
**UNIVERSITÀ DEGLI STUDI DI CATANIA**

Dipartimento di Matematica e Informatica

Corso di Laurea Triennale in Informatica

*Samuele Maria Gallina*


# Costruzione di un modello di classificazione per il riconoscimento di voci nei deepfake

Relazione Progetto Finale


Relatori: Prof. Filippo Stanco

Prof. Dario Allegra

Correlatore: Dott. Stefano Borzì



Anno Accademico 2021 - 2022

# Abstract

Nell'era delle fake news emerge una tecnologia nuovissima, innovativa ma allo stesso tempo molto pericolosa se usata maliziosamente: il deepfake.
I deepfake possono essere chiamati anche media sintetici ovvero dei media ricreati e/o simulati artificialmente.
Con un deepfake si può far credere che qualcuno abbia detto o fatto qualcosa quando questo non è vero. Celebre è il video sul deepfake di Barack Obama presente su Youtube dal 17 aprile del 2018.
Fortunatamente il video aveva solo scopi benefici, infatti esso non mirava a far credere che Barack Obama avesse detto qualcosa in particolare, anzi, il video mirava per l'appunto a far capire alle persone la potenza e la pericolosità dei deepfake.
I deepfake possono essere anche utilizzati per creare file audio che simulano la voce di una vittima per creare dichiarazioni false e quindi diffamarla. Con l'emergere del deepfake è stato necessario ricercare dei metodi e delle tecniche per quella che viene definita deepfake detection. In questo campo si sono adoperati soprattutto i laboratori di ricerca forense, infatti i falsi contenuti multimediali possono essere ricreati artificialmente oltre che per scopi politici, criminali o più in generale per creare fake news, anche per creare false prove da utilizzare in campo giudiziario.
Per queste motivazioni la ricerca su possibili strumenti dediti il deepfake detection è in pieno sviluppo. Lo scopo del seguente elaborato è introdurre il lettore nell'ambito della sintetizzazione di voci umane (si parla di deepfake qualora le voci siano indistinguibili da vere voci umane) e del deepfake detection. Viene discussa brevemente la storia delle tecnologie in grado di sintetizzare voci umane e delle tecnologie dedite al deepfake audio detection.

Vengono attenzionate alcune tecnologie moderne in grado di classificare i file audio e vengono analizzati i risultati ottenuti da esse per poi discutere un ulteriore metodo sperimentato per la creazione di un modello di classificazione dei file audio. Il metodo prevede un'analisi sulle caratteristiche estratte dai file audio e l'allenamento e verifica di vari modelli di classificazione presenti nella libreria Scikit-Learn[1] di Python oltre a due modelli di classificazione naive. I modelli moderni più performanti di deepfake audio detection sono basati sulle Neural Network o sulle Deep Neural Network, esse in genere sono molto performanti nella classificazione ma hanno tre principali debolezze: servono moltissimi dati per l'allenamento, quest'ultimo rappresenta un'operazione molto onerosa che comporta lunghi tempi di computazione e anche la classificazione vera e propria su dati sconosciuti è un'operazione onerosa. L'obiettivo è creare un modello di classificazione non basato sulle reti neurali che sfrutti delle semplici caratteristiche statistiche dei segnali audio sintetici generati tramite tecnologie deepfake.

# Indice

# Capitolo 1

# Stato dell'arte

## 1.1 Stato dell'arte

Gli algoritmi di intelligenza artificiale dediti alla generazione di voci umane sintetiche sono una tipologia di deepfake in grado di emulare la voce di una persona e creare dei contenuti multimediali riproducendo qualsiasi espressione vocale del soggetto di cui è stata clonata la voce.
I primi sintetizzatori di voce umana generavano voci per nulla simili a quella umana, un esempio di primo sintetizzatore vocale è Er Finestra [2], utilizzato da Radio Deejay all'inizio degli anni 2000. Le tecniche di sintetizzazione vocale erano di scarsa qualità, infatti anche un essere umano sarebbe facilmente in grado di capire che la voce generata da Er Finestra non è di origine umana. Successivamente sono stati sviluppati algoritmi sempre più sofisticati ed accurati nella sintetizzazione e clonazione della voce, essi ad oggi riescono a sintetizzare voci deepfake molto realistiche, praticamente indistinguibili dalle voci reali.
La generazione di audio deepfake in realtà nasce con obiettivi benefici come la creazione di assistenti vocali o aiuto a persone con difficoltà a comunicare a causa di disabilità.
Come qualunque innovazione, anche la generazione di audio deepfake viene usata per scopi malevoli.
La generazione di audio deepfake non è affidata unicamente a sistemi auto-

matici di sintetizzazione della voce umana come WaveNet [3], per rendere ancora più realistica la voce presente in una registrazione audio generata dal sistema di sintetizzazione, essa viene ritoccata ulteriormente, però questa volta non con sistemi automatici.
Ad esempio un file audio generato da WaveNet potrebbe essere poi ritoccato usando anche semplici software di manipolazione audio come Audacity [4].
Un caso d'uso potrebbe essere ritagliare manualmente una parte di un file audio dove è presente la voce della vittima (colui di cui si vuole emulare la voce) e posizionare il frammento ottenuto all'interno del file audio generato da WaveNet o un qualunque altro sistema di sintetizzazione e/o clonazione di voce umana.
I metodi di generazione di audio deepfake sono molteplici, essi fanno affidamento a sistemi basati sulla conversione della voce, la sintesi Text-to-Speech, modelli generativi e generatori di voce simile a quella umana basati sulle reti neurali.
Al fine di migliorare la ricerca nell'ambito della classificazione dei file audio per la deepfake audio detection sono state create delle challenge apposite, come ad esempio ASVspoof [5] che dal 2015 ogni due anni lancia una challenge dove l'obiettivo di ogni partecipante è quello di creare il miglior modello in grado di individuare audio deepfake.
Per la creazione di questi modelli vi sono più approcci, ad esempio l'allenamento di algoritmi di Machine Learning o Deep Learning. Se si sceglie un approccio di Machine Learning è necessaria l'implementazione di uno step preliminare, l'estrazione delle feature. Le feature di un file audio sono le sue caratteristiche specifiche che possono essere d'aiuto nella distinzione tra audio con voce sintetica e reale. Sfortunatamente le feature che sono discriminanti per un dataset potrebbero essere non discriminanti per altri dataset, questo è uno dei principali fattori di difficoltà nel creare un sistema universale di classificazione per l'identificazione di voci deepfake [6]. Questa difficoltà è dovuta alle diverse tecniche utilizzate per generare i dataset.
Una determinata tecnica potrebbe essere in grado per una feature di generare valori molto simili a quelli delle voci reali, mentre per un'altra feature potrebbe generare valori molto lontani, un'altra tecnica di sintetizzazione

potrebbe comportarsi in maniera opposta.

## 1.2 Sintetizzatori di voce umana nel passato

### 1.2.1 Il Voder

La macchina veniva comandata manualmente tramite una tastiera da un operatore e tramite dei circuiti elettronici, mostrati in Figura 1.1, veniva generata la voce sintetica.
La macchina era estremamente difficile da utilizzare, infatti richiedeva mesi di pratica da parte dell'operatore, e inoltre la voce riprodotta dal circuito aveva un suono sgradevole ed era facilmente distinguibile da una voce reale.

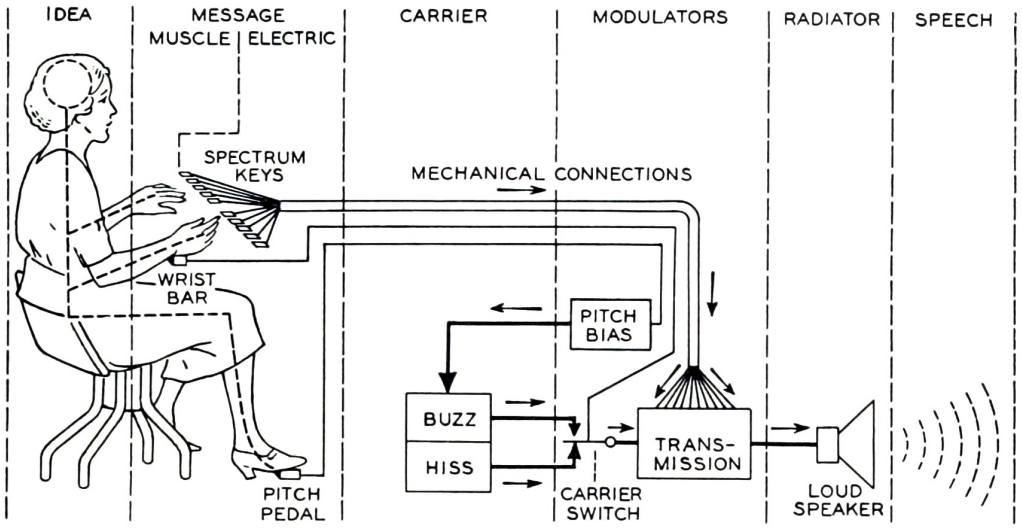


**Figura 1.1:** Schema del circuito del Voder.

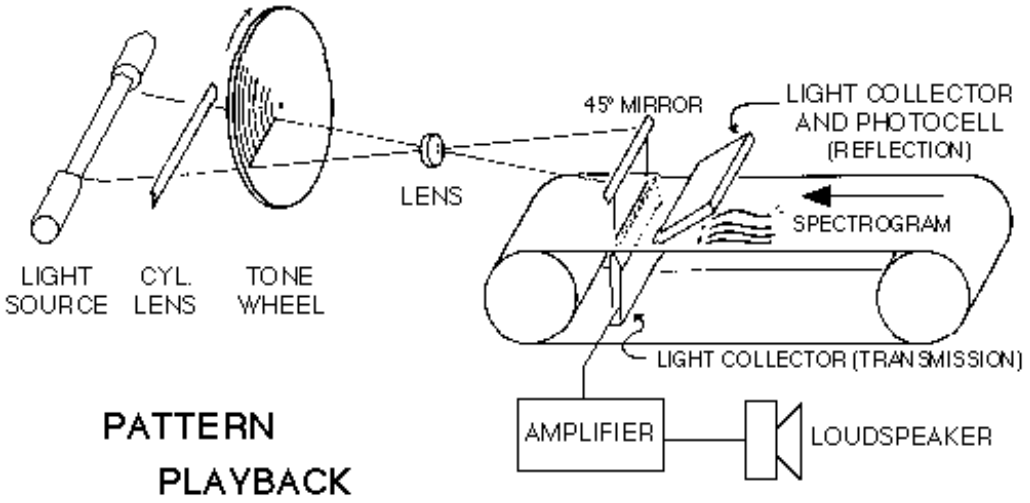


**Figura 1.2:** Schema del Pattern Playback.

### 1.2.2 Il Pattern Playback

Nel 1950 Franklin S. Cooper insieme ai suoi collaboratori creò il dispositivo chiamato poi Pattern playback, Figura 1.2. In breve, la macchina prende in input un'immagine contenente un pattern acustico e ottiene in output il relativo spettrogramma per poter poi generare il suono.
La macchina utilizza una sorgente luminosa ad arco che viene diretta verso un disco rotante formato da 50 tracce concentriche. Le trasparenze delle tracce variano secondo un certo ordine per poter riprodurre 50 parziali con una frequenza fondamentale pari a 120 Hz. La luce viene poi proiettata verso l'immagine di uno spettrogramma, la riflettenza corrisponde alla pressione sonora di ogni parziale del segnale, la luce viene riflessa verso una cella fotovoltaica che converte la variazione luminosa nella variazione di pressione sonora [7]. Il Voder e il Pattern Playback sono solo due dei possibili antenati dei moderni sistemi di sintetizzazione della voce, in particolare si possono considerare antenati dei moderni Text-to-Speech e non dei sistemi di Voice Cloning.

## 1.3 La moderna sintesi della voce umana

Dagli anni '70 la sintetizzazione vocale inizia ad abbracciare il ramo dell'informatica oltre a quello della fisica e della matematica poiché compaiono i primi veri e propri software per la sintesi vocale (o computer dedicati proprio a quello scopo). Alcuni esempi di primi software di sintesi vocale

- Software Automatic Mouth (SAM) [8]
- MacInTalk [9]
- Microsoft Agent [10]

Importanti passi in avanti sono stati fatti rispetto ai primi sintetizzatori vocali. Nacquero nuove tecniche di sintetizzazione come la **concatenazione** e **la parametrizzazione**. La **concatenazione** consiste semplicemente nel concatenare tanti file audio di buona qualità ottenuti tramite registrazione di voci vere e non sintetiche. Ad esempio la frase "Ciao, mi chiamo Maria" potrebbe essere sintetizzata concatenando 4 file audio contenenti ognuno una delle 4 parole registrate. La **concatenazione** non fu più ulteriormente utilizzata e/o migliorata in quanto presenta problemi di scalabilità e consistenza [11]. La tecnica che venne poi più utilizzata fu la **parametrizzazione**, questo metodo consiste nell'estrarre le caratteristiche linguistiche dall'input testuale e da queste estrarre quelle che chiamiamo vocoder feature da utilizzare per generare un audio vero e proprio tramite i vocoders. I primi vocoders erano degli ibridi tra HMM(hidden-Markov model) e GMM(Gaussian mixture model), questi avevano il pregio di generare delle voci non sgradevoli all'orecchio umano (come invece succede per Voder e Pattern Playback), ma avevano il difetto di generare voci comunque ancora facilmente distinguibili dalle voci umane, in gergo si potrebbero definire voci "robotiche" [12]. Il miglioramento dei Text-to-Speech parametrici non è dovuto a modifiche al flusso di lavoro, esso è rimasto sostanzialmente identico, ciò che ha comportato importanti miglioramenti è stato l'utilizzo delle reti neurali come vocoders. Oggi la tendenza è quella di utilizzare più tecnologie insieme per generare il risultato desiderato, ad esempio molti moderni Text-to-Speech parametrici utilizzano

contemporaneamente Tacotron e WaveNet come mostrato in Figura 1.3.

