UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Università degli Studi di Palermo

Analisi Intelligente dei Segnali

Speech Features per i tasks di Speaker Identification e Verification

Docente:
Sabato Marco Siniscalchi

Autore:
Gabriele Nicolò Costa

Corso di Laurea
Ingegneria Informatica LM-32

Anno Accademico 2024/25

Contents

Chapter 1

Introduzione

Nel contesto dell'analisi intelligente dei segnali un task molto importante e di notevole interesse scientifico-industriale è il riconoscimento del parlante o **speaker verification** o in genere dell'identificazione di chi parla o è autorizzato a parlare rispetto a chi non lo sia, **speaker identification**. Il seguente progetto affronta la tematica della **speaker identification** e della **speaker verification**, analizzando le principali tecniche che permettono di ottenere una rappresentazione alternativa a partire dalle tracce audio al fine di realizzare il riconoscimento degli speakers. Verranno quindi appprofondite le tecniche utilizzate dalla letteratura scientifica e le **features** che entrano in gioco, analizzando come e quali sistemi permettono l'analisi di queste caratteristiche. Infine verrà condotta una fase sperimentale dove verranno confrontate diverse tecniche e metodologie proposte in ambito di ricerca.
Il lavoro è strutturato nel seguente modo:

- Il capitolo 2 prenderà in considerazione il problema sotto un punto di vista formale, analizzando cosa viene fatto in letteratura scientifica
- Il capitolo 3 analizzerà invece quali sono le features largamente utilizzate per poter effettuare *speaker identification e verification*
- Il capitolo 4 fornirà una overview dei sistemi utilizzati per analizzare le features descritte prima, entrando anche nel dettaglio dell'architettura proposta ed utilizzata nella sezione sperimentale per realizzare speaker identification e verification in un unica pipeline integrata
- Il capitolo 5 infine valuterà gli esperimenti condotti, descrivendo dataset e risultati ottenuti, oltre che fornire le conclusioni finali del lavoro

Chapter 2

Speaker Identification e Verification

2.1 Descrizione del problema

Nel contesto dell'analisi intelligente dei segnali sappiamo che lavorando con sorgenti audio spesso ricorriamo ad una rappresentazione alternativa per ogni specifico segnale audio, chiamata appunto come rappresentazione di **features**, che possiamo usare per task diversi.
Uno dei task che da sempre è studio di lavori (sia per la sua applicabilità pratica nel settore industriale, che da un punto di vista di ricerca) è quello dello Speaker Identification. Infatti questo task trova grande interesse nei sistemi di riconoscimento biometrici, diventando quindi un punto cruciale nel settore della sicurezza informatica. In generale, i sistemi biometrici si basano sulle caratteristiche *individuali e altamente discriminative* delle persone, presentando già il principale compito dei sistemi di riconoscimento: identificare correttamente le persone e riconoscere se ci sono degli utenti non autorizzati (o impostori), sulla base delle caratteristiche (o *features*) di ogni singolo speaker.
Nel task della *speaker identification*, il nostro obiettivo principale è cercare di estrarre dagli human speech di ogni singolo speaker una rappresentazione alternativa utile a rappresentare univocamente un soggetto, quindi costruire dei **Speaker Models**, che permettano poi di riconoscere i singoli utenti tramite una classificaione o pattern-matching in generale. Lo scopo dello *speaker models* è quello alla fine di creare un database di persone (quindi di profili) che conosco, al fine di avere abbastanza informazioni per poter riconoscere quel determinato speaker in seguito. Uno schema di funzionamento è riportato in 2.1 Il sistema viene diviso in due fasi:

- una fase di training, o anche definita come *enrolment*, dove acquisiamo audio da cui estrarre features da cui estrarre un database di speaker models
- Una fase di *testing*, dove andiamo a valutare quanti speaker riusciamo correttamente a riconoscere

Quando però realizziamo la fase di testing dobbiamo distinguere tra una identificazione definita *closed-set*, ovvero abbiamo un insieme di identità che conosciamo (quindi abbiamo profilato) e dobbiamo distinguere gli uni dagli altri, ed una identificazione definita *open-set*,

dove invece dobbiamo identificare tra identità che conosciamo e quelle che invece non conosciamo e classifichiamo come sconosciute, o appunto impostori.
In questo contesto si inserisce la *speaker verification*. Partiamo da una base di identification (dove quindi abbiamo un database di identità conosciute), e cerchiamo di capire se un un nuovo speaker fa parte del nostro insieme o no. In questo caso il focus non è tanto l'identificazione, ma tanto il rejection dell'utente nel caso in cui non faccia realmente parte del mio insieme di identità conosciute. Facendo un esempio banale, se un utente parla ad un sistema di riconoscimento e dice di essere un determinato utente (claimed ID), il sistema deve essere in grado di identificarlo correttamente (anche qui, closed-set o open-set) e di assegnargli una identità, eventualmente "bloccandolo" se troviamo che non è che dice di essere o non fa parte dell'insieme. Parliamo quindi in questo caso di rejection e acceptance di un determinato speaker, per valutare correttamente il sistema, sulla base per esempio di uno score (compito di solito svolto dal Backend del sistema di verifica).

Inoltre, entrambi i task possono essere dipendenti o indipendenti dai vincoli lessicali [6]. Infatti distinguiamo tra *Text-Indipendent* e *Text-Dipendent*, sia in ottica Speaker Identification che Verification. Per vincolo lessicale intendiamo infatti che lo speaker deve dire una determinata sequenza di parole, che in pratica corrisponde a pronunciare una *keyword o passphrase*.

