Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Applicazione di algoritmi di Deep Learning alla Speech Emotion Recognition

Relatore:
Dott. Giovanni Delnevo

Presentata da:
Luca Bracchi

Sessione I
Anno Accademico 2022-2023

Introduzione

Il presente volume di tesi ha lo scopo di esporre i risultati del lavoro di ricerca svolto sulla Speech Emotion Recognition. Il problema è una specializzazione della Emotion Recognition su dati audio, ovvero riguarda l'identificazione delle emozioni espresse nelle tracce audio in esame ed è un compito che risulta di una complessità elevata sia per gli esseri umani sia per i sistemi automatici. Il grado di complessità è dovuto principalmente al fatto che soggetti differenti possono percepire emozioni diverse nella stessa traccia, mentre per quel che riguarda i sistemi automatici, che impiegano cioè il Machine Learning o, nel caso più specifico, il Deep Learning per effettuare la classificazione dei dati, le difficoltà hanno molteplici fonti differenti, descritte nel corso del documento.

Nel lavoro in esame la SER è stata affrontata impiegando quelle che ad oggi sono le più note tecniche di Deep Learning per la classificazione multi classe. Il campo di ricerca appena citato è una sotto branca del Machine Learning con il quale condivide la caratteristica principale di utilizzare algoritmi in grado di apprendere e di migliorare in autonomia. Tali algoritmi fanno uso di modelli profondi e più o meno complessi per etichettare i dati, rappresentati da un insieme di informazioni ritenute indicative rispetto alla classe di appartenenza della traccia audio. La profondità dei modelli è data dal numero di livelli interni della rete, detti livelli hidden, che sono in numero superiore all'unità. Lo scopo del lavoro è quello di valutare le principali tecnologie e metodologie nell'ambito del Deep Learning applicate al problema citato poc'anzi e per fare ciò sono stati allenati diversi modelli profondi tra

reti ricorrenti, convoluzionali e multi classificatori, adottando diverse tecniche per evitare i problemi classici nell'ambito del addestramento delle reti profonde, effettuando poi un'analisi dei dati raccolti riguardanti le prestazioni di tali modelli nella classificazione del dataset usato per l'allenamento.

Nel primo capitolo sarà fornita un'introduzione al contesto del problema, alla modellazione che ne è stata fatta, ai principali approcci di Deep Learning impiegati e alle principali opzioni disponibili in tema di dataset sul quale effettuare lo studio, di eventuali tecniche di data augmentation da impiegare, di tipologia e quantità di features da estrarre, di modelli da usare per classificare tali features e infine di metriche necessarie per condurre una valutazione accurata delle prestazioni, che si presti inoltre al paragone con altre soluzioni proposte in letteratura. Per lo stesso motivo verranno descritti anche gli accorgimenti da adottare quando si effettua la suddivisione del dataset nei tre sotto insiemi, training, validation e test set. Questa suddivisione è necessaria per ottenere risultati realistici. Verranno anche citati alcuni dei lavori effettuati sul tema della SER in ambito Machine Learning e Deep Learning, compresa TIM-Net, il modello convoluzionale che ad oggi è ritenuto essere lo stato dell'arte per il dataset selezionato.

Nel capitolo seguente si introdurrà l'esperimento, prima elencando le principali librerie usate e seguitando concentrandosi sui temi delle tecnologie e metodologie impiegate per analizzare il dataset, per estrarre le features dai dati, per effettuare la data augmentation incrementando così la dimensione del dataset, per costruire i modelli profondi usati come classificatori delle informazioni estratte, ed infine per allenare e valutare tali modelli. Verranno descritte le pratiche comuni e le principali categorie di modelli profondi da considerare quando si opera nel campo del Deep Learning.

Nel capitolo finale saranno forniti i risultati dell'esperimento descrivendo gli accorgimenti adottati per allenare le reti ricorrenti, le reti bidirezionali e quelle convoluzionali. Verrà illustrato il tentativo di costruzione di un multi classificatore per superare la qualità del miglior modello a disposizione. Infine si descriveranno i margini di miglioramento emersi dalle considerazioni

effettuate sull'esperimento eseguito e dal paragone fatto con la rete profonda TIM-Net.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione al contesto

In questo primo capitolo verranno fornite le informazioni necessarie a contestualizzare l'esperimento effettuato, descrivendo il problema affrontato, le soluzioni già presenti in letteratura, i dati usati e le lavorazioni che questi hanno subito.