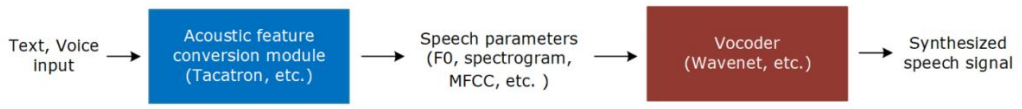


**Figura 1.3:** Un moderno Text-to-Speech parametrico che utilizza tecnologie come WaveNet e Tacotron.

Negli ultimi anni la qualità dei Text-to-Speech e dei sistemi di Voice cloning è migliorata considerevolmente con la nascita di tecnologie come WaveNet, Tacotron e DeepVoice, questi strumenti riescono a ricreare voci molto simili a quelle umane, anche un essere umano avrebbe difficoltà a capire se un file audio è stato ricreato o no da queste tecnologie o se è una vera registrazione audio di un'altra persona.
Si riporta lo stato dell'arte dei modelli e strumenti di sintetizzazione della voce umana [4].

**Tabella 1.1:** Panoramica sullo stato dell'arte della generazione di audio sintetici

| **Metodo** | **Tecniche** | **Feature** | **Dataset** |
|---|---|---|---|
| WaveNet [3] | Deep Neural Network | linguistic features fundamental frequency (log F0) | VCTK (44 hrs) |
| Tacotron [13] | Encoder-Decoder with RNN | Deep features | Private (24.6 hrs.) |
| Deep Voice 1 [14] | Deep Neural Network | linguistic features | Private (20 hrs.) |
| Deep Voice 2 [15] | RNN | Deep features | VCTK (44 hrs.) |

**Tabella 1.1:** Panoramica sullo stato dell'arte della generazione di audio sintetici

| **Metodo** | **Tecniche** | **Feature** | **Dataset** |
|---|---|---|---|
| DeepVoice3 [16] | Encoder-decoder | Deep features | Private (20 hrs.), VCTK (44 hrs.), LibriSpeech ASR (820 hrs.) |
| Parallel WaveNet [17] | Feed-forward Neural Network with dilated causal convolutions | linguistic features | Private |
| VoiceLoop [18] | Fully-connected Neural Network | 63-dimensional audio features | VCTK (44 hrs.), Private |
| Tacotron2 [19] | Encoder-decoder | linguistic features | Japanese speech corpus from the ATR Ximera dataset (46.9 hrs.) |
| Arik et al. [20] | Encoder-decoder | Mel spectrograms | LibriSpeech (820 hrs.), VCTK (44 hrs.) |
| Jia et al. [21] | Encoder-decoder | Mel spectrograms | LibriSpeech (436 hrs.), VCTK (44 hrs.) |
| Luong et al. [22] | Encoder-decoder | Mel spectrograms | LibriSpeech (245 hrs.), VCTK (44 hrs.) |
| Chen et al. [23] | Encoder + Deep Neural Network | Mel spectrograms | LibriSpeech (820 hrs.), Private |
| Cong et al. [24] | Encoder-decoder | Mel spectrograms | MULTI-SPK CHiME-4 |

## 1.4 Stato dell'arte della classificazione di file audio per il riconoscimento di voci sintetiche

Le tecniche di creazione di modelli in grado di individuare audio deepfake sono principalmente due, ovvero l'utilizzo di modelli di Machine Learning dopo una preliminare estrazione delle feature e l'utilizzo di modelli di Deep Learning. Come accennato in Sezione 1.1, l'estrazione delle feature può risultare molto difficoltosa in quanto non si può sapere a priori quali caratteristiche siano discriminanti in un certo dataset.
Per questo motivo i modelli di Deep Learning risultano più vantaggiosi, le reti neurali apprendono in modo autonomo come analizzare dati grezzi e come svolgere un compito, infatti non è prevista una preliminare estrazione delle caratteristiche nelle Deep Neural Network (DNN).
L'estrazione delle caratteristiche avviene in modo automatico [25].
Si riporta in Tabella 1.2 lo stato dell'arte riguardo la classificazione binaria dei file audio per il riconoscimento delle voci sintetiche [26] [4]. Nella tabella sono riportati vari modelli e le loro relative metriche, per questo problema di classificazione viene spesso usata la metrica Equal Error Rate (EER), ma non è l'unica, anche altre metriche come l'Accuracy (ACC) e l'Area Under the Curve (AUC) vengono utilizzate. Si assuma che le due etichetta siano "positivo" e "negativo". L'EER rappresenta un compromesso tra il False Acceptance Rate (FAR) e il False Rejection Rate (FRR), essi rappresentano rispettivamente la percentuale di negativi valutati erroneamente come positivi e la percentuale di positivi valutati erronamente come negativi. L'EER è preferibile al classico Error Rate, ovvero la percentuale di osservazioni classificate in modo errato, qualora il dataset sia molto sbilanciato, e come l'Error Rate esso rappresenta un buon classificatore se assume valori bassi. Il calcolo dell'EER viene approfondito in Sezione 3.3. L'Accuracy rappresenta la percentuale di successo nel test di classificazione, ovvero, se avesse un valore del 90% significherebbe che il 90% delle osservazioni del test sono state classificate correttamente. Il calcolo della metrica consiste nel rapporto tra

il numero di osservazioni classificate correttamente e il numero totale delle osservazioni. L'Accuracy è una metrica molto semplice e molto utilizzata ma non rappresenta un buon metodo di valutazione nel caso in cui il dataset utilizzato non è bilanciato. L'AUC, come si intuisce dal nome stesso, è una metrica rappresentante l'area sotto a una determinata curva, le curve solitamente utilizzate sono la Receiver Operating Characteristic (ROC) e la Precision-Recall (PR). Quest'ultime sono dei grafici utilizzati per descrivere le performance di un classificatore al variare di una soglia, la ROC rappresenta il True Positive Rate, detto anche Recall, ovvero il numero in percentuale di osservazioni positive classificate correttamente in funzione del False Positive Rate, ovvero il numero in percentuale di osservazioni negative classificate come positive, al variare della soglia. La PR rappresenta invece la Precision, calcolata come il rapporto tra il numero di osservazioni positive classificate correttamente e il numero totale di osservazioni classificate come positive, in funzione della Recall al variare della soglia. L'AUC assume valori tra 0 e 1, un valore vicino ad 1 indica un buon classificatore.

**Tabella 1.2:** Panoramica sullo stato dell'arte delle tecniche per la classificazione binaria dei file audio per il riconoscimento di voci sintetiche.

| **Autori** | **Tecnica** | **Metriche** | **Dataset** |
|---|---|---|---|
| Nagar. et al.[27] | SVM | EER=11.5% | ASVspoof2017 |
| Gunen. et al.[28] | GMM | EER=8.68(TLC-AM), 11.30 (TLC-FM) | ASVspoof2017 |
| Witkowski et al.[29] | GMM | EER=5.13(CQCC), 3.38(Cepstrum), 4.16(IMFCC), 16.76(MFCC), 6.37(LPCCres) | ASVspoof2017 |
| Saranya et al.[30] | GMM | EER=19.36 | ASVspoof2017 |
| Huang et al.[31] | DenseNet-BiLSTM | EER=0.53% EER=6.43% | BTAS2016 ASVspoof2017 |

| | | | |
|---|---|---|---|
| Wu et al.[32] | LCNN | EER=4.07% | ASVspoof2019 |
| Lai et al.[33] | ResNet | EER= 8.99% | ASVspoof2017 |
| Li et al.[34] | Res2Net | EER=2.502 | ASVspoof2019 |
| Yi et al.[35] | GMM<br>LCNN | EER=19.22(GMM)<br>EER=6.99(LCNN) | ASVspoofDatabases |
| Das et al.[36] | LCNN | EER=3.13 | ASVspoof2019 and VCTK |
| Aljasem et al.[37] | Asymmetric bagging | EER=5.22 | ASVspoof2019 and VSDC |
| Ma et al.[38] | CNN | EER=9.25 | ASVspoof2019 |
| AlBadawy et al.[39] | logistic regression classifier | AUC=0.99 | Own dataset |
| Singh et al.[40] | Quadratic SVM | ACC=96.1% | Own dataset |
| Gao et al.[6] | ResNet | EER=4.03 | ASVspoof2019 |
| Aravind et al.[41] | ResNet | EER=5.87 | ASVspoof2019 |
| Monteiro et al.[42] | LCNN ResNet | EER=6.38 | ASVspoof2019 |
| Chen et al.[43] | ResNet | EER=1.81 | VCC2018 ASVspoof2019 |
| Zhang et al.[44] | TEResNet | EER=5.89 ERR=3.99 | ASVspoof2019 FoR-norm |
| Zhang et al.[45] | ResNet-18+OC-softmax | EER=2.19 | ASVspoof2019 |

| | | | |
|---|---|---|---|
| Gomez-Alanis et al.[46] | LCG-RNN | EER=6.28 | ASVspoof2015 and 2017 |
| Hua et al.[47] | Res-TSSDNet | EER=1.64 | ASVspoof2015 and 2019 |
| Jiang et al.[48] | CNN | EER=5.31 | ASVspoof2019 |
| Wang et al.[49] | DNN | EER=0.021 | FoR, Sprocket-VC, MC-Text-to-Speech |
| Stefano Borzì et al.[26] | AdaBoost | ACC = 89.7%, AUC = 63.2%, EER = 5% | ASVspoof2019 |
| Yamag. et al.[50] | RawNet architectu-re | EER=22.38% | ASVspoof2021 |
| Chen et al.[51] | ECAPA-TDNN | EER=20.33% | ASVspoof2021 |
| Muller et al.[52] | FCNN RawNet2 | EER=17.42% EER=6.28% | ASVspoof2019 and 2021 |

Come riporta Parav Nagarsheth et. al, Support Vector Machine 3.3.3 si rivela un buon modello ottenendo un EER pari a 11,5% a patto che vengano estratte le feature da parte di una DNN. Il modello costruito si occupa di estrarre le feature HFCC e CQCC, costruire delle feature ibride tra queste due e utilizzare quest'ultime per l'addestramento del modello Support Vector Machine. Spesso alcune feature oltre ad essere non utili nella discriminazione potrebbero essere addirittura dannose per un corretto allenamento del modello, a volte è necessaria una feature subselection per evitare di utilizzare caratteristiche del file audio dannose per la creazione un corretto classificatore. Un esempio è quello riportato dagli esperimenti di Marcin Witkowski et al. dove vengono analizzate cinque diverse feature, ovvero CQCC, Cep-

strum, IMFCC, MFCC e LPCCres. Per ognuna di esse viene costruito un modello basato su Gaussian Mixture Model e in particolare per ognuna delle feature vengono costruiti 6 diversi classificatori basati su GMM. Ognuno dei 6 classificatori prevede una feature subselection basata sulle frequenze. Il primo classificatore utilizza le feature estratte considerando solamente le frequenze 16-8000 (Hz), gli altri cinque intervalli sono 16-1000, 1000-2000, 2000-4000, 4000-8000 e 6000-8000. Gli EER dei 6 classificatori GMM basati su Cepstrum sono rispettivamente 8.52%, 38.48%, 38.98%, 39.60%, 5.27% e 3.38%. Si capisce quindi che nel dataset ASVspoof 2017 applicare una feature subselection in modo tale che vengano utilizzate solamente le alte frequenze risulta essere molto vantaggioso. Grazie ai risultati ottenuti da Lian Huang e Chi-Man Pun si deducono due cose, l'importanza del dominio nel tempo e il vantaggio delle Deep Neural Network in questo particolare problema di classificazione. La tecnologia utilizzata viene chiamata DenseNet-BiLSTM. Quest'ultima è una Deep Neural Network dove i primi due layer sono un Convolutional Layer e un Pooling layer che prendono in input l'onda del file audio e riducono il numero di feature. Dopo questi primi due layer la rete è composta da tre blocchi densi alternati da due blocchi di transizione. Un blocco denso è un insieme di quattro layer tutti connessi (da qui il nome "denso") che sono in ordine: BottleNeck, ReLu, Conv2d e Concatenation. Il blocco di transizione invece è formato da due layer ovvero Convolution 2D e Average Pooling 2D. Dopo i blocchi densi, alternati dai blocchi di transizione, troviamo due blocchi BiLSTM. Le LSTM sono delle reti neurali dotate di due memorie, a lungo termine e breve termine, in queste reti neurali i calcoli relativi a un dato non dipendono solo da quel dato ma anche dai precedenti (la memoria a breve termine tiene conto di calcoli immediatamente precedenti, la memoria a lungo termine tiene conto di calcoli anche molto lontani nel tempo). In particolare nelle BiLSTM il dato "attraversa" due cammini, dal futuro (il layer del blocco BiLSTM più lontano dal layer di input) verso il passato (il layer del blocco BiLSTM più vicino dal layer di input) e viceversa, il vantaggio di questa rete è il poter ottenere e mantenere informazioni sia dal passato (layer precedenti nel cammino) sia dal futuro (layer successivi). La rete neurale ha ottenuto un EER pari a 0,53% con il test effettuato sul data-

set BTAS2016 e un EER pari a 6,43% su ASVspoof2017. Altre architetture molto utilizzate sono le Residual Network (ResNet) 2.2.3 e le Convolutional Neural Network (CNN), esse sono delle reti formate da tre principali tipi di layer, ovvero i convolutional layer, i pooling layer e i fully connected layer. Oltre alle ResNet e le CNN vengono usate in questo ambito alcune varianti come ad esempio Res2Net, TEResNet e la Light Convolutional Neural Network (LCNN) 2.2.2. Come riportano gli esperimenti di Muteb Aljasem, Arun Kumar Singh e Stefano Borzì, tra le soluzioni proposte troviamo anche soluzioni non basate sulle Neural Network.
Aljasem propone una soluzione basata su SVM con Asymmetric Bagging e sotto campionamento del dataset, vengono creati più classificatori SVM e ognuno di essi è allenato con un sample casuale del dataset e un sample casuale delle feature. Il classificatore integra i risultati di più classificatori SVM applicando la regola di voto normalizzato e ponderato (wNVR), esso ha ottenuto in seguito ai test di verifica un EER pari a 5,22%. Singh invece utilizza una specializzazione di SVM, chiamata Quadratic SVM. Questo modello è in grado di trovare dei separatori quadratici, quindi non per forza degli iperpiani. Questa è una buona soluzione qualora sia molto difficile o addirittura impossibile trovare dei buoni iperpiani separatori. Come metrica di verifica è stata utilizzata l'Accuracy ed è stato ottenuto in seguito alla verifica del modello un valore pari a 96,1%. Negli esperimenti di Stefano Borzì viene allenato un modello AdaBoost 3.3.8 dopo una preliminare estrazione delle feature con le librerie Spafe [53] e Pydub [54], il modello ha ottenuto un EER pari a 5%. Tra le soluzioni proposte vi sono anche degli ibridi tra vari modelli, Joao Monteiro propone un modello ibrido tra le LCNN 2.2.2 e le ResNet 2.2.3, il modello ha ottenuto in seguito alla sua verifica un EER pari a 6,38%. L'obiettivo è ottenere un modello con i vantaggi di entrambe le architetture, in sintesi il vantaggio della LCNN è che essa è molto performante per quanto riguarda la scelta delle feature discriminanti, invece il vantaggio delle ResNet è che, grazie alle skip connection, esse riescono ad evitare eventuali layer dannosi per la classificazione lungo il cammino di un dato. Un secondo modello ibrido è LCG RNN proposto da Alejandro Gomez-Alanis, il modello combina le LCNN e le Recurrent Neural Network (RNN)

ed è riuscito a ottenere un EER pari a 6,28%. Il vantaggio delle RNN è la presenza della memoria che permette un comportamento idoneo per trattare dati sequenziali (ad esempio le parole in un testo in linguaggio naturale e i campioni in un'onda di un file audio).