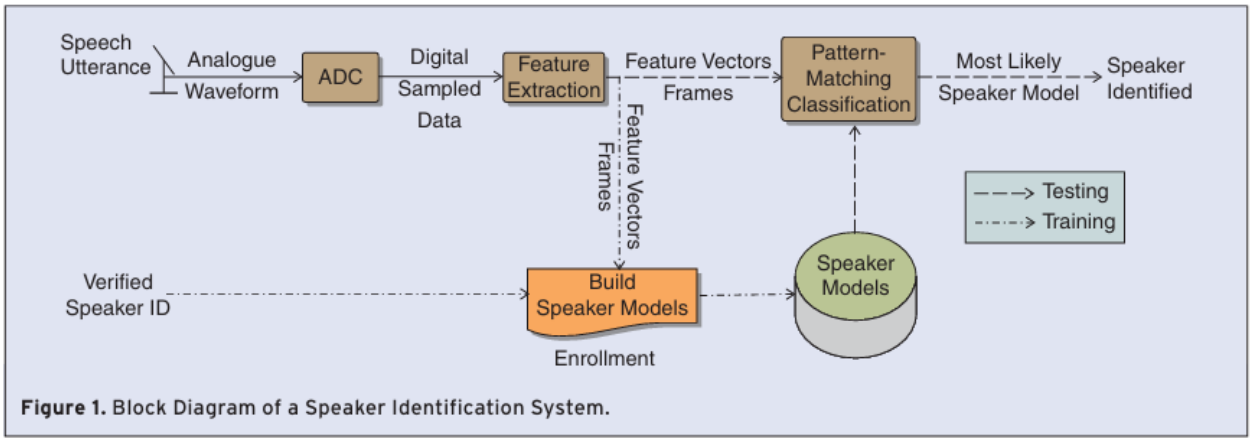

Figure 2.1: Sistema di Speaker identification, [5]

2.2 Valutazione dei sistemi: metriche e metodologie

Per poter correttamente valutare un sistema di identificazione e verifica delle identità dobbiamo basarci su specifiche metriche e metodologie. Per quanto riguarda le metodologie, sia in open-set che closed-set, partiamo da un training set composto da segnali audio a cui applichiamo la classica analisi *Short-Time*, al fine di poter effettuare operazioni di estrazione di features. Training e Testing sono composti da audio di speaker noti nel caso del task di Speaker Identification, da speaker diversi nel caso del task di Speaker Verification.

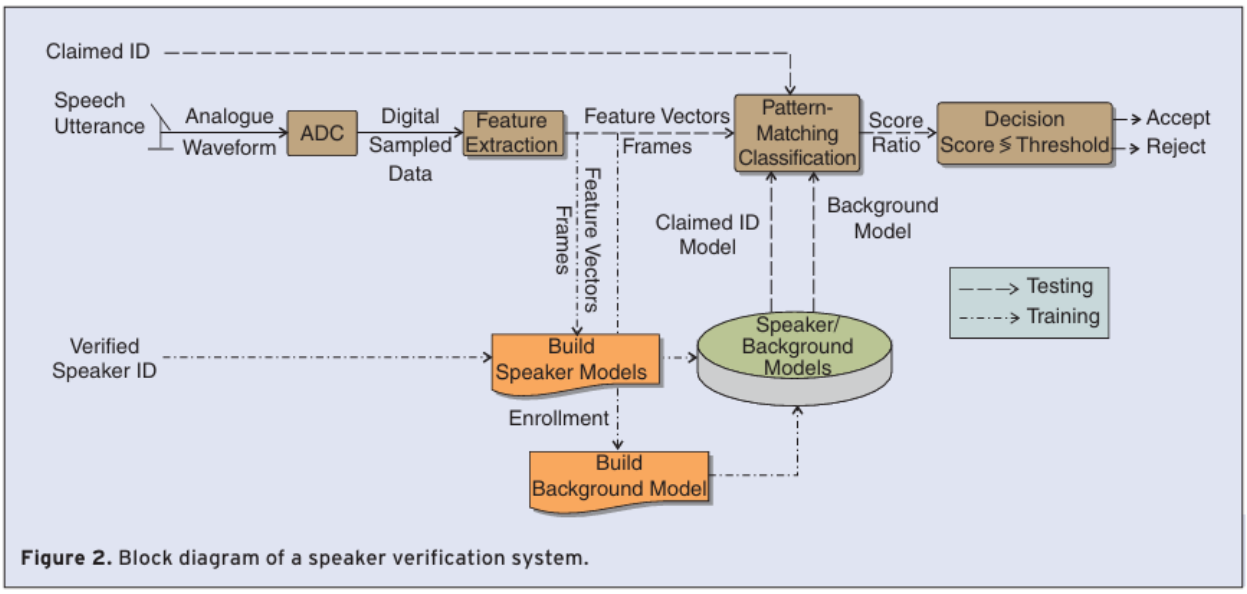

Figure 2.2: Sistema di Speaker Verification, [5]

Per quanto riguarda le metriche da utilizzare, queste sono diverse a seconda del task. Nel task della identificazione è importante capire quanti utenti riusciamo a riconoscere e distinguere correttamente, quindi valutare metriche come la *precisione, la accuracy, la f1score* per ogni speaker e in maniera complessiva del sistema.

Per quanto riguarda invece la verifica delle identità è più importante riconoscere quante persone riusciamo correttamente a riconoscere come intrusi e quindi bloccare e non far accedere al sistema. Quindi questo task è da vedere più come classificazione binaria che multiclasse, perché a noi interessano *True Positive Rate (quante persone riusciamo correttamente a far passare)* e *False Positive Rate (quante persone facciamo passare sebbene siano impostori)*. Una metrica aggregata è la **Equal Error Rate (EER)**, metrica comune nei sistemi biometrici. Si riferisce al punto in cui il tasso di falsi accettamenti (**(FAR - False Acceptance Rate**) è uguale al tasso di falsi rifiuti (**FRR - False Rejection Rate**). In termini di TPR e TNR, la FAR è il complementare del TNR (percentuale di impostori correttamente identificati), quindi coincide con la FPR, mentre la TPR è il complementare della FRR (percentuale di parlanti autentici e correttamente accettati). Un riassunto è riportato in Tab.2.1