Nella prima sezione verrà introdotto il problema della Speech Emotion Recognition (SER), la sua modellazione e alcune sue applicazioni nel mondo reale. Nella sezione seguente saranno descritti i principali approcci basati su Deep Learning, concentrandosi su quelli adoperati nel corso dell'esperimento descritto nel capitolo seguente. Sarà introdotto anche l'algoritmo della retro-propagazione dell'errore, fondamentale per l'allenamento di questo genere di modelli. Si discuterà poi il tema dei dataset disponibili per allenare i modelli sopracitati, esponendo una loro classificazione basata sulla natura dei dati che contengono e infine verrà puntualizzata l'importanza dell'esplorazione dei dati, mentre si rimanda al secondo capitolo per degli esempi pratici sullo specifico dominio del problema.

Successivamente verrà citata la data augmentation, tecnica atta ad incrementare il numero di dati disponibili: si partirà da un'introduzione generale per concludere citando alcuni lavori effettuati nel dominio della SER per aumentare i dati del training set. Nella quinta sezione si tratteranno le operazioni di raffinamento che generalmente subiscono i dati del dataset e le

informazioni che vengono comunemente estratte per essere poi date in input ai modelli. Saranno elencati inoltre i tipi di features usati nel corso dell'esperimento. Sarà poi fornita una descrizione di quello che è il processo di training di un modello di Machine Learning: partendo dalla suddivisione dei dati in training set, validation set e test set, citando l'importanza e le differenze di parametri e iperparametri e concludendo con la descrizione della cross validation, metodologia utilizzata per condurre il training del modello.

Nella sezione seguente si introdurrà il tema della valutazione delle prestazioni e delle metriche utilizzate per condurre tale studio. Saranno definite le metriche principali usate durante l'esperimento, quali loss, accuratezza e F1 Measure. Infine verranno citati alcuni dei lavori presenti in letteratura che hanno affrontato il problema della SER, partendo dagli algoritmi di Machine Learning tradizionali e concludendo con l'avvento del Deep Learning e l'uso che è stato fatto dei modelli profondi nel contesto del problema.

1.1 Introduzione alla Speech Emotion Recognition

Lo stato emotivo comporta svariate cambiamenti fisiologici nella persona, dalle pulsazioni alla pressione sanguigna, dalle onde cerebrali ai movimenti corporei. Alcuni cambiamenti, come quelli appena citati, richiedono oggi dispositivi medici di rilevazione portatili, necessitando del contatto di uno strumento al corpo del soggetto in esame. Altre componenti, come le espressioni facciali e i segnali audio possono essere rilevati in maniera meno invasiva. Per questo motivo gli studi nel settore del rilevamento automatico delle emozioni si sono concentrati sull'utilizzo di foto, video o segnali audio come fonte di informazioni per individuare le emozioni.

Il problema del Emotion Recognition riguarda l'individuazione automatica delle emozioni umane. Lo scopo della Speech Emotion Recognition, sottobranca del Emotion Recognition, è quello di individuare l'emozione espressa in un segnale audio basandosi su un insieme di informazioni, dette *features*,

estratte da tale segnale attraverso tecniche più o meno avanzate. La SER può essere vista come il parallelo sulle tracce audio della Sentiment Analysis, che mira invece a individuare sentimenti da dati testuali.

1.1.1 Modellazione del problema

La SER viene solitamente modellata come un problema di classificazione multiclasse, caratterizzato quindi da un dominio dell'output discreto. Tale dominio è rappresentato dall'insieme delle possibili emozioni rilevabili.

Durante lo svolgimento dell'esperimento è stato tentato anche un approccio orientato ai multiclassificatori. L'output di questi modelli è basato sull'aggregazione delle classificazioni fatte da più classificatori. L'idea è quella di raggiungere prestazioni migliori usando sottomodelli specializzati nel riconoscimento di emozioni specifiche. Sono stati usati classificatori che operavano in maniera uno-contro-tutti, restituendo una percentuale indice del grado di sicurezza dell'appartenenza del dato a una certa emozione, mentre l'output del multiclassificatore è stato definito scegliendo la classe che è stata etichettata con più sicurezza dal relativo sottomodello.

