] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel. Speaker and session variability in gmm-based speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15:1448–1460, 2007.

[9] Patrick Kenny, Gilles Boulianne, Pierre Ouellet, and Pierre Dumouchel. Joint factor analysis versus eigenchannels in speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15:1435–1447, 2007.

[10] Niko Brümmer, Albert Strasheim, Valiantsina Hubeika, Pavel Matejka, Luká Burget, and Ondrej Glembek. Discriminative acoustic language recognition via channel-compensated gmm statistics. In *Interspeech*, 2009.

[11] Najim Dehak. Discriminative and generative approaches for long- and short-term speaker characteristics modeling: application to speaker verification. 2009.

[12] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19:788–798, 2011.

[13] Yun Lei, Nicolas Scheffer, Luciana Ferrer, and Mitchell 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)*, pages 1695–1699, 2014.

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

[15] Paul Covington, Jay K. Adams, and Emre Sargin. Deep neural networks for youtube recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015.

[17] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *ArXiv*, abs/1604.06737, 2016.

[18] Yochai Konig, Larry Heck, Mitch Weintraub, and Kemal S. Sonmez. Nonlinear discriminant feature extraction for robust text-independent speaker recognition. 1997.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Larry Heck, Yochai Konig, M. Kemal Sönmez, and Mitch Weintraub. Robustness to telephone handset distortion in speaker recognition by discriminative feature design. *Speech Commun.*, 31:181–192, 2000.

[21] Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview. *Neural networks : the official journal of the International Neural Network Society*, 140:65–99, 2020.

[22] Alicia Lozano-Diez, Oldrich Plchot, Pavel Matejka, and Joaquín González-Rodríguez. Dnn based embeddings for language recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5184–5188, 2018.

[23] Salvatore Sarni, Sandro Cumani, Sabato Marco Siniscalchi, and Andrea Bottino. Description and analysis of the kpt system for nist language recognition evaluation 2022. *INTERSPEECH 2023*, 2023.

[24] Warren S. McCulloch and Walter H. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990.

[25] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[26] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[27] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

[29] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016.

[30] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[31] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Cresceptron: a self-organizing neural network which grows adaptively. *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, 1:576–581 vol.1, 1992.

[32] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986.

[33] Sebastian Ruder. An overview of gradient descent optimization algorithms. *ArXiv*, abs/1609.04747, 2016.

[34] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[35] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of dnns with natural gradient and parameter averaging. *arXiv: Neural and Evolutionary Computing*, 2014.

[36] Boris Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.

[37] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

[38] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[39] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[40] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[41] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6746, 2017.

[42] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25:926–930, 2018.

[43] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694, 2019.

[44] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu. Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition. *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1652–1656, 2019.

[45] Fadi Boutros, Naser Damer, Florian Kirchbuchner, and Arjan Kuijper. Elasticface: Elastic margin loss for deep face recognition. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1577–1586, 2021.

[46] Jichao Jiao, Weilun Liu, Yaokai Mo, Jian Jiao, Zhong liang Deng, and Xinping Chen. Dyn-arcface: dynamic additive angular margin loss for deep face recognition. *Multimedia Tools and Applications*, 80:25741 – 25756, 2021.

[47] Jiankang Deng, J. Guo, Tongliang Liu, Mingming Gong, and Stefanos Zafeiriou. Sub-center arcface: Boosting face recognition by large-scale noisy web faces. In *European Conference on Computer Vision*, 2020.

[48] Miao Zhao, Yufeng Ma, Yiwei Ding, Yu Zheng, Min Liu, and Minqiang Xu. Multi-query multi-head attention pooling and inter-topk penalty for speaker verification. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6737–6741, 2021.

[49] Alexander H. Waibel, Toshiyuki Hanazawa, Geoffrey E. Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust. Speech Signal Process.*, 37:328–339, 1989.

[50] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur. Speaker recognition for multi-speaker conversations using x-vectors. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5796–5800, 2019.

[51] Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Arab Yarmohammadi, and Sanjeev Khudanpur. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, 2018.

[52] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6655–6659, 2013.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[54] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matejka, and Oldrich Plchot. But system description to voxceleb speaker recognition challenge 2019. *ArXiv*, abs/1910.12592, 2019.

[55] Daniel Garcia-Romero, Gregory Sell, and Alan McCree. Magneto: X-vector magnitude estimation network plus offset for improved speaker recognition. In *The Speaker and Language Recognition Workshop*, 2020.