# Capitolo 2

# Metodi usati

## 2.1 Dataset e metodologie

Come accennato in Sezione 1.1, ASVspoof dal 2015 mette a disposizione ogni due anni dei dataset per delle challenge riguardo alla deepfake audio detection. Il dataset della challenge ASVspoof 2021 DF è una collezione di audio contenenti voci reali e voci sintetiche verosimili alle voci umane, processati con differenti codifiche di tipo lossy, quest'ultime sono le comuni codifiche utilizzate per la memorizzazione di file audio, come mp3. I dati sono codificati e poi decodificati, poiché la codifica è di tipo lossy l'audio ottenuto dopo la decodifica non sarà uguale all'originale, il processo di codifica e decodifica introduce delle distorsioni nei file audio. Il dataset è stato creato prendendo dati dal set ASVspoof LA evaluation e da altre fonti, il risultato è stato un dataset i cui audio deepfake sono stati generati usando più di 100 differenti algoritmi di spoofing [50]. Questa grande varietà di tecniche di generazione di audio deepfake è stata molto utile per rendere il dataset, e di conseguenza un possibile modello di classificazione costruito su di esso, più universale.
Il dataset è composto da 611.829 file audio in formato flac (per uno spazio del disco occupato pari a circa 34,5GB), ogni file audio ha un'etichetta, essa indica la natura del file audio. Se l'etichetta è Spoof significa che il file audio contiene una voce sintetica, se invece è Bonafide significa che il file contiene una voce reale. Il 96,3%(589.212) dei file audio ha etichetta Spoof, mentre

il restante 3,7%(22.617) ha etichetta Bonafide, ovviamente questa situazione non bilanciata è dovuta al fatto che è molto più veloce ottenere un file audio con una voce deepfake che un file audio con una voce reale, bisognerebbe avere a disposizione un gran numero di candidati pronti a registrare molti file audio. Il dataset 2021 DF è stato costruito come un set per l'evaluation, infatti la challenge prevede di creare un modello usando i dati del set ASVspoof 2019 per il training per poi valutarlo tramite il dataset di ASVspoof 2021.

## 2.2 Metodi usati per la deepfake audio detection in ASVspoof 2021

I metodi utilizzati maggiormente per la deepfake detection, soprattutto per ASVspoof 2021, sono basati sulle reti neurali.
È possibile distinguere due approcci principali, usare le classiche reti neurali che prendono in input feature numeriche, dove è necessario uno step di pre-processing per estrarre le feature dai dati del dataset, e utilizzare le Deep Neural Network le quali accettano come input i dati grezzi senza la necessità di estrarre specifiche feature numeriche da esso durante uno step di pre-processing.
Tra le feature maggiormente utilizzate troviamo constant-Q cepstral coefficients (CQCC), linear frequency cepstral coefficients (LFCC), mel frequency cepstral coefficients (MFCC) e constant-Q transform (CQT).
Gli organizzatori stessi di ASVspoof propongono quattro possibili tecniche, di cui le prime due non basate sulle reti neurali, per la deepfake detection.

- Modello GMM basato sulle feature CQCC (EER=25,25%)
- Modello GMM basato sulle feature LFCC (EER=25,56%)
- Modello LCNN basato sulle feature LFCC (EER=23,48%)
- Modello RawNet2 (deep learning) (EER=22,38%)

Di queste quattro tecniche, si evince dai risultati che la migliore è stata la tecnica basata su RawNet2 la quale ha ottenuto un EER pari a 22,38%

### 2.2.1 GMM

Il Gaussian Mixture Model, indicato solitamente con l'acronimo GMM, è un modello probabilistico di tipo unsupervised learning, solitamente utilizzato per applicare il clustering su una popolazione di dati.
Inoltre, il modello GMM viene utilizzato per la classificazione, infatti, funziona molto bene qualora i dati assumano una distribuzione simile a una Gaussiana. ASVspoof 2021 propone due modelli basati su GMM, essi si basano rispettivamente sulle caratteristiche CQCC e LFCC. Il modello proposto prevede l'allenamento di due GMM con 512 componenti su dati con etichetta Bonafide (audio con voce vera) e Spoof (audio con voce sintetica) [50][55]. L'algoritmo di training è Expectation-Maximisation (EM) con dei parametri casuali come configurazione iniziale. Viene utilizzato per trovare i valori dei parametri ottimali che garantiscono la likelihood massima di un modello statistico nei casi in cui siano coinvolte variabili latenti e i dati siano mancanti o incompleti. L'algoritmo è un processo iterativo che ad ogni iterazione esegue due step principali ovvero l'Expectation Step e il Maximization step.

Algoritmo EM

1: Expectation step: stimare i valori dei dati mancanti utilizzando i dati osservati disponibili del set di dati.
2: Maximization step: utilizzare i dati completi generati dopo l'Expectation step per aggiornare i parametri.
3: Ripetere le due operazioni precedenti finché la likelihood non converge.

Dopo aver allenato il modello si procede al testing tramite una formula, quest'ultima è utilizzata anche durante il processo di classificazione, essa calcola lo score relativo a un'osservazione del set di testing, lo score coincide con il log-likelihood ratio.

$$\Lambda(X) = logL(X|\Theta_n) - logL(X|\Theta_s) \tag{2.1}$$

Dove

- $X$ denota il vettore delle feature dell'osservazione
- $L$ denota la funzione di likelihood
- $\Theta_n$ denota la GMM allenata con i dati con etichetta Bonafide
- $\Theta_s$ denota la GMM allenata con i dati con etichetta Spoof

### 2.2.2 LCNN

La Light Convolutional Neural Network è una rete neurale che si presta molto bene a questo task di classificazione. Come riporta 'STC Antispoofing Systems for the ASVspoof2019 Challenge' [56], le caratteristiche principali di questa rete neurale sono

- Funzione di attivazione Max-Feature-Map (MFM)
- Funzione di loss A-softmax
- Inizializzione di Kaiming per i pesi della rete
- Step di batch normalization dopo i MaxPooling layer per garantire stabilità e maggiore velocità di convergenza nel processo di training
- Dropout pari a 0.75 per evitare overfitting

Le reti neurali che utilizzano la funzione di attivazione MFM riescono molto bene a scegliere le feature utili per la discriminazione delle osservazioni [57]. La funzione A-softmax è un'ottima funzione di loss in quanto costringe le feature ad essere discriminanti. La funzione può essere scritta come segue:

$$\mathrm{L}_{ang} = \frac{1}{N}\sum_i -log(\frac{e^{||\mathrm{x}_i||cos(m\Theta_{i,y_i})}}{e^{||\mathrm{x}_i||cos(m\Theta_{i,y_i})} + \sum_{i\neq y_i} e^{||\mathrm{x}_i||cos(m\Theta_{i,y_i})}}) \tag{2.2}$$

Dove

- $N$ indica il numero osservazioni del training set e m è un numero intero che controlla la dimensione dell'angular margin tra le classi
- $\Theta_{i,y_i}$ è l'angolo compreso tra l'osservazione i-esima $x_i$ e la corrispondente colonna $y_i$ dei pesi $W$ del layer di classificazione completamente connesso

Si riporta la struttura dettagliata della rete neurale Light CNN, Tabella 2.1.

**Tabella 2.1:** Architettura della rete neurale CNN [56]

| Type | Filter/Stride | Output | Params |
|---|---|---|---|
| Conv_1 | 5 × 5 / 1 × 1 | 863 × 600 × 64 | 1.6K |
| MFM_2 | - | 864 × 600 × 32 | - |
| MaxPool_3 | 2 × 2 / 2 × 2 | 431 × 300 × 32 | - |
| Conv_4 | 1 × 1 / 1 × 1 | 431 × 300 × 64 | 2.1K |
| MFM_5 | 1 × 1 / 1 × 1 | 431 × 300 × 32 | - |
| BatchNorm_6 | - | 431 × 300 × 32 | - |
| Conv_7 | 3 × 3 / 1 × 1 | 431 × 300 × 96 | 27.7K |
| MFM_8 | - | 431 × 300 × 48 | - |
| MaxPool_9 | 2 × 2 / 2 × 2 | 215 × 150 × 48 | - |
| BatchNorm_10 | - | 215 × 150 × 48 | - |
| Conv_11 | 1 × 1 / 1 × 1 | 215 × 150 × 96 | 4.7K |
| MFM_12 | - | 215 × 150 × 48 | - |
| BatchNorm_13 | - | 215 × 150 × 96 | - |
| Conv_14 | 3 × 3 / 1 × 1 | 215 × 150 × 128 | 55.4K |
| MFM_15 | - | 215 × 150 × 64 | - |
| MaxPool_16 | 2 × 2 / 2 × 2 | 107 × 75 × 64 | - |
| Conv_17 | 1 × 1 / 1 × 1 | 107 × 75 × 128 | 8.3K |
| MFM_18 | - | 107 × 75 × 64 | - |
| BatchNorm_19 | - | 107 × 75 × 64 | - |
| Conv_20 | 3 × 3 / 1 × 1 | 107 × 75 × 64 | 36.9K |
| MFM_21 | - | 107 × 75 × 32 | - |
| BatchNorm_22 | - | 107 × 75 × 32 | - |
| Conv_23 | 1 × 1 / 1 × 1 | 107 × 75 × 64 | 2.1K |
| MFM_24 | - | 107 × 75 × 32 | - |
| BatchNorm_25 | - | 107 × 75 × 32 | - |
| Conv_26 | 3 × 3 / 1 × 1 | 107 × 75 × 64 | 18.5K |
| MFM_27 | - | 107 × 75 × 32 | - |
| MaxPool_28 | 2 × 2 / 2 × 2 | 53 × 37 × 32 | - |
| FC_29 | - | 160 | 10.2MM |
| MFM_30 | - | 80 | - |
| BatchNorm_31 | - | 80 | - |
| **FC_32** | - | **2** | **64** |
| **Total** | - | - | **371K** |

### 2.2.3 RawNet2

I creatori della challenge ASVspoof 2021 propongono un modello basato su RawNet2, ovvero una Deep Neural Network, come possibile soluzione per la challenge DF.
RawNet2 elabora una forma d'onda data in input e restituisce un dato a due dimensioni, questo dato ci permette di acquisire informazioni riguardo la probabilità di appartenenza dell'audio alla classe (Bonafide o Spoof).
Si può vedere RawNet2 come un insieme di cinque sotto-reti messe insieme. Il primo layer è rappresentato da SincNet, questa rete convoluzionale interagisce in maniera diretta con la forma d'onda del file audio, essa consiste in un banco di filtri passa-banda parametrizzati.
Il secondo e il terzo layer sono rappresentati da due Residual Network , ovvero delle reti in cui vi sono dei collegamenti tra layer non adiacenti, un collegamento tra due layer non adiacenti viene detto skip connection. Un dato sotto opportune condizioni potrebbe quindi bypassare alcuni layer. Il vantaggio dell'aggiunta di questo tipo di connessione è che se un layer è dannoso per le prestazioni dell'architettura esso viene saltato dalla regolarizzazione. In questo modo, si ottiene l'addestramento di una rete neurale evitando quanto più possibile i problemi causati di esplosione o vanishing del gradiente. Si riporta un esempio di Residual Network in Figura 2.1.

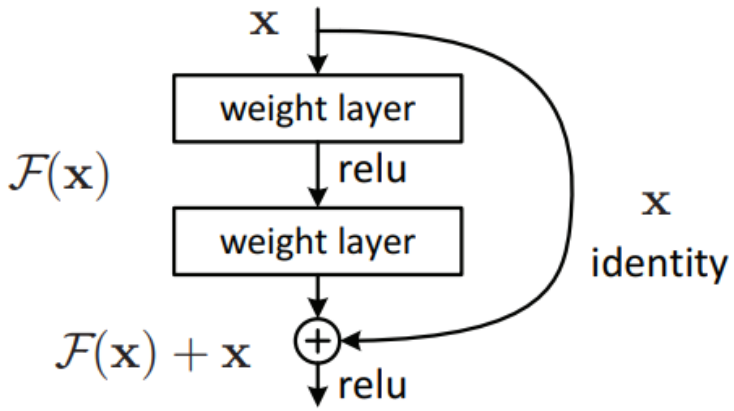


**Figura 2.1:** Esempio di Residual Network.

Il quarto layer è rappresentato da una Gated Recurrent unit (GRU), una unità specializzata usata nelle Recurrent Neural Network (RNN).
Le RNN sono delle particolari reti neurali dotate di "memoria" interna, il calcolo dell'output del dato N non dipende solamente dallo stesso dato ma anche dai precedenti input e output dei dati $0...N-1$. Le memorie utilizzate sono principalmente di due tipi, ovvero a breve termine (tengono conto di input processati poco tempo prima del nuovo input) e a lungo termine (tengono conto anche di input molto vecchi). Le RNN sono dunque adatte a lavorare con dati sequenziali, ovvero dati che hanno una dipendenza tra loro nel tempo come il testo e l'audio.
Le RNN soffrono il vanishing gradient problem, il gradiente durante il training potrebbe assumere un valore troppo piccolo rendendo un layer inallenabile. Se un layer non riuscisse ad apprendere, l'intera RNN si comporterebbe come se "dimenticasse" informazioni precedenti facendo svanire il vantaggio della memoria. Le GRU risolvono questo problema con l'utilizzo di due gate, chiamati update gate e reset gate. I gate sono delle unità speciali delle Neural Network che decidono, secondo un certo criterio, quali informazioni trasmettere in output. Questi due gate decidono dunque quali informazioni trasmettere in output al layer successivo e possono essere addestrati per memorizzare le informazioni più lontane nel tempo, questa caratteristica permette al gate di memorizzare solo informazioni rilevanti per fare delle previsioni migliori.