2.3 Scenari applicativi

Come detto prima, questi task sono alla base sopratutto dei sistemi di riconoscimento biometrici, con particolare focus sia al riconoscimento che alla verifica dell'identità. Ques'ultimo aspetto diventa chiave e cruciale per i sistemi di sicurezza basati su riconoscimento della voce (*biometrica dinamica*), per il controllo degli accessi o ulteriore fattore di autenticazione.
Lo scopo di questo progetto è quello di esplorare entrambi gli aspetti, con particolare focus

Task	**Obiettivo**	**Metriche utilizzate**
Speaker Identification (Closed-set)	Riconoscere l'identità dello speaker tra un insieme noto di speakers (classificazione multiclasse).	• Accuracy • Precision, Recall, F1-score (per classe e macro media)
Speaker Verification (Open-set)	Verificare se uno speaker è chi dichiara di essere (classificazione binaria: accetta/rifiuta).	• True Positive Rate (TPR) • False Positive Rate (FPR) • False Acceptance Rate (FAR) • False Rejection Rate (FRR) • Equal Error Rate (EER)

Table 2.1: Task, obiettivi e metriche nei sistemi di speaker recognition

alle tecniche principali che permettono di identificare gli speaker con metodologie basate sul Deep Learning, esplorando anche come un sistema di identificazione può essere traslato ed adattato in un sistema di verifica delle identità tramite classificazione binaria *one-vs-others*, dove i nostri utenti VERI sono quelli del nostro dataset *closed-set*, quelli invece FALSI (o impostori) sono quelli che non fanno parte del dataset.

Chapter 3

Speech Features & Embeddings

In molte applicazioni di speech processing è spesso conveniente ed utile avere una rappresentazione parametrica o alternativa dell'informazione trasportata dal segnale. L'informazione trasportata dal segnale infatti include importanti fattori come il pitch period, il modello dell'impulso glottale, l'eccitazione, il modello del tratto vocale: tutte informazioni utili a poter distinguere i diversi speaker in task come lo speaker identification e lo speaker verification. L'obiettivo della Digital Speech Processing (DSP) è proprio quello di trovare metodi per poter elaborare i segnali speech audio e poter quindi estrarre queste particolari informazioni in forma parametrica, avendo difatti una **rappresentazione alternativa** del segnale.

Questo permette difatti di andare ad estrarre **speech features**, dunque parametri o caratteristiche, che possiamo dare in input a modelli di machine learning, per permette alle reti di poter eseguire particolari combiti. Spesso però il machine learning ha una duplice valenza: è spesso usato sia come "modello" di elaborazione delle features, sia come modello per estrarre rappresentazioni più raffinate delle features in input. Queste rappresentazioni più raffinate prendono il nome di **speech embeddings**, e permettono di aggiungere un ulteriore informazione: permette difatti di aggregare i contributi di ogni singola features in ingresso, cercando di ottenere rappresentazioni non lineari più complesse e talvolte più discriminanti.

3.1 Speech Features

In ogni caso, le proprietà dello speech cambiano lentamente nel tempo e nella frequenza. Per poter fronteggiare questo fenomeno si ricorre spesso a metodi di analisi definiti come *short time*, dove andiamo ad eseguire una analisi frame per frame. Questo deriva dal fatto che effettuando una analisi short time a livello di frame, andiamo a processare singoli segmenti cui possiamo assumere abbiano proprietà fisse e permettono di estrarre le informazioni relative per lo speech processing. Le metodologie di short time possono essere o basate su tempo o su frequenza. L'analisi time-domain è utile per andare ad isolare segmenti dove è presente un pezzo di segnale *voiced* oppure *unvoiced*. Permette infatti di isoalre meglio segmenti dal rumore di fondo, utili per massimizzare ulteriormente l'estrazione di features solo su segmenti definiti come voiced, ad esempio integrando un **Voice Activity Detector (VAD)**,

che tronca andando a valutare l'energia del segmento, ad esempio con analisi **Short Time Energy**.

Tuttavia, nei task di **Speaker Verification** e **Speaker Identification**, conviene trattare di metodi short time basati sulla frequenza, dal momento che molte informazioni riguardanti formanti e armoniche (utili per poter modellare un filtro che rappresenti il tratto vocale di una persona, quindi un modello che riesce a parametrizzare), oltre che informazioni definibili come "percettive". Alcune features basate su metodi frequency-domain soon gli MFCCs (Mel Frequency Cepstral Coefficients). Talvolta abbianiamo a queste features anche altri valori di derivata prima o seconda, per poter esprimere anche la variabilità dell'analisi short time, come i metodi Delta e Delta Delta.

3.1.1 Cepstrum

Il cepstrum è definito come la IDFT della log-magnitudo della DFT del segnale, ovvero:

$$c[n] = F^{-1}[log(|F[x[n]]|)]$$

Dove i due operatori sono la trasformata discreta di fourier DFT e la inverse DFT, la IDFT. A partire del cepstrum possiamo definire una rappresentazione alternativa, ma non ci basta ancora: vorremmo smussare le attenuazioni, estraendo anche componenti come il pitch e sopratutto modellare l'inviluppo della trasformata stessa. Tuttavia il cepstrum ci permette di poter separare la sorgente (voiced o treno di impulsi, e unvoiced o rumore gaussiano) ed il filtro (il tratto vocale), riuscendo quindi a modelleare in maniera compatta l'inviluppo spettrale.

3.1.2 Filter Banks e Mel-Scale

Lo smooth con filterbanks riesce ad approssimare piccole variazioni, rendendo le features più robuste e meno rumorose. I filterbanks sono una serie di filtri triangolari che permettono di andare a catturare selettivamente energia da uno specifico range di frequenze, pesando e sommando i bins spettrali in quella regione. Questo processo permette di approssimare lo spettro e fornire una rappresentazione compatta di come l'energia è distribuita lungo le diverse bande di frequenza. La rappresentazione smoothed cattura una forma complessiva dello spettro definita come "envelope", esaltando l'importanza delle features da un punto di vista percettivo.