[56] João Monteiro, Md. Jahangir Alam, and Tiago H. Falk. Residual convolutional neural network with attentive feature pooling for end-to-end language identification from short-duration speech. *Comput. Speech Lang.*, 58:364–376, 2019.

[57] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xinyu Zhang, Ming-Hsuan Yang, and Philip H. S. Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:652–662, 2019.

[58] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[59] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda. Attentive statistics pooling for deep speaker embedding. In *Interspeech*, 2018.

[60] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2017.

[61] Jenthe Thienpondt, Brecht Desplanques, and Kris Demuynck. Integrating frequency translational invariance in tdnns and frequency positional information in 2d resnets to enhance speaker verification. In *Interspeech*, 2021.

[62] Saurabh Kataria, Phani Sankar Nidadavolu, Jesús Villalba, Nanxin Chen, Paola García, and Najim Dehak. Feature enhancement with deep feature losses for speaker verification. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7584–7588, 2019.

[63] Li Zhang, Qing Wang, and Lei Xie. Duality temporal-channel-frequency attention enhanced speaker representation learning. *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 206–213, 2021.

[64] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. *ArXiv*, abs/2005.08100, 2020.

[65] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.

[66] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[67] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. In *Interspeech*, 2020.

[68] Zhuang Liu, Hanzi Mao, Chaozheng Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11966–11976, 2022.

[69] Bei Liu, Zhengyang Chen, and Yanmin Qian. Depth-first neural architecture with attentive feature fusion for efficient speaker verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1825–1838, 2023.

[70] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[71] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks.

2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4510–4520, 2018.

[72] David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. Spoken language recognition using x-vectors. In *The Speaker and Language Recognition Workshop*, 2018.

[73] Woohyun Kang, Jahangir Alam, and Abderrahim Fathan. Deep learning-based end-to-end spoken language identification system for domain-mismatched scenario. In *International Conference on Language Resources and Evaluation*, 2022.

[74] Nist 2022 language recognition evaluation plan. 2022.

[75] Sergey Ioffe. Probabilistic linear discriminant analysis. In *European Conference on Computer Vision*, 2006.

[76] Patrick Kenny. Bayesian speaker verification with heavy-tailed priors. In *The Speaker and Language Recognition Workshop*, 2010.

[77] Daniel Garcia-Romero and Carol Y. Espy-Wilson. Analysis of i-vector length normalization in speaker recognition systems. In *Interspeech*, 2011.

[78] Niko Brümmer and Edward de Villiers. The speaker partitioning problem. In *The Speaker and Language Recognition Workshop*, 2010.

[79] Sandro Cumani, Niko Brümmer, Luká Burget, Pietro Laface, Oldrich Plchot, and Vasileios Vasilakakis. Pairwise discriminative speaker verification in the i-vector space. *IEEE Transactions on Audio, Speech, and Language Processing*, 21:1217–1227, 2011.

[80] Sandro Cumani and Pietro Laface. Large-scale training of pairwise support vector machines for speaker recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22:1590–1600, 2014.

[81] Kevin P. Murphy. Machine learning - a probabilistic perspective. In *Adaptive computation and machine learning series*, 2012.

[82] Niko Brümmer. *Measuring, refining and calibrating speaker and language information extracted from speech.* PhD thesis, 2010.

[83] George R. Doddington, Mark A. Przybocki, Alvin F. Martin, and Douglas A. Reynolds. The nist speaker recognition evaluation - overview, methodology, systems, results, perspective. *Speech Commun.*, 31:225–254, 2000.

[84] Niko Brümmer and Johan A. du Preez. Application-independent evaluation of speaker detection. In *Computer Speech and Language*, 2006.

[85] David A. van Leeuwen and Niko Brümmer. An introduction to application-independent evaluation of speaker recognition systems. In *Speaker Classification*, 2007.

[86] Niko Brümmer, Luká Burget, Jan Honza ernocký, Ondrej Glembek, Frantisek Grézl, Martin Karafiát, David A. van Leeuwen, Pavel Matejka, Petr Schwarz, and Albert Strasheim. Fusion of heterogeneous speaker recognition systems in the stbu submission for the nist speaker recognition evaluation 2006. *IEEE Transactions on Audio, Speech, and Language Processing*, 15:2072–2084, 2007.

[87] Niko Brümmer and George R. Doddington. Likelihood-ratio calibration using prior-weighted proper scoring rules. In *Interspeech*, 2013.

[88] David A. van Leeuwen and Niko Brümmer. The distribution of calibrated likelihood-ratios in speaker recognition. In *Interspeech*, 2013.