1.1.2 Ambiti applicativi

La SER fornisce oggi un contributo importante nel campo della Human-Computer Interaction [1] [2]. Tra i molteplici ambiti di utilizzo ci sono marketing, healthcare, customer satisfaction, stress monitoring e social media analysis, ma pur permettendo di carpire informazioni fondamentali per comprendere la conversazione e rispondere in maniera appropriata, fino a qualche anno fa esisteva un numero molto limitato di soluzioni generali al problema [3].

Nell'ambito dell'educazione a distanza è possibile individuare alunni o, più in generale, utenti annoiati, in modo tale da correggere lo stile di insegnamento e il materiale fornito [4]. Per quanto riguarda le automobili è stato dimostrato che le prestazioni alla guida sono influenzate dalle emozioni

provate dal guidatore [5]. La SER può dunque trovare un'applicazione in tale settore per promuovere l'esperienza di guida e migliorare le prestazioni del guidatore.

Nel campo della sicurezza i sistemi di rilevamento automatico delle emozioni possono fungere da supporto nei luoghi pubblici, permettendo di individuare emozioni estreme come ansia e paura. Parlando di comunicazioni, i call center che usano sistemi di risposta automatica possono beneficiare di sistemi di SER per capire le emozioni dell'interlocutore e adoperare tale informazione per migliorare il servizio offerto. Un esempio di applicazione di tali sistemi nel settore della sanità riguarda le persone affette da autismo [6]: sarà loro possibile utilizzare dispositivi portatili per rilevare le proprie emozioni, in modo da regolarle per adattare il comportamento sociale di conseguenza.

1.2 Deep Learning

Il Deep Learning è una sottobranca del Machine Learning, con il quale condivide la caratteristica principale, quella cioè di applicare algoritmi atti ad apprendere e ad auto-migliorarsi. A differenza del Machine Learning, nel Deep Learning vengono impiegate reti neurali profonde, ispirate alla struttura neuronale del cervello umano. La profondità di tali reti è data dal numero di livelli che compongono i modelli e che processano i dati a differenti livelli di astrazione, elaborando l'informazione in maniera sempre più completa.

I principali limiti imposti dal Deep Learning sono la necessità di dataset etichettati di grandi dimensioni [7], che richiedono una fase di etichettatura manuale ad opera di esperti, e l'elevata voracità di tali tecniche in termini di risorse computazionali [8]. Sono le moderne tecnologie che hanno recentemente abilitato un uso efficace di approcci basati su Deep Learning per la risoluzione di problemi complessi.

Studi più recenti sul tema della SER vertono sull'utilizzo di classificatori profondi: si tratta di reti neurali in grado di processare i descrittori estratti dai file audio e di classificare tali descrittori, restituendo in output l'emozio-

ne corrispondente. I modelli più comuni sono le reti neurali convoluzionali profonde (Convolutional Neural Network, DCNN) e le reti neurali ricorrenti (Recurrent Neural Network, RNN).

In questo contesto una rete neurale artificiale (ANN) è un sistema composto da neuroni artificiali connessi tra loro: queste connessioni permettono di trasmettere un segnale da un neurone all'altro. I neuroni sono aggregati in livelli o *layers*, dove il primo è detto livello di input, l'ultimo livello di output e i livelli interni sono detti hidden layers. Le reti neurali profonde (DNN) sono caratterizzate da un numero di livelli hidden superiore a 1 e la loro complessità è influenzata dalla profondità (numero di livelli hidden), dal numero di neuroni e dalle connessioni tra questi.

1.2.1 Allenamento di una rete neurale artificiale

Ogni input di ogni neurone artificiale è associato a un peso. Il neurone calcola il proprio output facendo prima la somma pesata degli input, sommando poi un proprio peso associato al neurone stesso, chiamato *bias*, e infine applicando al risultato ottenuto la funzione di attivazione. I pesi determinano quindi il contributo del relativo input nel calcolo dell'output del neurone e, più in generale, determinano il mapping dallo spazio degli input verso quello degli output. L'obiettivo dell'allenamento di una rete neurale artificiale consiste nel determinare il valore dei pesi che determina il mapping desiderato.