Si riporta in Tabella 2.2 l'architettura dettagliata di RawNet2 [58].

**Tabella 2.2:** Architettura della rete neurale RawNet2

| Layer | Input:64000 samples | Output shape |
|---|---|---|
| Fixed Sinc Filters | Conv(129,1,128)<br>MaxPooling(3)<br>BatchNormalisation<br>LeakyReLU | (21290,128) |
| Res block | $\left\{\begin{array}{c} BatchNormalisation \\ LeakyReLU \\ Conv(3,1,128) \\ BatchNormalisation \\ LeakyReLU \\ Conv(3,1,128) \\ MaxPooling(3) \\ FMS \end{array}\right\} \times 2$ | (2365,128) |
| Res block | $\left\{\begin{array}{c} BatchNormalisation \\ LeakyReLU \\ Conv(3,1,512) \\ BatchNormalisation \\ LeakyReLU \\ Conv(3,1,512) \\ MaxPooling(3) \\ FMS \end{array}\right\} \times 4$ | (29,512) |
| GRU | GRU(1024) | (1024) |
| FC | 1024 | (1024) |
| **Output** | **1024** | **2** |

## 2.3 Altri metodi

Oltre ai quattro modelli proposti dai creatori della challenge ASVspoof 2021, si riportano altri tre modelli illustrati in *UR Channel-Robust Synthetic Speech Detection System for ASVspoof 2021* [51] e *Speech is Silver, Silence is Golden:*

*What do ASVspoof-trained Models Really Learn?*[52].
Trai tre modelli uno di essi è basato sull'architettura RawNet2 illustrata in precedenza, gli altri due modelli sono ECAPA-TDNN e FCNN.

### 2.3.1 ECAPA-TDNN

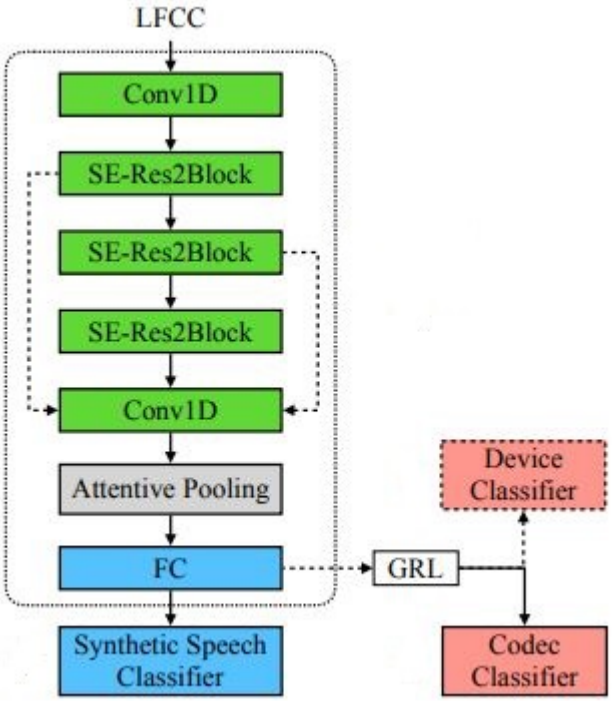


**Figura 2.2:** Architettura di ECAPA-TDNN.

Illustrato in *UR Channel-Robust Synthetic Speech Detection System for ASVspoof 2021.*
ECAPA-TDNN viene presentata come una versione migliorata di Time Delay Neural Network (TDNN) [59]. Nelle TDNN tutte le unità di un layer ottengono input dalle unità del layer precedente. Nel caso la TDNN lavori con segnali che variano nel tempo, ogni unità è connessa all'output delle unità precedenti e riceverà in input degli output trasmessi di proposito in ritardo dalle stesse unità. Le TDNN classificano molto bene segnali che variano nel tempo, come il suono.
ECAPA-TDNN migliora l'architettura per questo caso d'uso introducendo

layer di tipo Squeeze-Excitation (SE) e layer di tipo Res2Net che migliorano le performance e riducono il numero di parametri introducendo una serie di convoluzioni dei dati. Nella rete sono presenti anche connessioni tra layer non adiacenti che permettono la features aggregation.
Questo modello è in grado anche di predire il metodo di augmentation applicata nel dataset. Si riporta in figura 2.2 la struttura di ECAPA-TDNN [51]. ECAPA-TDNN addestrato sulle feature LFCC estratte da ASVspoof 2019, come riportato da *UR Channel-Robust Synthetic Speech Detection System for ASVspoof 2021* ha ottenuto un EER pari a 20,33% in seguito al test su ASVspoof 2021 DF.

### 2.3.2 FCNN

Illustrato in *Speech is Silver, Silence is Golden: What do ASVspoof-trained Models Really Learn?*. La Fully Connected Neural Network (FCNN), come si può intuire dal nome, è una rete neurale completamente connessa, ovvero una rete dove tutti i nodi di un layer sono connessi a tutti i nodi del layer successivo, si mostra un esempio di FCNN in Figura 2.3.

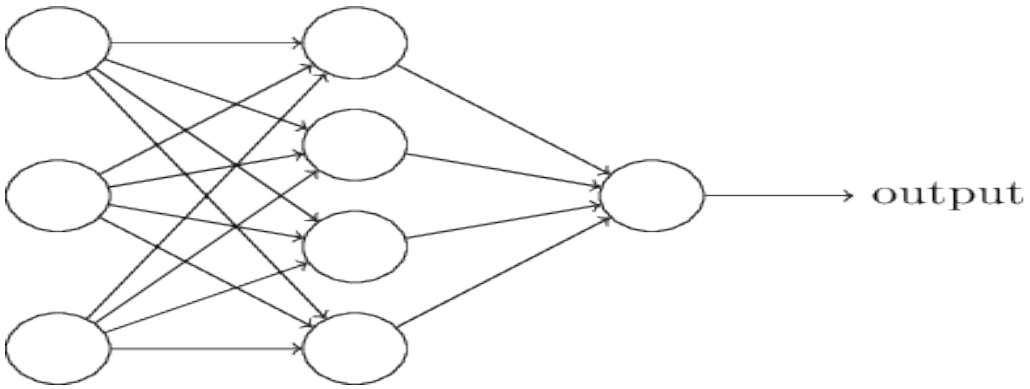


**Figura 2.3:** Esempio di Fully Connected Neural Network.

Nicolas Müller [52] utilizza una FCNN con due hidden layer di 128 nodi ciascuno, funzione di attivazione ReLU, valore di Dropout pari a 10% e un solo nodo di output con funzione di attivazione sigmoide. La feature utilizzata per la classificazione è la durata del silenzio in secondi presente nel file

audio. Lo scopo del suo esperimento non è ottenere un ottimo sistema di antispoofing, lo scopo è scoprire se la semplice durata del silenzio in secondi nel file audio è una feature discriminante. Se la durata del silenzio non fosse una caratteristica utile per la distinzione tra file audio con voce reale e con voce sintetica, l'EER del classificatore dovrebbe essere pari a (circa) 50%. L'intuizione si è rivelata corretta in quanto il classificatore ha ottenuto un EER pari a 17,42% (media aritmetica di quattro EER ottenuti con quattro test distinti). Si può quindi affermare che la **durata del silenzio è una caratteristica significativa** per la deepfake audio detection.

### 2.3.3 RawNet2 (esperimenti di Nicolas Müller et al.)

L'esperimento svolto utilizzando una FCNN è servito solamente per trovare una possibile caratteristica utile per costruire un buon modello di classificazione, Nicolas Müller propone poi un possibile miglioramento per la soluzione, basata sulla rete RawNet2, proposta dai creatori di ASVspoof 2021.
RawNet2 per design applica una feature subselection di 4 secondi, in pratica utilizza solamente 4 secondi del file audio in input.
La parte selezionata della forma d'onda del file audio probabilmente contiene meno silenzio rispetto alla forma d'onda completa, verrebbero dunque prese in considerazione meno feature relative al silenzio presente nel file audio. Si sceglie quindi di non applicare la feature subselection e utilizzare la completa forma d'onda di ogni file audio. Grazie a questa piccola modifica si è ottenuto un EER pari a 6,28% contro il 22,38% ottenuto applicando la feature subselection. Come fa intuire il titolo della pubblicazione, *Speech is Silver, Silence is Golden: What do ASVspoof-trained Models Really Learn?*, il silenzio all'interno del file audio si rivela molto utile per la deepfake audio detection.

## 2.4 Metodo proposto

Al contrario di quanto svolto per la challenge ASVspoof 2021 DF, ovvero l'allenamento di un modello usando il dataset ASVspoof 2019 e la valutazione

usando il dataset 2021, si propone di utilizzare solamente il dataset ASVspoof 2021 DF. Dunque dal dataset dovranno essere estratti il training set e il test set. Il lavoro è stato suddiviso in tre principali fasi ovvero l'estrazione delle feature, la visualizzazione delle feature e l'allenamento e il test del modello. L'estrazione delle feature viene effettuata con la libreria Spafe [53] di Python, ad eccezione del bitrate il quale è ottenuto utilizzando la libreria Mutagen. La visualizzazione delle feature consiste nella creazione di grafici che facciano capire quali feature sono importanti per la discriminazione. La creazione del modello consiste nell'allenamento e testing di vari modelli presenti nella libreria Scikit-learn di Python. Si propone questo metodo per la creazione di un modello di classificazione in grado di distinguere file audio contenenti voci reali da file audio contenenti voci sintetiche con l'obiettivo di costruire un modello non basato sulle reti neurali. Lo scopo è dunque evitare l'utilizzo delle reti neurali in quanto, sebbene esse siano molto performanti, sono molto difficili da allenare e sono poco efficienti quando si deve classificare un nuovo dato sconosciuto rispetto ai modelli di Machine Learning non basati sulle reti neurali.

# Capitolo 3

# Esperimenti e Risultati

## 3.1 Estrazione delle caratteristiche

Come anticipato in Sezione 2.4, la prima fase del metodo proposto consiste nell'estrazione delle caratteristiche dei file audio del dataset ASVspoof2021 DF. Tramite le funzioni e i metodi offerti dalla libreria Spafe sono state estratte le seguenti feature: bfcc, lfcc, lpc,lpcc,mfcc, imfcc, msrcc, ngcc, psrcc, plp, rplp, mel_filter_banks, bark_filter_banks, gammatone_filter_banks, spectrum, mean_frequency, peak_frequency, frequencies_std, amplitudes_cum_sum, mode_frequency, median_frequency, frequencies_q25, frequencies_q75, iqr, freqs_skewness, freqs_kurtosis, spectral_entropy, spectral_flatness, spectral_centroid, spectral_bandwidth, spectral_spread, spectral_rolloff, energy, rms, zcr, spectral_mean, spectral_rms, spectral_std, spectral_variance, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange, modindex, bit_rate. Lo script si occupa di estrarre le feature per ogni file audio e salvare i risultati in un file csv. In quanto i file sono approssimativamente 600.000, durante lo sviluppo degli script Python è stata utilizzata la libreria Multiprocessing al fine di parallelizzare in 4 processi l'estrazione del dataset analogamente suddiviso in 4 parti. I risultati della parallelizzazione sono stati ben visibili: circa 96 ore di computazione senza parallelizzazione contro circa 48 ore di computazione con parallelizzazione.
Note sull'estrazione delle feature:

- Alcune feature non sono semplici numeri, sono degli nd-array (esempio: bfcc) o strutture simili come tuple e liste, si è optato per applicare la media per ottenere un solo valore numerico.

- Eventuali feature composte da numeri complessi, sono state opportunamente convertite in float tramite la funzione abs() di Python.

## 3.2 Visualizzazione delle caratteristiche

Tramite la libreria MatplotLib di Python e alcune funzionalità della libreria NumPy sono stati creati dei grafici per ottenere una completa visualizzazione delle distribuzioni dei valori delle feature e per capire quindi quali feature possano essere importanti nella discriminazione dei file audio. L'obiettivo è individuare le feature che assumono un valore significativo in base all'etichetta associata al file audio. Ogni grafico è composto da due istogrammi convertiti in curve per migliorare la visualizzazione dei dati, i grafici rappresentano rispettivamente i valori di una specifica feature per le etichette Bonafide e Spoof. Sull'asse orizzontale si trovano i valori assunti dalla feature, mentre sull'asse verticale si trova il numero di occorrenze in percentuale di ogni valore. Nel grafico è presente inoltre una legenda che indica il numero di occorrenze per ogni etichetta. La creazione degli istogrammi prevede la conoscenza a priori del numero di bin, per il calcolo dell'opportuno valore dei bin sono state usate due tecniche:

- Regola di Freedman-Diaconis [60]

- Algoritmo Knuth per il numero ottimale di bins [61]

I risultati sono molto simili ma l'algoritmo di Knuth ha una complessità computazionale molto più alta rispetto alla regola di Freedman-Diaconis. Si è dunque deciso di creare i grafici utilizzando queste due tecniche, l'algoritmo di Knuth viene però eseguito utilizzando un campione di 10.000 osservazioni, 5000 per ogni etichetta. Tra le caratteristiche osservate solo cinque di esse hanno riportato piccole differenze nella distribuzione. Si riportano di seguito i grafici relativi alle cinque caratteristiche, ovvero: bit_rate in Figura 3.1,

meanfun in Figura 3.2, meandom in Figura 3.3, mindom in Figura 3.4 e zcr in Figura 3.5, con il numero di bin calcolato grazie alla regola di Freedman-Diaconis.