3.1.3 Mel-Frequency Cepstral Coefficients (MFCCs)

La scala di Mel è una scala percettiva di pitch che approssima il modo in cui gli umani percepiscono i suoni delle frequenze. La scala di Mel è progettata per riflettere una relazione non lineare tra la frequenza fisica e il tono percepito, risultando più sensibile alle basse frequenze, il quale riesce a farci a capire come l'orecchio umano percepisce il suono. Formiamo quindi un filterbank cosicché i centri dei triangoli sono le frequenze che corrispondono a equal

distanza sulla scala di mel. Effettuiamo quindi una analisi short time dove applichiamo alla trasformata DFT i valori pesi del filtro filteranks:

$$MF_m[r] = \frac{1}{A_r} \sum_{l=L_r}^{U_r} |V_r[k] X_m[k]|^2$$

Dove $V_r[k]$ è la funzione di peso per il filtro r-esimo filtrando gli indici della DFT da L_r e U_r e A_r definito come fattore di normalizzazione per il filtro r-esimo di mel.

$$A_r = \sum_{l=L_r}^{U_r} |V_r[k]|^2$$

L'inviluppo di mel chiaramente modella le basse frequenze accuratamente, il quale è molto importante perché è qua che risiedono tutte le formanti. Le alte frequenze sono poco modellate, il quale generalmente è dove non è presente la maggior parte dell'energia.

Tuttavia, se cambiamo il come calcolare la trasformata (cambiamo operatore), usando la Discrete Cosine Transform (DCT), utile per decorrelare dati sequenzialmente correlati, prendiamo la DCT del Log-Mel Spectrum, il quale è conosciuto come **Mel-Frequency Cepstral Coefficient (MFCC)**. Eseguiamo prima una mappatura nella frequenza di mel, poi prendiamo il log spettro e infine applichiamo la IDCT. Ad ogni frame m-esimo, la trasformata del log della magnitudo del risultato dei filtri viene calcolato per formare la funzione $MFCC_m[n]$, definita come:

$$MFCC_m[n] = \frac{1}{R} \sum_{r=1}^{R} log(MF_m[r]) cos\left[\frac{2\pi}{R}\left(r + \frac{1}{2}\right)\right]$$

3.1.4 Delta e Delta Delta

Il segnale acustico è una sequenza di trasizioni tra fonemi, quindi dobbiamo capire quando riconoscerne uno e scinderlo dall'altro. Eseguendo una analisi short time ci portiamo appresso questo fatto, e bisogna tenerne in considerazione quando calcoliamo le features. Tuttavia spesso è più informativo analizzare la forma generale della traccia della features in relazione con le variazioni acustiche. Un metodo comune per estrarre informazioni circa le transizioni per determinare la prima differenza delle feature, conosciute come delta di feature. Specificamente, per una feature f_k al tempo k, il corrispondente delta è definito come:

$$\Delta_k = f_k - f_{k-1}$$

Mentre la delta-delta è definita come:

$$\Delta\Delta_k = \Delta_k - \Delta_{k-1}$$

Queste ulteriori features permettono di definire ed interpretrare il segnale come la derivata prima e la derivata seconda. I delta e i deltas sono componenti classici degli algoritmi di machine learning.

3.2 Speech Embeddings

L'obiettivo degli speech embeddings è quello di trasformare le sequenze di features estratte a partire dei metodi descritti prima in un vettore che cattura le caratteristiche distintive di ogni singolo speaker, rendendolo adatto per compiti come la speaker verification ed identification. Questo vettore processato a partire dalle speech features prende il nome di **embeddings**. In letteratura sono stati usati diversi embeddings, spesso combinando diverse features tra di loro, sebbene oggi la strada più promettente è quella dell'utilizzo di metodi di deep learning. Gli embeddings estratti dai metodi di deep learning rappresentano l'identità di una persona tramite un vettore a dimensione fissa codificato da una espressione vocale di lunghezza variabile.

3.2.1 I-Vectors e Deep Embeddings

Gli i-vectors sono stati per anni una rappresentazione standard e di successo per la speaker recognition. Questi sono vettori a bassa dimensione derivati da un modello statistico chiamato Total Variability Space. Essi catturano sia la variabilità legata al parlante che quella legata al canale di comunicazione. Nonostante la loro comprovata efficacia, mostrano alcune limitazioni, soprattutto quando la durata dell'espressione vocale è breve. L'introduzione delle Deep Neural Networks ha rivoluzionato il campo degli embeddings: l'idea è di addestrare una DNN per produrre embedding che siano altamente discriminativi per l'identità del parlante. Un approccio chiave è *l'estrazione di embedding da segmenti di parlato*. Invece di aggregare informazioni su un'intera espressione vocale all'inizio del processo, le DNN possono elaborare finestre temporali più piccole del segnale (segmenti) e poi aggregare le informazioni a un livello successivo. Ad esempio, gli autori di [2] hanno proposto un sistema in cui gli embeddings vengono estratti da una rete di deep learning. La caratteristica distintiva di questo approccio è l'utilizzo di pooling temporale, che aggrega le informazioni estratte dalla rete su segmenti di input. Questo permettealla rete di essere addestrata per discriminare gli speakers direttamente sulla base del segmento vocale.

3.2.2 X-Vectors e TDNNs

L'evoluzione degli embedding basati su DNN ha portato allo sviluppo degli x-vectors, che rappresentano una delle tecniche più avanzate per la speaker recognition. Come descritto da [3], gli x-vectors sono embedding DNN robusti ottenuti utilizzando una particolare architettura di rete neurale nota come Time Delay Neural Network (TDNN). Le TDNNs sono particolarmente adatte per il trattamento di sequenze temporali come lo speech, in quanto possono catturare contesti temporali più ampi senza la necessità di un numero eccessivo di parametri, grazie all'uso di connessioni ritardate e strati con pooling temporale. L'architettura TDNN consente alla rete di apprendere relazioni a lungo termine tra le caratteristiche vocali a livello di segmenti. Nello specifico, gli x-vectors sono estratti da una DNN.