[89] Sandro Cumani. On the distribution of speaker verification scores: Generative models for unsupervised calibration. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:547–562, 2021.

[90] Miranti Indar Mandasari, Rahim Saeidi, Mitchell McLaren, and David A. van Leeuwen. Quality measure functions for calibration of speaker recognition systems in various duration conditions. *IEEE Transactions on Audio, Speech, and Language Processing*, 21:2425–2438, 2013.

[91] Miranti Indar Mandasari, Rahim Saeidi, and David A. van Leeuwen. Quality measures based calibration with duration and noise dependency for speaker recognition. *Speech Commun.*, 72:126–137, 2015.

[92] Andreas Nautsch, Rahim Saeidi, Christian Rathgeb, and Christoph Busch. Robustness of quality-based score calibration of speaker recognition systems with respect to low-snr and short-duration conditions. In *The Speaker and Language Recognition Workshop*, 2016.

[93] Luciana Ferrer, Mitchell McLaren, and Niko Brümmer. A speaker verification backend with robust performance across conditions. *Comput. Speech Lang.*, 71:101258, 2021.

[94] Sandro Cumani and Salvatore Sarni. A generative model for duration-dependent score calibration. In *Interspeech*, 2021.

[95] Sandro Cumani and Salvatore Sarni. The distributions of uncalibrated speaker verification scores: A generative model for domain mismatch and trial-dependent calibration. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2204–2219, 2023.

[96] Zenna Tavares, Xin Zhang, Edgar Minaysan, Javier Burroni, Rajesh Ranganath, and Armando Solar-Lezama. The random conditional distribution for higher-order probabilistic inference. *ArXiv*, abs/1903.10556, 2019.

[97] Sandro Cumani, Ondrej Glembek, Niko Brümmer, Edward de Villiers, and Pietro Laface. Gender independent discriminative speaker recognition in i-vector space. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4361–4364, 2012.

[98] Luká Burget, Oldrich Plchot, Sandro Cumani, Ondrej Glembek, Pavel Matejka, and Niko Brümmer. Discriminatively trained probabilistic linear discriminant analysis for speaker verification. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4832–4835, 2011.

[99] Sandro Cumani, Oldrich Plchot, and Pietro Laface. On the use of i–vector posterior distributions in probabilistic linear discriminant analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22:846–857, 2014.

[100] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Md. Jahangir Alam, and Pierre Dumouchel. Plda for speaker verification with utterances of arbitrary duration. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7649–7653, 2013.

[101] Ondrej Glembek, Luká Burget, Pavel Matejka, Martin Karafiát, and Patrick Kenny. Simplification and optimization of i-vector extraction. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4516–4519, 2011.

[102] Sandro Cumani. Fast scoring of full posterior plda models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23:2036–2045, 2015.

[103] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018.

[104] Niko Brümmer, Anna Silnova, Luká Burget, and Themos Stafylakis. Gaussian meta-embeddings for efficient scoring of a heavy-tailed plda model. *ArXiv*, abs/1802.09777, 2018.

[105] Roland Auckenthaler, Michael J. Carey, and Harvey Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digit. Signal Process.*, 10:42–54, 2000.

[106] Douglas E. Sturim and Douglas A. Reynolds. Speaker adaptive cohort selection for tnorm in text-independent speaker verification. *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, 1:I/741–I/744 Vol. 1, 2005.

[107] Zahi N. Karam, William M. Campbell, and Najim Dehak. Towards reduced false-alarms using cohorts. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4512–4515, 2011.

[108] Sandro Cumani, Pier Domenico Batzu, Daniele Colibro, Claudio Vair, Pietro Laface, and Vasileios Vasilakakis. Comparison of speaker recognition approaches for real applications. In *Interspeech*, 2011.

[109] Sandro Cumani and Salvatore Sarni. Impostor score statistics as quality measures for the calibration of speaker verification systems. In *The Speaker and Language Recognition Workshop*, 2022.

[110] Sandro Cumani and Salvatore Sarni. From adaptive score normalization to adaptive data normalization for speaker verification systems. *INTERSPEECH 2023*, 2023.

[111] Mitchell McLaren, Md. Hafizur Rahman, Diego Castán, Mahesh Kumar Nandwana, and Aaron D. Lawson. Adaptive mean normalization for unsupervised adaptation of speaker embeddings. In *The Speaker and Language Recognition Workshop*, 2020.

[112] Sandro Cumani and Pietro Laface. Exact memory-constrained upgma for large scale speaker clustering. *Pattern Recognit.*, 95:235–246, 2019.