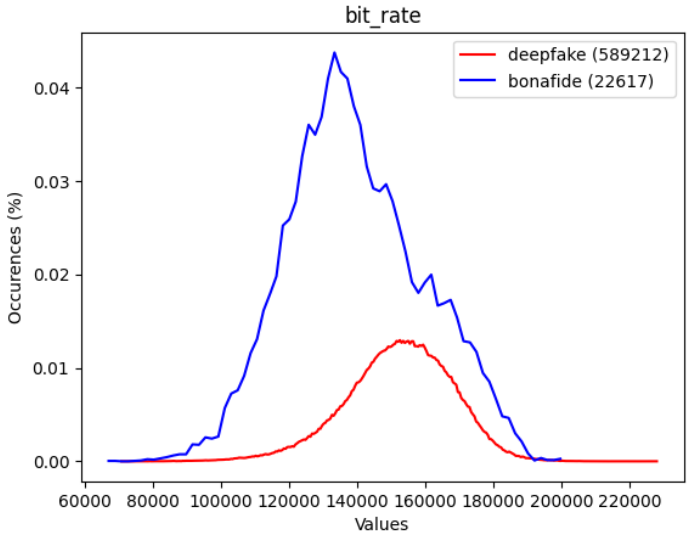


**Figura 3.1:** Istogrammi relativi alla feature bit_rate.

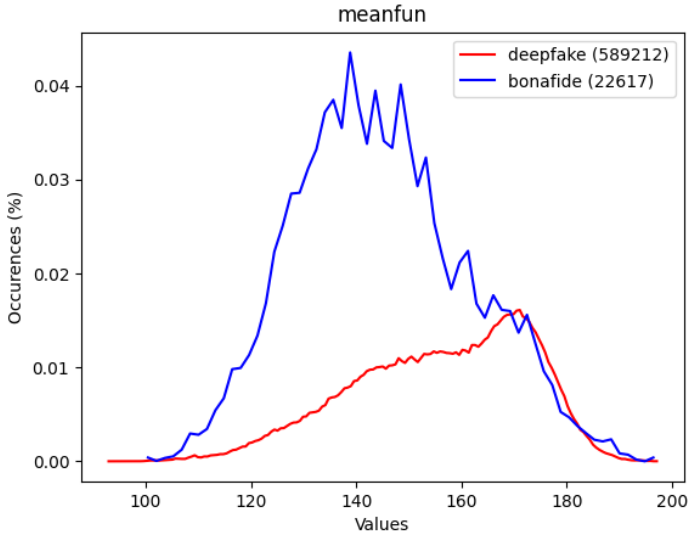


**Figura 3.2:** Istogrammi relativi alla feature meanfun.

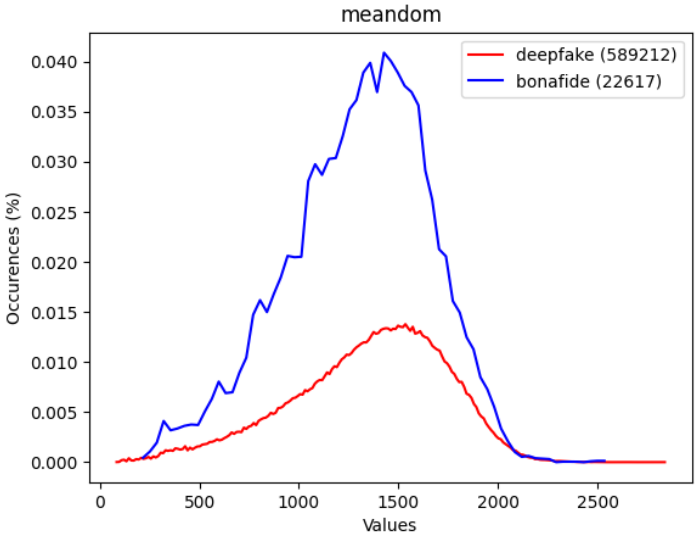


**Figura 3.3:** Istogrammi relativi alla feature meandom.

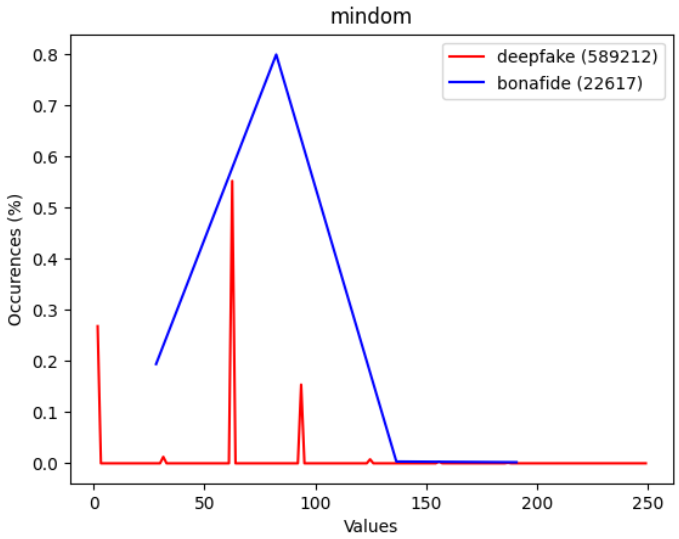


**Figura 3.4:** Istogrammi relativi alla feature mindom.

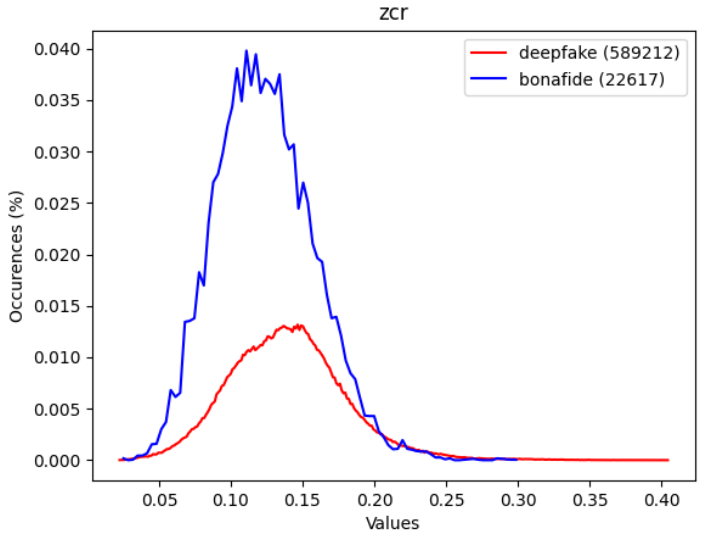


**Figura 3.5:** Istogrammi relativi alla feature zero crossing rate.

Le due distribuzioni per le altre feature invece non fanno risaltare nessuna differenza. Si riportano come esempi i grafici relativi alla feature bfcc in Figura 3.6 e dfrange in Figura 3.7.

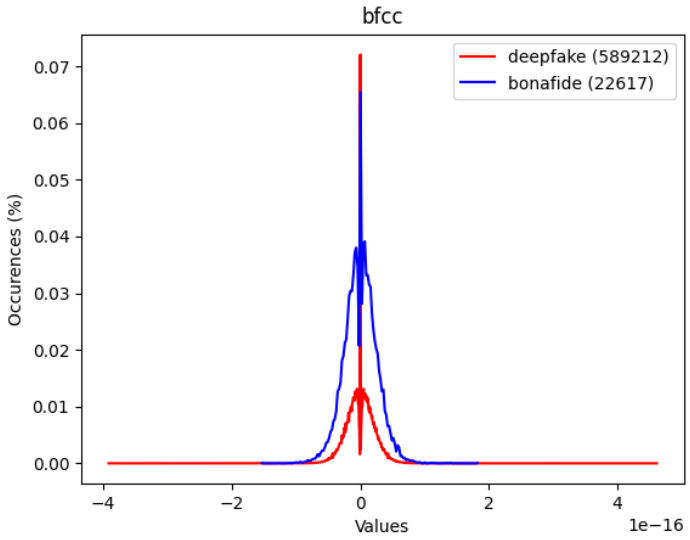


**Figura 3.6:** Istogrammi relativi alla feature bfcc.

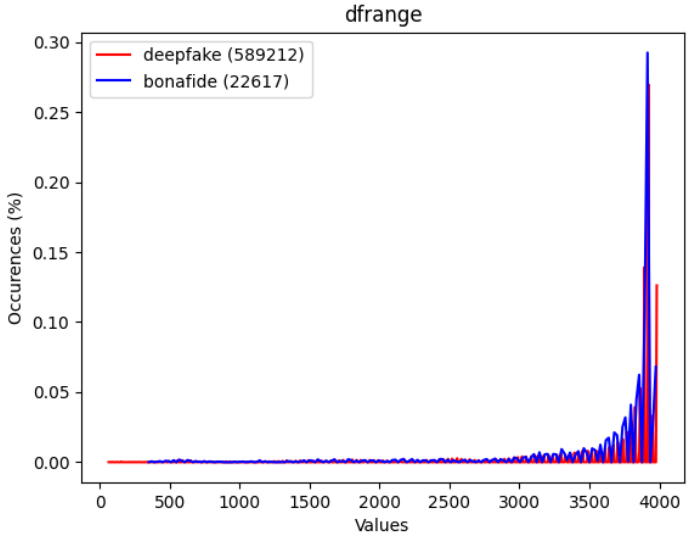


**Figura 3.7:** Istogrammi relativi alla feature dfrange.

## 3.3 Creazione del modello

L'ultima fase del lavoro consiste nella creazione del miglior modello di classificazione possibile. Basandosi sui risultati ottenuti mediante la visualizzazione delle caratteristiche, è possibile applicare una selezione delle feature più significative, in particolare durante questo studio verranno utilizzate solamente le cinque caratteristiche che rpesentano distribuzioni diverse.
La metrica principale con cui verranno giudicati i classificatori è l'Equal Error Rate (EER), calcolato con le seguenti formule:

$$FAR = \frac{FP}{(TP + FP + TN + FN)} \tag{3.1}$$

$$FRR = \frac{FN}{(TP + FP + TN + FN)} \tag{3.2}$$

$$EER = \frac{FAR + FRR}{2} \tag{3.3}$$

Dove FP indica i falsi positivi, TP i veri positivi, TN i veri negativi e FN i falsi negativi (la label considerata come positiva è Bonafide). FAR rappresenta il False Acceptance Rate e FRR rappresenta il False Rejection Rate [26]. Oltre all'EER viene calcolata anche l'Accuracy per class tramite le seguenti formule:

$$TPR = \frac{TP}{TP + FN} \tag{3.4}$$

$$TNR = \frac{TN}{TN + FP} \tag{3.5}$$

$$ACC\_per\_Class = \frac{(TPR + TNR)}{2} \tag{3.6}$$

Altre metriche utilizzate sono: Accuracy, Precision e Recall.

### 3.3.1 Classificatori di Scikit-Learn

La strategia applicata consiste nell'allenare e testare più modelli, i modelli usati sono: DecisionTreeClassifier, SVC, LogisticRegression, KNeighborsClassifier, LinearDiscriminantAnalysis, RandomForestClassifier, MLPClassifier, AdaBoostClassifier, GaussianNB, MultinomialNB e QuadraticDiscriminantAnalysis, tutti resi disponibili dalla libreria SciKit-Learn.

**Tabella 3.1:** Note sui parametri

| **Modello** | **Parametri** |
|---|---|
| DecisionTreeClassifier | Default |
| SVC | probability = True |
| LinearRegression | Default |
| KNeighbors Classifier | n_neighbors = 2 |
| LinearDiscriminant Analysis | Default |
| RandomForest Classifier | Default |
| MultiLayerPerceptron Classifier | Default |

| | |
|---|---|
| AdaBoost Classifier | Default |
| GaussianNaiveBayes | Default |
| MultinomialNaiveBayes | Default |
| QuadraticDiscriminant Analysis | Default |

Si precisa che per il modello Multinomial Naive Bayes viene applicato uno scaling dei dati prima dell'allenamento e del testing in quanto si è notato un leggero miglioramento delle metriche utilizzando dati scalati. Viene utilizzato l'oggetto MinMaxScaler presente nella libreria di Scikit-Learn.

Per ogni modello vengono eseguite dieci iterazioni, in ogni iterazione viene allenato e verificato un classificatore basato su quel modello. Ogni iterazione consiste nel flusso di lavoro illustrato di seguito.

Singola iterazione del processo di creazione di un modello

1: Estrazione di un sample **Data** di N osservazioni dal dataset
2: Suddivisione del sample **Data** in training set e test test
3: Allenare il classificatore utilizzando il training set
4: Verificare il classificatore calcolando le seguenti metriche: Accuracy, Accuracy per Class, Precision, Recall, Equal Error Rate.
5: Costruire la matrice di confusione
6: Restituire i valori delle metriche e la matrice di confusione

Dopo aver concluso le dieci iterazioni vengono ottenuti dieci classificatori, vengono conservati poi i file relativi al classificatore migliore, al classificatore peggiore e al classificatore più vicino alla media. La qualità di ogni modello è dettata dal valore dell'Equal Error Rate. Bisogna però fare attenzione, ci sono situazioni in cui l'Equal Error Rate potrebbe assumere valori molto bassi ma il modello associato potrebbe essere non adatto per una classificazione accurata, per questo motivo vengono calcolate anche altre metriche e si costruisce la matrice di confusione. La procedura appena illustrata viene applicata per ogni modello. Gli esperimenti principali sono stati due, ovvero la creazione del modello in una situazione non bilanciata a favore dei file audio

con etichetta Spoof e la creazione del modello con un sample reso bilanciato di proposito. Si mostrano le metriche ottenute con un sample non bilanciato in Tabella 3.2.