Chapter 4

Speaker Models e Backend

4.1 Applicazione del Deep Learning

Nel contesto della ASR, come visto anche nel capitolo precedente, si è registrato un notevole interesse verso approcci basati su Deep Learning. Il Deep Learning (DL) grazie ad approcci e metodologie basate sul machine learning in generale, riusciamo a processare meglio le informazioni delle features estratte.
Il deep learning [4] permette di poter elaborare le features ed estrarre rappresentazioni più complesse dei dati, permettendo quindi di realizzare dei veri e propri **Speaker Models**. Gli speaker models costituiscono il nostro database, dove possiamo andare a salvare le caratteristiche o in generale le informazioni importanti per poter riconoscere gli utenti. Una considerazione importante va fatta sui modelli utilizzati, come le Deep Neural Network (DNN), reti ricorrenti con memoria (come le Recurrent Neural Network, RNN, con Long Short Term Memory, LSTM) o reti che applicano filtri di convoluzione Convolutional Neural Network. Sebbene le DNN siano quelle più semplici e di facile interpretazione, permettono semplicemente di estrarre pattern attraverso layer multistrato. Mentre, la combinazione di reti RNN e CNN permette di unire i due principali vantaggi di queste reti: capacità di estrazione di features complesse applicando filtri convolutivi (tramite la CNN), e capacità di memoria e ricordo (tramite la RNN con LSTM). In ogni caso, anche reti più complesse sono state proposte, sia per i task che per estrazione di embedding. Va infatti ricordato che queste reti fungono spesso anche da estrazione di embeddings, sulla base di features estratte precedentemente (ad esempio tramite MFCCs).
Un altro aspetto chaive, riguardando le immagini sugli schemi di funzionamento, è legato alla fase successiva di **pattern matching**. Questa parte spesso viene definita come backend del sistema, dal momento che spesso questa fase avviene dopo che è stato effettuato l'**enrolment** del modello. Spesso come backend si preferisce lavorare con gli embeddings elaborati dalle reti di DL, che permettono di lavorare con informazioni più raffinate. Tipici backend prevedono l'applicazione di modelli come le Gaussian Mixture Models (GMM), che usiamo per estrarre degli scores di similarità con cui valutare score di EER per stabilire l'affidabilità del modello stesso.
Gli autori di [1] ad esempio hanno valutato la possibilità di applicare il Deep Learning nel

task di Speaker Verification, andando a riassumere quali sono i principali modelli utilizzati sia come speaker models che come backend. Una tabella riassuntiva difatti è portata in Fig.4.1. Come si evince dallostudio condotto, spesso ci basiamo su score di similarità con distanza coseno o PLDA, avendo comunque sistemi basati su scores di un backend basato su GMM-UBM (ovvero le **Gaussian Mixture Model - Universal Background Model**), che permette di realizzare un modello probabilistico che rappresenta la distribuzione di dati come una combinazione di distribuzion gaussiane, andando quindi a modellare la voce (gli embeddings) di ogni speaker, difatti rappresentando un databse (Universal Background Model).

TABLE I: COMPARATIVE STUDY OF VARIOUS SPEAKER VERIFICATION SYSTEM BASED ON DNN ARCHITECTURES

Reference	Type of System	Input Features	DNN Type	Score Function	Baseline System	Dataset	Score (%EER)
[10]	Text-dependent	40 log Mel-filter bank coefficients	7-layered, fully-connected	PLDA	UBM/i-vector	NIST SRE'12, Noisy narrowband	1.39
[12]	Text-independent	39 dimension PLP	7-layered RBM	Cosine Distance	GMM-UBM	NIST SRE'05-06	0.88
[14]	Text-independent	60 dimension MFCCs	5-layered	PLDA	UBM/i-vector	NIST SRE'12, Switchboard I, II, III	1.58
[16]	Text-dependent	39 dimension PLP	4-layered, fully-connected	Cosine Distance	UBM/i-vector	Self-created	1.21
[17]	Text-dependent	20 dimension MFCCs	4-layeres, fully-connected	PLDA	UBM/i-vector, GMM-DTW	RSR 2015 Specifically designed for text dependent speaker verification.	0.2
[18]	Text-dependent	39 dimension PLP	4-layered, fully-connected	PLDA	GMM-UBM, d-vector, j-vector	SR 2015	0.54
[19]	Text-dependent	40 dimension MFCCs	7-layered, multi-splice time delay	GPLDA	GMM-UBM	NIST SRE'10	7.2
[20]	Text-independent	Phone-blind & Phone-aware 40 dimensional d-vectors	7-layered, time-delay	PLDA	UBM/i-vector	Fisher dataset, CSLT-CUDGT2014	8.37
[21]	Text-independent	20 dimension MFCCs	4-layered, temporal pooling	PLDA	UBM/i-vector	US English telephonic speech,	5.3

Figure 4.1: Overview del Deep Learning applicato al problema, [1]

4.2 Metodologia del progetto

Per la realizzazione del progetto abbiamo esplorato diverse strade basate sul deep learning, partendo comunque dal problema della speaker identification. Difatti, sulla base della speaker identification, possiamo poi usare la rete basata su deep learning stessa per poter identificare correttamente un utente tramite classificazione in termini di closed-set. Dopodiché siamo passati al task di speaker verification, partendo però dal modello addestrato che adesso funzionerà da generatore di embeddings. Con questi embeddings costruiamo un modello GMM-UBM sulla base dei nostri utenti conosciuti, e valutiamo su utenti non conosciuti,

tramite scores di similarità valutando il rapporto di verosimiglianza. Per valutare il ratio di accettazione o rifiuto, il sistema calcola un valore di soglia che permette di definire il valore di EER ottimale, ricordando che EER viene valutato quando FAR (False Acceptance Rate) e FRR (False Rejection Rate).