[113] Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. Voxceleb: Large-scale speaker verification in the wild. *Computer Science and Language*, 2019.

[114] David Snyder, Guoguo Chen, and Daniel Povey. Musan: A music, speech, and noise corpus. *ArXiv*, abs/1510.08484, 2015.

[115] Kong Aik LEE, Tomi H. Kinnunen, Daniele Colibro, Claudio Vair, Andreas Nautsch, Hanwu Sun, Liang He, Tianyu Liang, Qiongqiong Wang, Mickael Rouvier, Pierre-Michel Bousquet, Rohan Kumar Das, Ignacio Viñals Bailo, Meng Liu, H'ector Deldago, Xuechen Liu, Md. Sahidullah, Sandro Cumani, Boning Zhang, Koji Okabe, Hitoshi Yamamoto, Ruijie Tao, Haizhou Li, Alfonso Ortega Gim'enez, Longbiao Wang, and Luis Buera. I4u system description for nist sre'20 cts challenge. *ArXiv*, abs/2211.01091, 2022.

[116] The nist 2018 speaker recognition evaluation. 2018.

[117] The nist 2019 speaker recognition evaluation: Cts challenge. 2019.

[118] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[119] Leslie N. Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2015.

[120] David Snyder, Jesús Villalba, Nanxin Chen, Daniel Povey, Gregory Sell, Najim Dehak, and Sanjeev Khudanpur. The jhu speaker recognition system for the voices 2019 challenge. In *Interspeech*, 2019.

[121] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak. State-of-the-art speaker recognition with neural network embeddings in nist sre18 and speakers in the wild evaluations. *Comput. Speech Lang.*, 60, 2020.

[122] Svetlana Tchistiakova. Time delay neural network. Available at =https://kaleidoescape.github.io/tdnn/, Nov 2019.

[123] Mirco Ravanelli, Titouan Parcollet, Peter William VanHarn Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, Ju-Chieh Chou, Sung-Lin Yeh, Szu-Wei Fu, Chien-Feng Liao, E. N. Rastorgueva, Franccois Grondin, William Aris, Hwidong Na, Yan Gao, Renato De Mori, and Yoshua Bengio. Speechbrain: A general-purpose speech toolkit. *ArXiv*, abs/2106.04624, 2021.

[124] NIST 2017 language recognition evaluation plan, 2017. Available at https://www.nist.gov/system/files/documents/2017/09/29/lre17_eval_plan-2017-09-29_v1.pdf.

[125] Najim Dehak, Pedro A. Torres-Carrasquillo, Douglas A. Reynolds, and Réda Dehak. Language recognition via i-vectors and dimensionality reduction. In *Interspeech*, 2011.

[126] NIST 2022 language recognition evaluation plan, 2022. Available at https://lre.nist.gov/uassets/3.

[127] Yooyoung Lee, Craig S. Greenberg, Eliot Godard, Asad Anwar Butt, Elliot Singer, Trang Nguyen, Lisa P. Mason, and Douglas A. Reynolds. The 2022 nist language recognition evaluation. *ArXiv*, abs/2302.14624, 2023.

[128] Mitchell McLaren, Luciana Ferrer, Diego Castán, and Aaron D. Lawson. The speakers in the wild (sitw) speaker recognition database. In *Interspeech*, 2016.

[129] Patrick Kenny, Themos Stafylakis, Pierre Ouellet, Vishwa Gupta, and Md. Jahangir Alam. Deep neural networks for extracting baum-welch statistics for speaker recognition. *The Speaker and Language Recognition Workshop (Odyssey 2014)*, 2014.

[130] Sandro Cumani, Pietro Laface, and Farzana Kulsoom. Speaker recognition by means of acoustic and phonetically informed gmms. In *Interspeech*, 2015.

[131] Sandro Cumani and Pietro Laface. Speaker recognition using e–vectors. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26:736–748, 2018.

[132] Bianca Zadrozny and Charles Peter Elkan. Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[133] Sandro Cumani, Oldrich Plchot, and Pietro Laface. Probabilistic linear discriminant analysis of i-vector posterior distributions. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7644–7648, 2013.

[134] Pavel Matejka, Ondrej Novotný, Oldrich Plchot, Luká Burget, Mireia Díez Sánchez, and Jan Honza ernocký. Analysis of score normalization in multilingual speaker recognition. In *Interspeech*, 2017.

[135] Pierre-Michel Bousquet and Mickael Rouvier. On robustness of unsupervised domain adaptation for speaker recognition. In *Interspeech*, 2019.