**Tabella 3.2:** Metriche ottenute con un sample non bilanciato. Si mostrano le metriche del modello più vicino alla media. I valori sono approssimati alla quarta cifra decimale.

| Modello | EER | Accuracy | Accuracy per class | Precision | Recall |
|---|---|---|---|---|---|
| DTC[1] | 0.0321 | 0.9358 | 0.5916 | 0.2222 | 0.216 |
| SVC[2] | 0.0183 | 0.9633 | 0.5 | 0.0 | 0.0 |
| LR[3] | 0.0189 | 0.9622 | 0.5 | 0.0 | 0.0 |
| KNC[4] | 0.0487 | 0.9027 | 0.5606 | 0.0875 | 0.1932 |
| LDA[5] | 0.0187 | 0.9627 | 0.5062 | 0.3 | 0.0136 |
| RFC[6] | 0.0183 | 0.9633 | 0.5221 | 0.3333 | 0.0476 |
| MLPC[7] | 0.0183 | 0.9633 | 0.5 | 0.0 | 0.0 |
| ABC[8] | 0.0182 | 0.9635 | 0.5 | 0.0 | 0.0 |
| GNB[9] | 0.0207 | 0.9587 | 0.522 | 0.1803 | 0.0526 |
| MNB[10] | 0.0184 | 0.9632 | 0.5 | 0.0 | 0.0 |
| QDA[11] | 0.0197 | 0.9607 | 0.5341 | 0.2963 | 0.0748 |

Considerando solo l'EER e l'Accuracy (chiamata anche Accuracy overall) si potrebbe pensare di aver ottenuto degli ottimi classificatori, le altre metriche fanno però notare qualcosa che non va. Il classificatore ottiene un EER molto basso e un'Accuracy molto alta perchè banalmente classifica la

[1]Decision Tree Classifier
[2]Support Vector Machine
[3]Logistic Regression
[4]KNeighbors Classifier
[5]Linear Discriminat Analysis
[6]Random Forest Classifier
[7]Multi Layer Perceptron Classifier
[8]AdaBoostClassifier
[9]Gaussian Naive Bayes
[10]Multinomial Naive Bayes
[11]Quadratic Discriminant Analysis

maggior parte delle osservazioni come Spoof e gli audio con etichetta Spoof sono circa il 97% del dataset. Il classificatore non ha appreso come riconoscere un file audio con etichetta Bonafide, per questo motivo è stato creato un classificatore allenandolo però con un sample bilanciato. Si riportano in Tabella 3.3 i valori delle metriche ottenute con un sample bilanciato.

**Tabella 3.3:** Metriche ottenute con un sample bilanciato. Si mostrano le metriche del modello più vicino alla media. I valori sono approssimati alla quarta cifra decimale.

| Modello | EER | Accuracy | Accuracy per class | Precision | Recall |
|---|---|---|---|---|---|
| DTC | 0.1578 | 0.6845 | 0.6845 | 0.6884 | 0.6799 |
| SVC | 0.1638 | 0.6725 | 0.6718 | 0.7507 | 0.5122 |
| LR | 0.2083 | 0.5833 | 0.5832 | 0.5802 | 0.5716 |
| KNC | 0.2054 | 0.5891 | 0.593 | 0.5558 | 0.8155 |
| LDA | 0.1618 | 0.6765 | 0.6766 | 0.6881 | 0.65 |
| **RFC** | **0.118** | **0.764** | **0.764** | **0.7744** | **0.7466** |
| MLPC | 0.2463 | 0.5075 | 0.5 | 0.0 | 0.0 |
| ABC | 0.1528 | 0.6945 | 0.695 | 0.717 | 0.656 |
| GNB | 0.1593 | 0.6813 | 0.6812 | 0.7002 | 0.632 |
| MNB | 0.1957 | 0.6085 | 0.6086 | 0.5979 | 0.6592 |
| QDA | 0.1534 | 0.6932 | 0.6929 | 0.7253 | 0.6184 |

L'EER è aumentato e l'Accuracy overall è diminuita ma le altre metriche adesso assumono valori più stabili. Fa eccezione MLPClassifier che sembra comportarsi come un Random Classifier. Il miglior modello è **Random Forest Classifier** con i seguenti valori:

- EER = 0.118
- Accuracy = 0.764
- Accuracy per class = 0.764
- Precision = 0.7744

- Recall = 0.7466

Si riporta la matrice di confusione relativa a Random Forest Classifier in Figura 3.8.

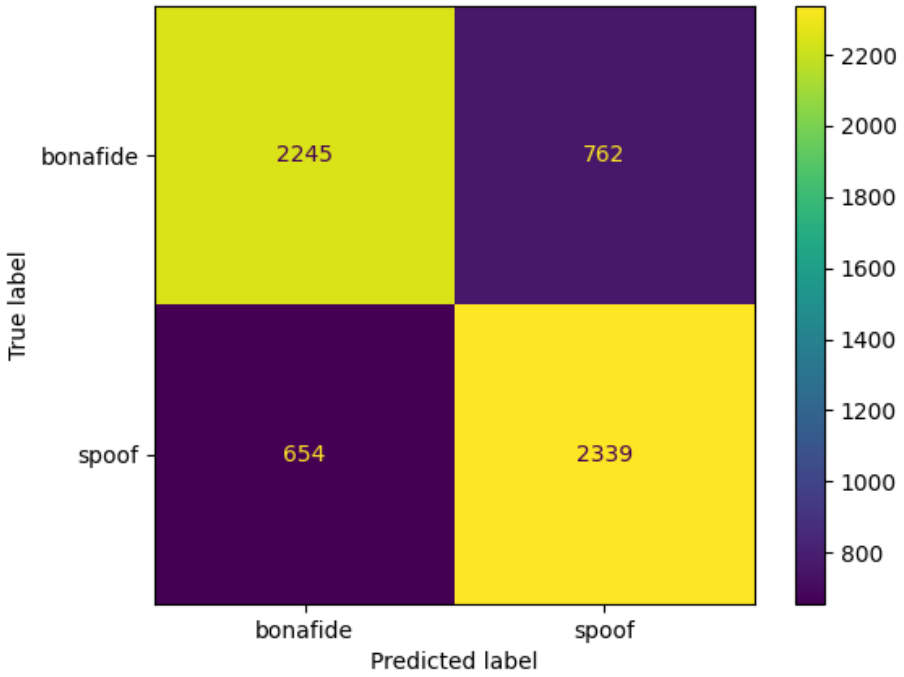


**Figura 3.8:** Matrice di confusione di RFC.

### 3.3.2 Decision Tree Classifier

I Decision Tree Classifier, o alberi decisionali in Italiano, sono dei modelli di classificazione supervisionati. Come si capisce dal nome, questi modelli sono rappresentabili come degli alberi, in ogni nodo vi è un test su una variabile, in base all'esito del test viene deciso in quale nodo figlio proseguire. Alla fine di ogni percorso vi è una foglia, ogni foglia rappresenta un'etichetta da assegnare all'osservazione. Si riporta un esempio di albero decisionale in Figura 3.9.

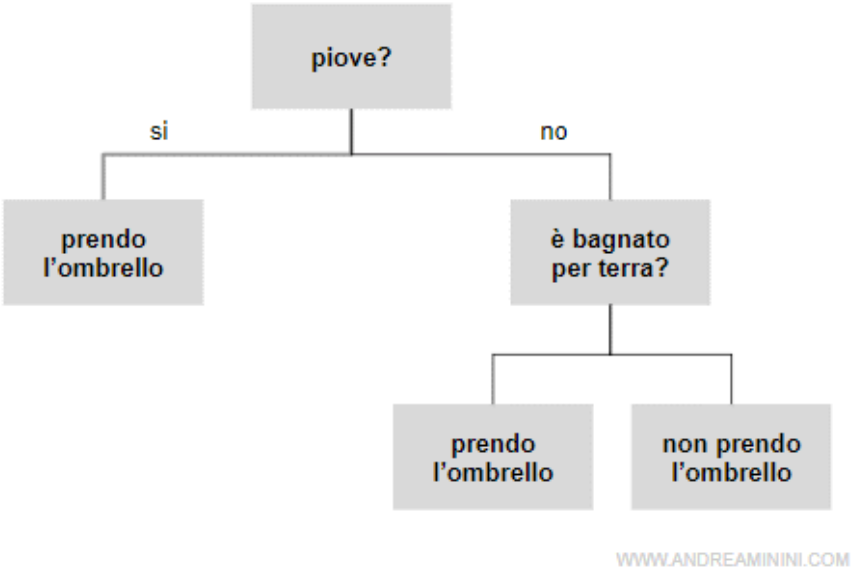


**Figura 3.9:** Esempio di albero decisionale.

Vi sono due tipologie di test, essi dipendono dalla natura della variabile che si prende in considerazione, se essa è discreta il test consiste semplicemente nel verificare quale degli N valori è assunto e proseguire il cammino decisionale nel corrispettivo figlio (tra N figli). Se la variabile è continua i figli del nodo saranno due, se l'osservazione ha variabile inferiore a una soglia il cammino prosegue in un figlio, altrimenti prosegue nel secondo figlio. La soglia utilizzata nel test è calcolata in maniera idonea durante la fase di allenamento.

### 3.3.3 Support Vector Machine

Support Vector Machine è un modello di apprendimento supervisionato utilizzato per creare classificatori lineari non probabilistici. Tramite poi alcune tecniche ed espansioni è possibile utilizzare SVM anche per la classificazione non lineare e non binaria. Per creare il classificatore vengono mappate su uno spazio geometrico le osservazioni del dataset, si prevede che queste osservazioni si trovino in zone diverse dello spazio geometrico in base alla loro etichetta. Vengono calcolati tre iperpiani paralleli (nel caso vi siano solo due caratteristiche numeriche si parla di rette), ovvero uno per ogni etichet-

ta più un iperpiano equidistante dagli altri due detto iperpiano separatore. La classificazione di una osservazione sconosciuta si basa sulla distanza dagli iperpiani delle due etichette. Viene assegnata l'etichetta corrispondente all'iperpiano più vicino. Il modello è chiamato Support Vector Machine perchè gli iperpiani sono calcolati in base ai vettori di supporto, ovvero le osservazioni del training set che si trovano più vicine alla zona dello spazio dove si concentrano le osservazioni della diversa etichetta, i vettori di supporto definiscono il "margine" della zona dove si concentrano le osservazioni con una certa etichetta. Si riporta un esempio relativo a un classificatore binario lineare in Figura 3.10.

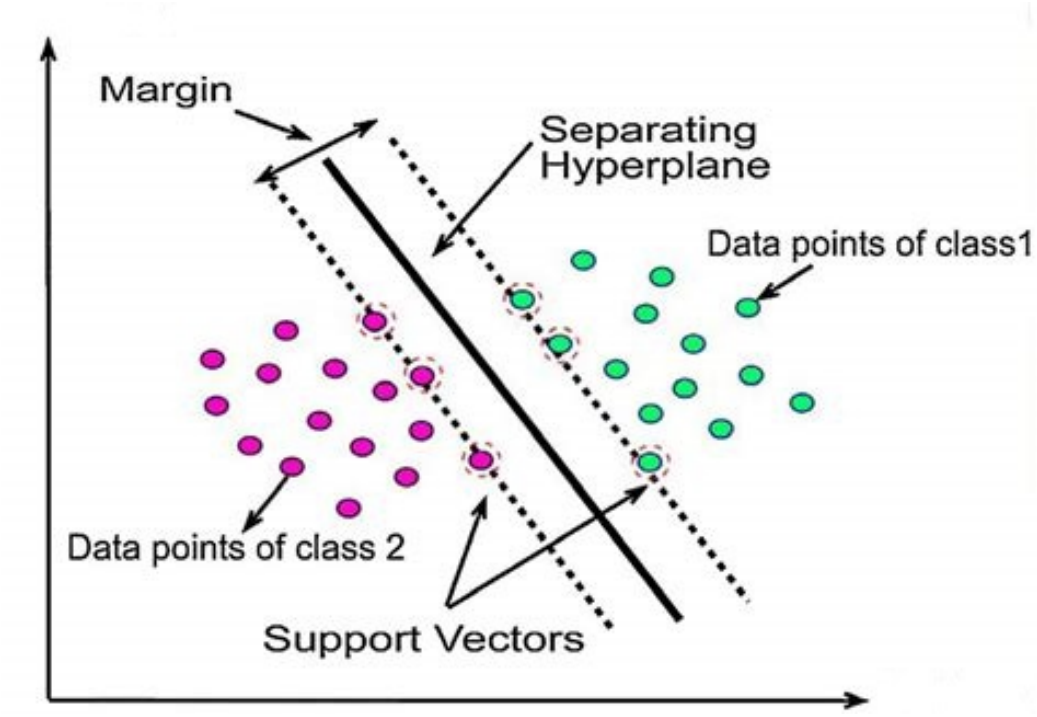


**Figura 3.10:** Esempio di vettori di supporto e relativo margine in SVM.

### 3.3.4 Logistic Regression

La regressione logistica è un algoritmo di apprendimento supervisionato che costruisce un modello probabilistico lineare di classificazione dei dati. Il modello di classificazione basato sulla regressione logistica nasce dal desiderio di

applicare la regressione per la classificazione. Il modello sfrutta la funzione logit, una funzione sigmoide, essa viene utilizzata poiché molti fenomeni naturali assumono un comportamento riportabile a questa curva, utilizzare una retta calcolata con un algoritmo di regressione lineare non otterrebbe buoni risultati. Inoltre un modello di regressione lineare restituisce valori in tutto $\mathbb{R}$, mentre per un classificatore binario l'intervallo d'interesse è [0,1], il valore trovato in questo intervallo denota la probabilità che l'osservazione appartenga a una delle due classi. Quello che viene fatto è costruire una funzione che permetta di mappare $\mathbb{R}$ in [0,1] e successivamente assegnare un'etichetta se il valore è inferiore a 0,5 o la seconda etichetta in caso contrario. La probabilità $p$ può essere calcolata come segue:

$$p = \frac{1}{1 + e^{-logit(p)}} \tag{3.7}$$

$logit(p)$ assume un comportamento lineare, è quindi approssimabile usando la regressione lineare.

$$logit(p) = \Theta_0 x_0 + \Theta_1 x_1 + \Theta_2 x_2 + ... + \Theta_n x_n \tag{3.8}$$

$\Theta_i$ indica il coefficiente della caratteristica numerica $i$ dell'osservazione, il training consiste nel ricercare i valori ideali da assegnare a questi coefficienti. Si costruisce una funzione costo e si sfrutta l'algoritmo di discesa del gradiente per trovare i migliori valori per i coefficienti. Si riporta in Figura 3.11 un grafico con una sigmoide utilizzata per la regressione logistica nel caso in cui vi sia una sola caratteristica numerica, messa a confronto con il risultato ottenuto con una regressione lineare.