4.3 Architettura Proposta

Uno schema esemplificato della metodologia applicata è riportata in Fig.4.2. Lo schema prevede una prima fase iniziale di Speaker Identification, dove a partire dagli speakers autorizzati (closed-set), estraiamo features (ad esempio MFCCs) ed addestriamo uno speaker models basato su Deep Learning. Dopodiché, lo stesso modello verrà usato come estrattore di embeddings per poter successivamente addestrare un modello di GMM-UBM (Universal Background Model, il nostro backend) per poter realizzare un sistema che permetta di fare acceptance o rejection di speakers non autorizzati. L'idea generale è quello di applicare quanto fatto finora in letteratura per poter unire i due task in uno unico, riconoscendo i nostri utenti e poi fare classificazione ed associazione. Per quanto riguarda lo score ci basiamo

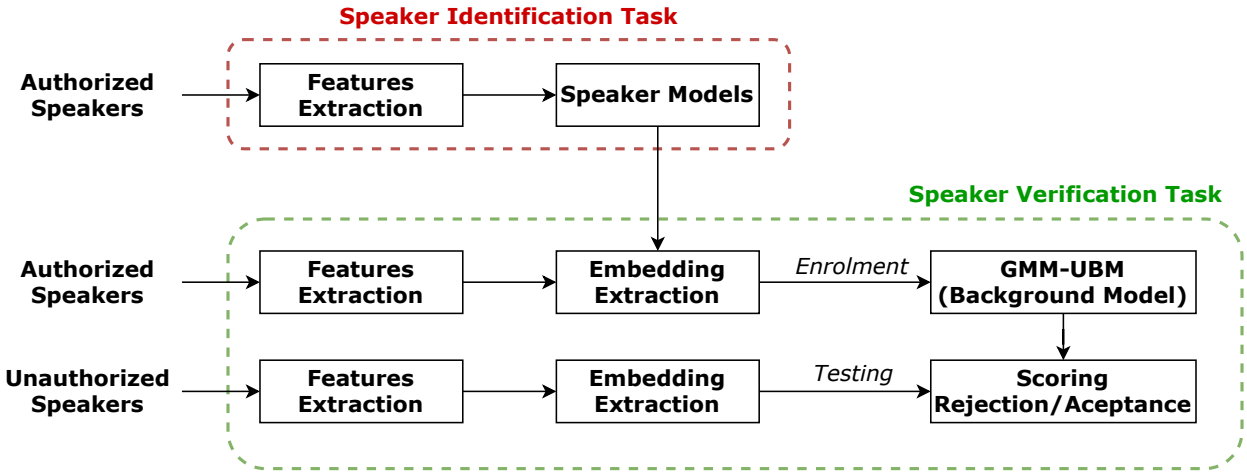

Figure 4.2: Architettura del progetto proposta

su un valore di soglia sull'EER (Equal Error Rate). La soglia viene utilizzata per decidere quando classificare un'istanza come genuina o impostore. Nel contesto di un sistema di autenticazione biometrica o di rilevamento delle frodi, la soglia è il valore sopra o sotto il quale una decisione viene presa. Calcolo della soglia (Threshold) alla Equal Error Rate (EER), ovvero il punto in cui il tasso di falsi positivi (FPR) e il tasso di falsi negativi (FNR = 1 - TPR) sono uguali. Si identifica il valore della soglia corrispondente al punto in cui (FPR - FNR) è minimo. Dopo aver calcolato la soglia all'EER, questa viene utilizzata per prendere decisioni di classificazione:

- Se il punteggio di un campione è superiore alla soglia, viene considerato normale (positivo)
- Se è inferiore alla soglia, viene considerato impostore

Un particolare utilizzo di questo sistema potrebbe essere quello di un sistema di riconoscimento biometrico di una azienda, dove solo alcuni dipendenti sono autorizzati ad accedere (controllo degli accessi), mentre gli altri non devono accedere. E qualora un utente dovesse essere autorizzato ad accedere, quindi viene verificata la sua identità, il sistema di identificazione associerà un ID che permette di associare a quell'utente i privilegi specifici.

Chapter 5

Sezione Sperimentale

5.1 Dataset utilizzato

Per quanto riguarda i dataset, è stata condotta una prima fase di ricerca che portato all'utilizzo di LibriSpeech. LibriSpeech è un dataset audio utilizzato per l'addestramento e la valutazione di modelli di automatic speech recognition (ASR). Il suo punto di forza è la grande quantità di dati disponibili, derivati da audiolibri di pubblico dominio, il che lo rende ideale per sviluppare sistemi che comprendano l'inglese parlato in modo naturale. Sono presenti speakers differenti, bilanciati in termini di sesso e di durata delle tracce audio. Gli audio vengono forniti già normalizzati e puliti da eventuale rumore. Il dataset è molto adatto per task di Speaker identification e verification in quanto gli audio sono molto chiari e ben distinguibili.
I file audio che compongono LibriSpeech provengono dagli audiolibri registrati nel progetto LibriVox, una piattaforma dove volontari leggono e registrano opere letterarie di dominio pubblico. Grazie a questa fonte, il dataset offre un ampio spettro di voci, con accenti e tonalità diverse, riflettendo meglio la variabilità del parlato umano, permettendo quindi di avere dati il quanto più possibili generalizzanti.