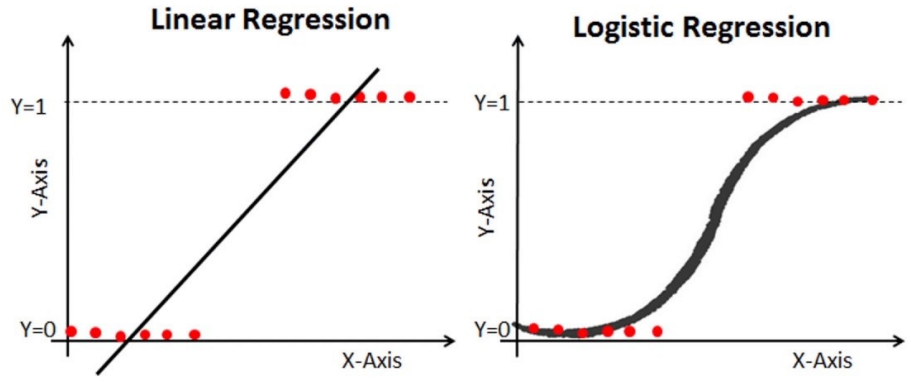


**Figura 3.11:** Curva sigmoide a confronto con una retta per risolvere il problema di classificazione.

Dal confronto riportato si nota come la curva sigmoide si avvicina meglio al comportamento naturale della feature. A primo impatto si potrebbe pensare di utilizzare una funzione scalino invece di una sigmoide, non si usa la funzione scalino in quanto essa non è derivabile e quindi non si potrebbe applicare l'algoritmo di discesa del gradiente.

### 3.3.5 KNeighbors Classifier

KNeighbors Classifier, detto anche K-Nearest Neighbors è un modello utilizzabile per costruire un classificatore basato sulla distribuzione delle osservazioni del dataset su uno spazio geometrico. È un classificatore detto 'lazy' in quanto non vi è un vero e proprio addestramento del modello fino alla classificazione di un'osservazione sconosciuta. Per classificare un'osservazione essa viene proiettata nello spazio geometrico, vengono prese in considerazione le K osservazioni più vicine, verrà assegnata l'etichetta di maggioranza. Per questo motivo nel caso di classificazione binaria, K è solitamente un numero dispari, in questo modo si evitano situazioni in cui non vi è un'etichetta di maggioranza. Alcune versioni più sofisticate di KNN non si limitano ad assegnare l'etichetta di maggioranza, piuttosto assegnano un peso ad ogni osservazione tra le K prese in considerazione, in pratica l'osservazione più

vicina avrà un peso maggiore mentre l'osservazione più lontana avrà un peso inferiore. La debolezza di questo classificatore risiede nella scelta del parametro K, un valore troppo basso potrebbe rendere il classificatore troppo influenzabile dal rumore, un valore troppo alto creerebbe una situazione di overfitting.

### 3.3.6 Linear Discriminant Analysis e Quadratic Discriminant Analysis

Linear Discriminant Analysis è un metodo utilizzato nell'apprendimento automatico per trovare una combinazione lineare di caratteristiche in grado di separare due o più classi. La combinazione lineare può quindi essere usata per costruire un classificatore lineare. LDA prima di costruire il classificatore lineare vero e proprio si occupa di massimizzare la distanza tra le medie delle due distribuzioni e minimizzare la varianza all'interno di ogni distribuzione, questi risultati sono ottenibili grazie a tecniche di features reduction. Grazie a queste due operazioni il classificatore lineare che verrà creato sarà più performante. LDA fallisce quando le medie delle due distribuzioni sono molto simili. Il concetto dietro a Quadratic Discriminant Analysis è molto simile a LDA, esso è maggiormente idoneo rispetto ad LDA quando non è possibile suddividere I dati dello spazio geometrico con un iperpiano in maniera performante. La differenza tra LDA e QDA consiste nel diverso trattamento dato alle matrici di covarianza relative alle distribuzioni dei dati, entrambi i modelli assumono che i dati per ogni classe siano distribuiti secondo una Normale, ma LDA assume anche che le matrici di covarianza delle distribuzioni delle varie classi coincidano, invece QDA permette che le matrici di covarianza non coincidano. Quadratic Discriminant Analysis è in grado di apprendere dei margini non lineari, è quindi più flessibile, soffre però problemi di efficienza in quanto il numero di parametri dipende in maniera quadratica dal numero di caratteristiche, molte feature significherebbero un scarsa efficienza dell'algoritmo. Si riporta in Figura 3.12 un esempio sul comportamento assunto da LDA e QDA.

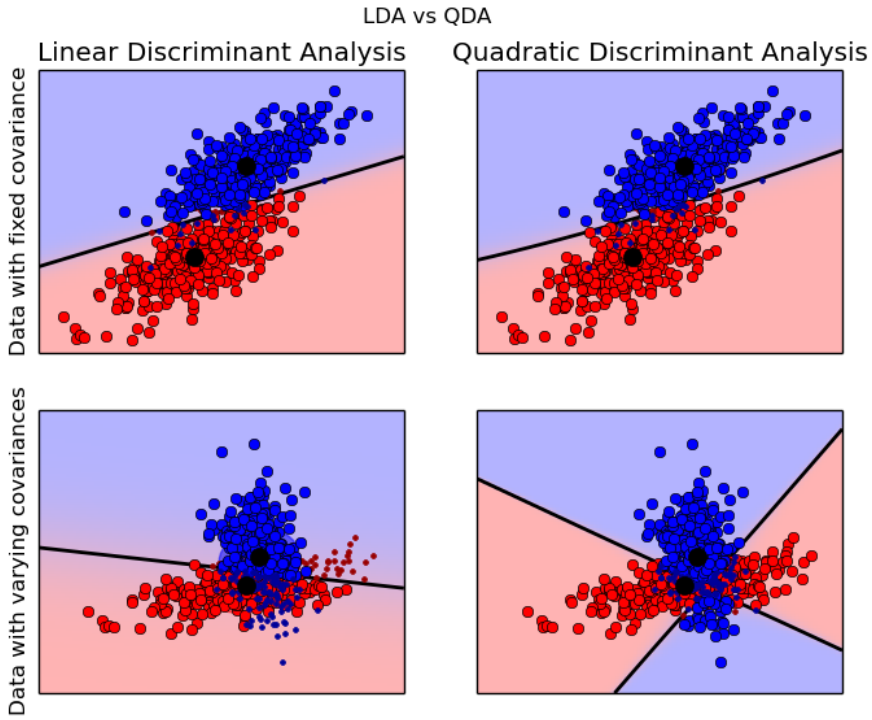


**Figura 3.12:** Confronto tra LDA e QDA, si nota come QDA si comporti meglio qualora sia molto difficile dividere i dati con un iperpiano.

### 3.3.7 Random Forest Classifier

Il Random Forest è un modello di classificazione d'insieme, esso è dunque costituito da più classificatori basati sugli alberi decisionali e sulla base delle decisioni prese da tutti gli alberi verrà poi data in output un'unica etichetta, ad esempio potrebbe essere ritornata l'etichetta di maggioranza. Ogni albero viene addestrato usando un sottoinsieme casuale del dataset e un sotto insieme casuale degli attributi dell'osservazione, in questo modo si evita che gli alberi siano correlati tra loro. Random Forest è in genere più accurato dei semplici alberi decisionali e tramite il campionamento casuale del dataset e degli attributi evitano overfitting.

### 3.3.8 AdaBoost Classifier

AdaBoost Classifier è un modello facente parte dei classificatori d'insieme che utilizzano la tecnica di Boosting. Questi classificatori hanno l'obiettivo di costruire un classificatore "forte" partendo da più classificatori "deboli" che vengono allenati in sequenza. In pratica viene allenato il primo modello debole, dopo ciò viene allenato un secondo modello che ha l'obiettivo di capire cosa ha sbagliato il primo modello e correggere i suoi errori e così via fino all'n-esimo classificatore. Questo processo iterativo continua finché non sono soddisfatte delle performance specifiche o fino a un numero massimo di iterazioni. La creazione del classificatore AdaBoost può essere riassunta col seguente pseudocodice.

Allenamento del classificatore AdaBoost

1: Assegnazione di un peso uguale a ciascuna osservazione del dataset
2: Fornire il dataset preso in input al modello e identificare le osservazioni classificate in modo errato dal modello precedente (se c'è)
3: Aumentare il peso delle osservazioni classificate in modo errato
4: Se i risultati sono quelli desiderati finire l'esecuzione dell'algoritmo e ritornare il modello, altrimenti tornare al passo 2

### 3.3.9 Gaussian e Multinomial Naive Bayes

I classificatori Naive Bayes sono una famiglia di modelli di apprendimento probabilistico basati sul teorema di Bayes e sull'assunzione che le caratteristiche dell'osservazione siano condizionalmente indipendenti tra loro. Si riporta il teorema di Bayes

$$P(H_C|X) = \frac{P(X|H_C) \times P(H_C)}{P(X)} \tag{3.9}$$

Dove $H_C$ indica l'ipotesi che l'osservazione $X$ sia di classe $C$. L'obiettivo è stabilire la classe C che massimizza $P(H_C|X)$, si può procedere non tenendo in considerazione il denominatore in quanto $P(X)$ è fissa e si comporta come

un fattore scala. $P(H_C)$ si può stimare dal dataset, ma è comunque comune assumere che le classi abbiano tutte la stessa probabilità.

Il calcolo di $P(X|H_C)$ viene semplificato dall'assunzione naive che tutte le variabili del vettore X siano condizionalmente indipendenti. Si può calcolare come segue

$$P(X|H_C) = \prod_{k=1}^{n} P(X_k|H_C) \tag{3.10}$$

Dove $X_k$ indica la variabile k-esima del vettore $X$. Il calcolo viene eseguito per ogni classe, la classe con la probabilità più alta sarà quella assegnata. Ciò che differenzia Gaussian Naive Bayes e Multinomial Naive Bayes è il fatto che nel primo modello si assume che le features si distribuiscono secondo una Normale mentre per il secondo modello si assume che esse si distribuiscono come una Multinomiale. In genere Gaussian Naive Bayes da migliori risultati in presenza di features continue, mentre Multinomial Naive Bayes funziona molto bene per features discrete.

### 3.3.10 Classificatori Naive

Oltre ai classici modelli offerti dalla libreria Scikit-Learn si è optato per costruire due classificatori naive basati sulla feature bit_rate. Il primo classificatore prima calcola il valore medio del bit_rate sia per gli audio Spoof sia per gli audio Bonafide, successivamente per classificare un'osservazione calcolerà la distanza del corrispondente bit_rate dalle due medie. Verrà assegnata la label corrispondente alla media più vicina. Ad esempio se le due medie fossero 120.000 per Spoof e 160.000 per Bonafide e l'osservazione avesse il valore della feature bit_rate pari a 150.000, la label assegnata sarebbe Bonafide. Per il secondo modello si cerca un valore idoneo per una soglia, la classificazione si baserà sul confronto tra il valore assunto dalla feature e il valore della soglia. In base all'esito del confronto sarà predetta una delle due etichette. La ricerca del valore ideale consiste nel testare 100 soglie e scegliere la migliore. Ad esempio se il valore minimo fosse 70.000 e il massimo fosse 210.000, si dovrebbe calcolare (210000 - 70000)/100 ovvero 1400. I seguenti valori verranno utilizzati come valore di soglia: 70.000, 71.400, 72.800, ...,

208.600, 210.000, la soglia che permetterà di ottenere l'EER più basso sarà considerata la soglia ideale. Se si volesse calcolare una soglia ancora più accurata basterebbe usare un numero più alto di 100 valori da testare.
La creazione del classificatore naive basato sui due valori di media segue un flusso di lavoro simile a quello usato per i modelli di Scikit Learn, le due medie vengono calcolate su tutto il dataset, vengono poi eseguite dieci iterazioni simili al procedimento 2, la differenza nelle due strategie è l'assenza del training nel classificatore naive, viene semplicemente classificato tutto il sample, si calcolano le metriche e si costruisce la matrice di confusione. Svolte le dieci iterazioni di classificazione vengono consevati I file relativi al classificatore migliore, al classificatore peggiore e al classificatore più vicino alla media. Per il classificatore naive basato sulla soglia non vengono effettuate le dieci iterazioni, poiché la migliore soglia viene già trovata con i 100 (o più in generale N) tentativi. Si riportano di seguito le tabelle con le metriche ottenute per i due classificatori.

**Tabella 3.4:** Sample non bilanciato

| Modello | EER | Accuracy | Accuracy per class | Precision | Recall |
|---|---|---|---|---|---|
| Naive_mean | 0.1594 | 0.6812 | 0.6786 | 0.0811 | 0.6758 |
| Naive_th | 0.0175 | 0.965 | 0.5012 | 0.1667 | 0.0029 |

**Tabella 3.5:** Sample bilanciato

| Modello | EER | Accuracy | Accuracy per class | Precision | Recall |
|---|---|---|---|---|---|
| Naive_mean | 0.1668 | 0.6664 | 0.6664 | 0.6731 | 0.6473 |
| Naive_th | 0.1628 | 0.6744 | 0.6744 | 0.7448 | 0.5308 |

Anche per i classificatori naive usare un sample bilanciato risulta essere una scelta migliore e nonostante la semplicità dietro le due strategie, i risultati ottenuti non sono molto lontani dai valori delle metriche relative ai classificatori di Scikit-Learn.