5.2 Dettagli Esperimenti

In termini implementativi, per quanto riguarda i vari moduli o componenti, le configurazioni sono riportate in tabella Tab.5.1. In particolare si è voluto dare maggior enfasi alla sperimentazione di diversi modelli, diversi embedding e sopratutto features differenti. Ogni modello viene addestrato su un training e testing set comune, composto dagli audio degli speaker autorizzati della top 10 di uomini e donne. Come si evince dalla tabella, per quanto riguarda le speech features da dare in input ai modelli comunque applichiamo analisi short-time. L'analisi short-time deriva dal fatto che vogliamo estrarre le caratteristiche di uno speaker a livello di frame (che possiamo pensare ad un singolo fonema), cercando di estrarre le features spaziali e temporali caratteristiche. Proprio per questa possibilità, applichiamo diversi modelli che cercano ognuno di catturare questi aspetti. In particolare,per quanto riguarda la I-DNN,

Feature Extraction		***Speaker Identification***	***Speaker Verification***	
Features used	**Frame Analysis**	**Model Used**	**Embedding**	**Backend**
25 MFCCs	25ms frame length 10ms frame hop	DNN	Bottle-neck layer	GMM-UBM
25 MFCCs	25ms frame length 10ms frame hop	DNN	Last Hidden layer	GMM-UBM
25 MFCCs	25ms frame length 10ms frame hop	RNN	Mean Hidden Layers	GMM-UBM
25 MFCCs	25ms frame length 10ms frame hop	CNN+LSTM	Mean Hidden Layers	GMM-UBM
20 MFCCs	25ms frame length 10ms frame hop	I-DNN	Segment Embedding	GMM-UBM
24 MFCCs, 24 Delta-MFCCs VAD 30% energy	25ms frame length 10ms frame hop	TDNN	X-embedding	GMM-UBM

Table 5.1: Riassunto degli esperimenti condotti

ispirata alla rete utilizzata per il calcolo degli I-Embeddings, validi sostituti degli I-Vectors (obsoleti e superati), descritta in [2], andiamo ad estrarre embedding a livello di segmento, oltre che di singolo frames. Invece, gli X-Embedding si basano più sulla rete descritta in [3], ovvero una Temporal Delay Neural Network.

5.3 Risultati sperimentali

Di seguito verranno riportate sia i risultati della Speaker Identification che Speaker Verification (in termini di EER). Per quanto riguarda i risultati della prima, si riporta nel complessivo le prestazioni del sistema in termini di macro F1,Accuracy, e Precision mentre i risultati sulla precisione del singolo speaker sono salvati nei rispettivi notebook python. Discorso analogo per quanto rigaurda il valore di EER.

Feature Extraction		***Speaker Identification***	***Metriche***		
Features used	**Frame Analysis**	**Model Used**	**Acc**	**F1**	**Precision**
25 MFCCs	25ms frame length 10ms frame hop	DNN	1.0	1.0	1.0
25 MFCCs	25ms frame length 10ms frame hop	DNN	1.0	1.0	1.0
25 MFCCs	25ms frame length 10ms frame hop	RNN	0.97	0.96	0.97
25 MFCCs	25ms frame length 10ms frame hop	CNN+LSTM	0.99	0.99	0.99
20 MFCCs	25ms frame length 10ms frame hop	I-DNN	0.98	0.98	0.98
24 MFCCs, 24 Delta-MFCCs VAD 30% energy	25ms frame length 10ms frame hop	TDNN	0.94	0.94	0.95

Table 5.2: Speaker Identification

Feature Extraction		***Speaker Identification***	***Speaker Verification***		***Metrics***
Features used	**Frame Analysis**	**Model Used**	**Embedding**	**Backend**	**EER**
25 MFCCs	25ms frame length 10ms frame hop	DNN	Bottle-neck layer	GMM-UBM	0.03
25 MFCCs	25ms frame length 10ms frame hop	DNN	Last Hidden layer	GMM-UBM	0.13
25 MFCCs	25ms frame length 10ms frame hop	RNN	Mean Hidden Layers	GMM-UBM	0.12
25 MFCCs	25ms frame length 10ms frame hop	CNN+LSTM	Mean Hidden Layers	GMM-UBM	0.15
20 MFCCs	25ms frame length 10ms frame hop	I-DNN	I-embedding	GMM-UBM	0.16
24 MFCCs, 24 Delta-MFCCs VAD 30% energy	25ms frame length 10ms frame hop	TDNN	X-embedding	GMM-UBM	0.48

Table 5.3: Speaker Verification

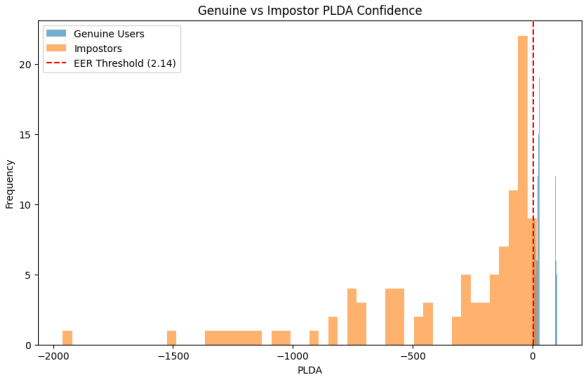

Figure 5.1: DNN (1)

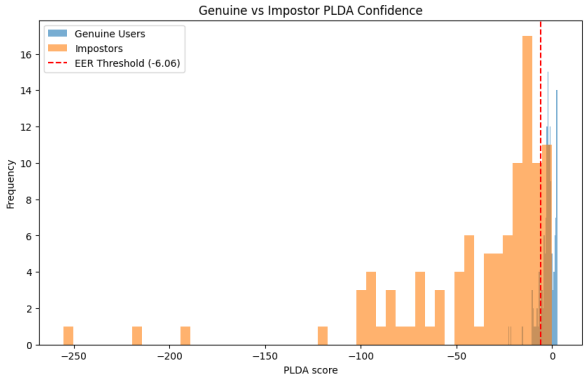

Figure 5.2: DNN (2)