# Conclusioni

I modelli di classificazione binaria utilizzati per la deepfake audio detection sono spesso basati sulle Neural Network o sulle Deep Neural Network, questi modelli sono in genere molto affidabili per la classificazione ma sono molto difficili da allenare. Per un buon allenamento servono molti più dati e molto più tempo di computazione rispetto all'allenamento di un classico modello di machine learning, come ad esempio gli alberi decisionali. Gli esperimenti sono mirati a costruire un modello anti-spoofing nell'ambito dei file audio senza ricorrere all'utilizzo delle reti neurali. Per trovare un buon classificatore sono stati provati più modelli offerti dalla libreria Scikit-Learn. Random Forest Classifier, allenato e testato su un sample bilanciato del dataset, è risultato essere il miglior modello di classificazione. Si capisce da questi esperimenti l'importanza che bisogna dare all'estrazione e al trattamento delle caratteristiche, non è semplice nell'ambito dell'audio capire quali possano essere delle feature discriminanti per la classificazione. Per questi motivi i migliori modelli di deepfake audio detection sono le Deep Neural Network, esse interagiscono direttamente con il file audio e da esso capiscono quali sono le feature importanti da utilizzare per la discriminazione. Non per questo bisogna completamente ignorare le feature e lasciare tutto il lavoro alla reti, grazie infatti agli esperimenti riportati in *Speech is Silver, Silence is Golden: What do ASVspoof-trained Models Really Learn?*[52] si comprende l'importanza del silenzio per la discriminazione tra file audio con voce reale e file audio con voce sintetica, grazie alle intuizioni di Nicolas Müller et al. la rete RawNet2 ha subito una piccola quanto importante modifica che ha permesso di ottenere un EER pari a 6,28% contro il 22,38% ottenuto originariamente. La modifica consiste nel non eseguire la features subselection, essa fa in

modo che del file audio vengano usati solo 4 secondi, in questo modo gran parte del silenzio non verrebbe utilizzato nel training. Dagli esperimenti si evince un aspetto importante tanto quanto banale, ovvero l'attenzione da dare alla suddivisione del sample utilizzato per la creazione del classificatore. Durante gli esperimenti sono stati ottenuti risultati molto diversi in base al bilanciamento del sample utilizzato per l'allenamento e verifica del modello. Solamente il 3,7% delle osservazioni nel dataset ha come etichetta Bonafide, la percentuale di Bonafide in un sample estratto in maniera casuale sarebbe simile. Il modello non avrebbe a disposizione un numero sufficiente di osservazioni con etichetta Bonafide per apprendere come riconoscere un audio contenente una voce vera. Le metriche Accuracy per class, Precision e Recall si rivelano molto utili in quanto identificano una scarsa affidabilità del modello, nonostante quest'ultimo in seguito ai test di verifica abbia ottenuto un valore di EER e un valore di Accuracy overall rispettivamente molto basso e molto alto. È importante dunque attenzionare il numero di osservazioni per ogni classe che viene utilizzato per l'allenamento e la verifica di un modello. Dalle considerazioni espresse riguardo alle feature dei file audio e dai risultati ottenuti utilizzando il dataset ASVspoof 2021 in questi esperimenti e negli esperimenti documentati nelle referenze citate, si ipotizza che gli unici modelli in grado di ottenere risultati ottimali per la deepfake audio detection siano i classificatori basati sulle Neural Network, specialmente le Deep Neural Network.

# Bibliografia


[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Er finestra, 2000. `https://www.shouldiremoveit.com/Er-Finestra-29853-program.aspx`.

[3] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[4] Momina Masood, Mariam Nawaz, Khalid Mahmood Malik, Ali Javed, Aun Irtaza, and Hafiz Malik. Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward. *Applied Intelligence*, pages 1–53, 2022.

[5] Héctor Delgado, Nicholas Evans, Tomi Kinnunen, Kong Aik Lee, Xuechen Liu, Andreas Nautsch, Jose Patino, Md Sahidullah, Massimiliano Todisco, Xin Wang, Junichi Yamagishi. Asvspoof, 2021. `https://www.asvspoof.org/`, 2021.

[6] Yang Gao, Tyler Vuong, Mahsa Elyasi, Gaurav Bharaj, and Rita Singh. Generalized spoofing detection inspired from audio generation artifacts. *arXiv preprint arXiv:2104.04111*, 2021.

[7] Hartmut Traunmüller. Wolfgang von kempelen's speaking machine and its successors. *Wolfgang von Kempelen's speaking machine and its successors*, 1997.

[8] Mark Barton. Software Automatic Mouth. `https://yamm.finance/wiki/Software_Automatic_Mouth.html`, 1982.

[9] MacInTalk. `https://en.wikipedia.org/wiki/PlainTalk#Original_MacInTalk`.

[10] Windows Agent. `https://en.wikipedia.org/wiki/Microsoft_Agent`.

[11] Sciforce Text-to-Speech Synthesis: an Overview, 2020. `https://medium.com/sciforce/text-to-speech-synthesis-an-overview-641c18fcd35f`.

[12] Okko Räsänen Statistical parametric speech synthesis, 2020. `https://wiki.aalto.fi/display/ITSP/Statistical+parametric+speech+synthesis`.

[13] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[14] Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. In *International Conference on Machine Learning*, pages 195–204. PMLR, 2017.

[15] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. *Advances in neural information processing systems*, 30, 2017.

[16] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. *arXiv preprint arXiv:1710.07654*, 2017.

[17] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018.

[18] Yaniv Taigman, Lior Wolf, Adam Polyak, and Eliya Nachmani. Voiceloop: Voice fitting and synthesis via a phonological loop. *arXiv preprint arXiv:1707.06588*, 2017.

[19] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[20] Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. *Advances in neural information processing systems*, 31, 2018.

[21] Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31, 2018.

[22] Hieu-Thi Luong and Junichi Yamagishi. Nautilus: a versatile voice cloning system. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2967–2981, 2020.

[23] Yutian Chen, Yannis Assael, Brendan Shillingford, David Budden, Scott Reed, Heiga Zen, Quan Wang, Luis C Cobo, Andrew Trask, Ben Laurie, et al. Sample efficient adaptive text-to-speech. *arXiv preprint arXiv:1809.10460*, 2018.

[24] Jian Cong, Shan Yang, Lei Xie, Guoqiao Yu, and Guanglu Wan. Data efficient voice cloning from noisy samples with domain adversarial training. *arXiv preprint arXiv:2008.04265*, 2020.

[25] Nicoletta Boldrini Deep Learning, cos'è l'apprendimento profondo, come funziona e quali sono i casi di applicazione, 2021. `https://www.ai4business.it/intelligenza-artificiale/deep-learning/deep-learning-cose/#Le_reti_neurali_artificiali_la_base_del_Deep_Learning`.

[26] Stefano Borzì, Oliver Giudice, Filippo Stanco, and Dario Allegra. Is synthetic voice detection research going into the right direction? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 71–80, 2022.

[27] Parav Nagarsheth, Elie Khoury, Kailash Patil, and Matt Garland. Replay attack detection using dnn for channel discrimination. In *Interspeech*, pages 97–101, 2017.

[28] Tharshini Gunendradasan, Saad Irtza, Eliathamby Ambikairajah, and Julien Epps. Transmission line cochlear model based am-fm features for replay attack detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6136–6140. IEEE, 2019.

[29] Marcin Witkowski, Stanislaw Kacprzak, Piotr Zelasko, Konrad Kowalczyk, and Jakub Galka. Audio replay attack detection using high-frequency features. In *Interspeech*, pages 27–31, 2017.

[30] MS Saranya, R Padmanabhan, and Hema A Murthy. Replay attack detection in speaker verification using non-voiced segments and deci-

sion level feature switching. In *2018 international conference on signal processing and communications (SPCOM)*, pages 332–336. IEEE, 2018.

[31] Lian Huang and Chi-Man Pun. Audio replay spoof attack detection by joint segment-based linear filter bank feature extraction and attention-enhanced densenet-bilstm network. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1813–1825, 2020.

[32] Zhenzong Wu, Rohan Kumar Das, Jichen Yang, and Haizhou Li. Light convolutional neural network with feature genuinization for detection of synthetic speech attacks. *arXiv preprint arXiv:2009.09637*, 2020.

[33] Cheng-I Lai, Alberto Abad, Korin Richmond, Junichi Yamagishi, Najim Dehak, and Simon King. Attentive filtering networks for audio replay attack detection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6316–6320. IEEE, 2019.

[34] Xu Li, Na Li, Chao Weng, Xunying Liu, Dan Su, Dong Yu, and Helen Meng. Replay and synthetic speech detection with res2net architecture. In *ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6354–6358. IEEE, 2021.

[35] Jiangyan Yi, Ye Bai, Jianhua Tao, Zhengkun Tian, Chenglong Wang, Tao Wang, and Ruibo Fu. Half-truth: A partially fake audio detection dataset. *arXiv preprint arXiv:2104.03617*, 2021.

[36] Rohan Kumar Das, Jichen Yang, and Haizhou Li. Data augmentation with signal companding for detection of logical access attacks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6349–6353. IEEE, 2021.

[37] Muteb Aljasem, Aun Irtaza, Hafiz Malik, Noushin Saba, Ali Javed, Khalid Mahmood Malik, and Mohammad Meharmohammadi. Secure automatic speaker verification (sasv) system through sm-altp features and asymmetric bagging. *IEEE Transactions on Information Forensics and Security*, 16:3524–3537, 2021.

[38] Haoxin Ma, Jiangyan Yi, Jianhua Tao, Ye Bai, Zhengkun Tian, and Chenglong Wang. Continual learning for fake audio detection. *arXiv preprint arXiv:2104.07286*, 2021.

[39] Ehab A AlBadawy, Siwei Lyu, and Hany Farid. Detecting ai-synthesized speech using bispectral analysis. In *CVPR workshops*, pages 104–109, 2019.

[40] Arun Kumar Singh and Priyanka Singh. Detection of ai-synthesized speech using cepstral & bispectral statistics. In *2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 412–417. IEEE, 2021.

[41] PR Aravind, Usamath Nechiyil, Nandakumar Paramparambath, et al. Audio spoofing verification using deep convolutional neural networks by transfer learning. *arXiv preprint arXiv:2008.03464*, 2020.

[42] Joao Monteiro, Jahangir Alam, and Tiago H Falk. Generalized end-to-end detection of spoofing attacks to automatic speaker recognizers. *Computer Speech & Language*, 63:101096, 2020.

[43] Tianxiang Chen, Avrosh Kumar, Parav Nagarsheth, Ganesh Sivaraman, and Elie Khoury. Generalization of audio deepfake detection. In *Odyssey*, pages 132–137, 2020.

[44] Zhenyu Zhang, Xiaowei Yi, and Xianfeng Zhao. Fake speech detection using residual network with transformer encoder. In *Proceedings of the 2021 ACM workshop on information hiding and multimedia security*, pages 13–22, 2021.

[45] You Zhang, Fei Jiang, and Zhiyao Duan. One-class learning towards synthetic voice spoofing detection. *IEEE Signal Processing Letters*, 28:937–941, 2021.

[46] Alejandro Gomez-Alanis, Antonio M Peinado, Jose A Gonzalez, and Angel M Gomez. A light convolutional gru-rnn deep feature extractor

for asv spoofing detection. In *Proc. Interspeech*, volume 2019, pages 1068–1072, 2019.

[47] Guang Hua, Andrew Beng Jin Teoh, and Haijian Zhang. Towards end-to-end synthetic speech detection. *IEEE Signal Processing Letters*, 28:1265–1269, 2021.

[48] Ziyue Jiang, Hongcheng Zhu, Li Peng, Wenbing Ding, and Yanzhen Ren. Self-supervised spoofing audio detection scheme. In *INTERSPEECH*, pages 4223–4227, 2020.

[49] Run Wang, Felix Juefei-Xu, Yihao Huang, Qing Guo, Xiaofei Xie, Lei Ma, and Yang Liu. Deepsonar: Towards effective and robust detection of ai-synthesized fake voices. In *Proceedings of the 28th ACM international conference on multimedia*, pages 1207–1216, 2020.

[50] Junichi Yamagishi, Xin Wang, Massimiliano Todisco, Md Sahidullah, Jose Patino, Andreas Nautsch, Xuechen Liu, Kong Aik Lee, Tomi Kinnunen, Nicholas Evans, et al. Asvspoof 2021: accelerating progress in spoofed and deepfake speech detection. *arXiv preprint arXiv:2109.00537*, 2021.

[51] Xinhui Chen, You Zhang, Ge Zhu, and Zhiyao Duan. Ur channel-robust synthetic speech detection system for asvspoof 2021. *arXiv preprint arXiv:2107.12018*, 2021.

[52] Nicolas M Müller, Franziska Dieckmann, Pavel Czempin, Roman Canals, Konstantin Böttinger, and Jennifer Williams. Speech is silver, silence is golden: What do asvspoof-trained models really learn? *arXiv preprint arXiv:2106.12914*, 2021.

[53] Ayoub Malek. spafe/spafe: 0.1.2, April 2020.

[54] James Robert, Marc Webbie, et al. Pydub, 2018.

[55] Massimiliano Todisco, Héctor Delgado, and Nicholas Evans. Constant q cepstral coefficients: A spoofing countermeasure for automatic speaker verification. *Computer Speech & Language*, 45:516–535, 2017.

[56] Galina Lavrentyeva, Sergey Novoselov, Andzhukaev Tseren, Marina Volkova, Artem Gorlanov, and Alexandr Kozlov. Stc antispoofing systems for the asvspoof2019 challenge. *arXiv preprint arXiv:1904.05576*, 2019.

[57] Galina Lavrentyeva, Sergey Novoselov, Egor Malykh, Alexander Kozlov, Oleg Kudashev, and Vadim Shchemelinin. Audio replay attack detection with deep learning frameworks. In *Interspeech*, pages 82–86, 2017.

[58] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-end anti-spoofing with rawnet2. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373. IEEE, 2021.

[59] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[60] David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory, 1981.

[61] Kevin H. Knuth. Optimal data-based binning for histograms and histogram-based probability density models. *Digital Signal Processing*, 95:102581, 2019.

# Ringraziamenti

Ringrazio vivamente chi mi ha accompagnato durante questo mio percorso e non solo, ringrazio chi mi è stato accanto quindi tutta la mia famiglia e tutti i miei amici e compagni, di vita, di gioco, di scuola e di università.
Ringrazio ovviamente anche tutti i miei professori che, inaspettatamente, sono stati anche miei maestri di vita e ringrazio chi mi ha supportato per l'elaborazione della mia tesi di laurea.