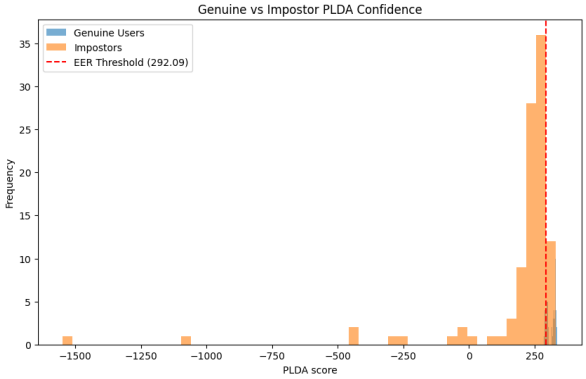

Figure 5.3: RNN

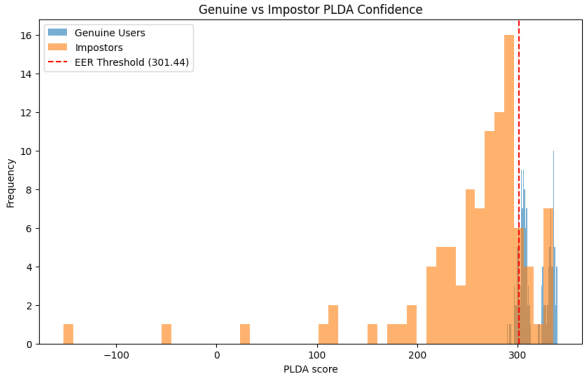

Figure 5.4: CNN+LSTM

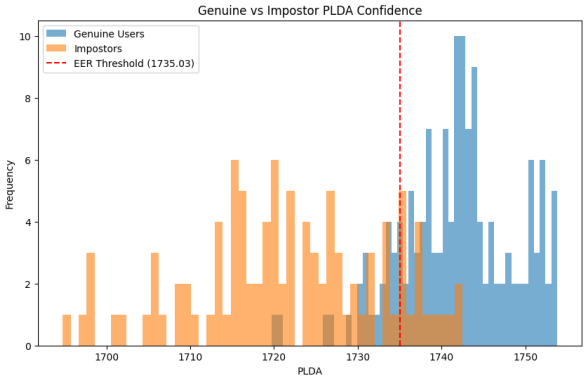

Figure 5.5: I-DNN

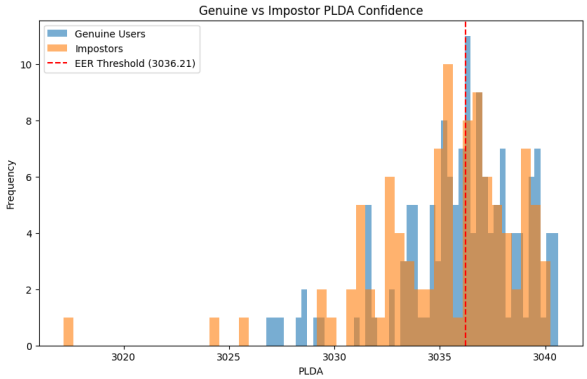

Figure 5.6: X-DNN

5.4 Conclusioni

Alla luce delle sperimenazioni effettuate, dalla tabella Tab.5.2 vediamo come effettivamente tutti i modelli si comportano molto bene nella identificazione degli utenti. Tuttavia va precisato che il dataset non è di grandi dimensioni, difatti abbiamo a disposizione circa 100 audio per ogni speaker per training, il che sicuramente è molto limitante come numero per riuscire ad apprendere caratteristiche raffinate. Tuttavia, è nella fase di sperimentazione della Speaker Verification (riportate in tabella Tab.5.3) dove notiamo che la che la rete che presenta le migliori prestazioni sembra essere la DNN tradizionale con estrazione degli embedding dal Bottleneck.

Difatti entrambe le DNN presentate, in termini di distribuzione degli scores da parte del backend (vedi Fig.5.1 e Fig.5.2) siano molto precise nel trovare embedding abbastanza discriminativi. Discorso diverso invece va fatto per le due ultime reti, I-DNN e X-DNN, dove vediamo che la distribuzione è abbastanza spalmata e sovrapposta, sintomo che la rete non riesce ancora bene a distinguere tra le varie identità. Questo lo possiamo notare sopratutto nella X-DNN, che riporta il peggior risultato in termini di EER. Questo potrebbe essere dovuto al fatto che la rete presenta un numero di parametri abbastanza elevato e rischia di overfittare quasi subito, andando difatti a riconoscere bene gli speaker, ma nonr riuscendo a discriminare quelli "reali" da quelli "estranei".

Concluendo, possiamo dire che questi sistemi basati su deep learning sul dataset attualmente in uso permettono in maniera precisa di identificare gli speaker, quindi si prestano molto bene per task in closed-set, ma non possiamo sicuramente dire lo stesso per quanto riguarda il sistema di verifica, che presenta un EER troppo alto, sintomo che ci sarebbero tanti falsi positivi che riuscirebbero ad entrare nel sistema, venendo identificati erroneamente come utenti legittimi

Bibliography

[1] Amna Irum and Ahmad Salman. "Speaker verification using deep neural networks: A". In: *International Journal of Machine Learning and Computing* 9.1 (2019).

[2] David Snyder et al. "Deep neural network embeddings for text-independent speaker verification." In: *Interspeech*. Vol. 2017. 2017, pp. 999–1003.

[3] David Snyder et al. "X-vectors: Robust dnn embeddings for speaker recognition". In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2018, pp. 5329–5333.

[4] Sreenivas Sremath Tirumala and Seyed Reza Shahamiri. "A review on deep learning approaches in speaker identification". In: *Proceedings of the 8th international conference on signal processing systems*. 2016, pp. 142–147.

[5] Roberto Togneri and Daniel Pullella. "An overview of speaker identification: Accuracy and robustness issues". In: *IEEE circuits and systems magazine* 11.2 (2011), pp. 23–61.

[6] Youzhi Tu, Weiwei Lin, and Man-Wai Mak. "A survey on text-dependent and text-independent speaker verification". In: *IEEE Access* 10 (2022), pp. 99038–99049.