L'aggiornamento dei pesi, e di conseguenza l'addestramento di tali reti, è reso possibile dall'algoritmo di retropropagazione dell'errore tramite discesa stocastica del gradiente. Si tratta di un algoritmo iterativo, dove ogni iterazione è composta da due fasi, una di *forward* e una di *backward*. Nel passo di forward i dati vengono fatti passare attraverso la rete, dal primo livello fino all'ultimo: ogni livello calcola il proprio output e lo passa al livello successivo, fino a raggiungere l'ultimo livello, che restituirà l'output della rete. Con questo output viene calcolata la loss, attraverso la loss function, che ne determina la qualità rispetto al risultato atteso. Nel passo di backward

vengono aggiornati i pesi della rete in maniera tale da minimizzare il valore della loss, partendo dal livello di output e tornando indietro fino a quello di input.

Il nome dell'algoritmo deriva dal fatto che i pesi nel passo di backward vengono aggiornati di una quantità pari al prodotto tra il learning rate e il gradiente dell'errore rispetto al peso stesso. Il *learning rate* è uno degli iperparametri più importanti del Machine Learning: matematicamente permette di pesare il contributo del gradiente nell'aggiornamento dei pesi, mentre da un punto di vista più pratico determina quanto la rete impara dall'errore commesso al termine di ogni passo di training. Tarare il learning rate è un processo delicato, in quanto valori troppo bassi richiederanno un numero di passi di allenamento molto elevato e tra questi il comportamento della rete subirà modifiche minime, mentre valori troppo alti portano a cambi drastici del comportamento della rete tra i vari passi di addestramento, o addirittura alla divergenza, che comporta l'impossibilità di convergere.

Le numerose moltiplicazioni effettuate durante la retropropagazione possono causare due problemi differenti che rendono ingestibile l'addestramento di reti profonde. Il problema del vanishing gradient è dovuto ai valori troppo piccoli assunti dal gradiente: in questi casi non si sa in che direzione variare i pesi per ridurre la loss e, di conseguenza, migliorare la rete. Il problema del exploding gradient deriva invece dai valori troppo alti assunti dal gradiente, che portano a un modello instabile e incapace di imparare efficacemente, in quanto la rete ad ogni passo di training avrà un comportamento molto differente da quello del passo precedente.

Entrambi i problemi sono individuabili in fase di addestramento monitorando il valore della loss e quello dei pesi della rete. Se si verifica il vanishing gradient la loss subirà variazioni minime tra un passo di training e quello successivo oppure i pesi assumeranno valori molto piccoli, prossimi allo 0; gli indicatori relativi al exploding gradient sono invece un valore di loss che varia fortemente tra un passo di addestramento e l'altro e i valori dei pesi molto alti o con valori indefiniti (NaN). Nel secondo capitolo verranno descritte le

soluzioni adottate per mitigare questi due problemi.

1.2.2 Reti convoluzionali

Le CNN sono un tipo di reti neurali Feed-Forward(FFNN), che quindi non ammettono cicli al loro interno. La loro struttura è composta da due sottomodelli, una prima parte convoluzionale, alla quale è deputata la funzione di estrarre le features dall'input, e una parte completamente connessa, che si occupa invece di classificare le informazioni estratte dalla parte convoluzionale, producendo l'output del modello. La parte convoluzionale impiega livelli di pooling e convoluzione, mentre la parte completamente connessa è un percettrone multi-livello (MLP).

A differenza dei MLP, il sottomodello convoluzionale è localmente connesso, cioè i singoli neuroni non sono connessi a tutti i neuroni del layer precedente, ma solo a un loro sottoinsieme. Altra caratteristica fondamentale riguarda i pesi, che sono condivisi a livello di layer. Queste due caratteristiche, connessioni locali e pesi condivisi, permettono di ridurre il numero di parametri addestrabili, riducendo i tempi necessari per allenare la rete.

1.2.3 Reti ricorrenti

Le RNN sono reti neurali con cicli, ovvero dove sono presenti collegamenti dai neuroni di un livello verso neuroni dello stesso livello o di livelli precedenti. Nelle RNN vi è un concetto di cella, una parte della rete stessa dove è presente un'astrazione di memoria interna, detta hidden state, che permette di memorizzare informazioni sull'input attuale per riutilizzarle durante l'elaborazione degli input successivi. Questa caratteristica le rende particolarmente efficaci nell'analisi delle serie temporali, dove l'output della rete non dipende solo dall'input nello stesso istante, ma anche dagli input precedenti.

Per addestrare una RNN tramite l'algoritmo di retropropagazione dell'errore occorre effettuare l'*unfolding in time* della cella ricorrente, stabilendo a priori il numero di passi temporali sui quali effettuare l'analisi. La rete un-

folded rende più facile notare come questo tipo di modelli presentino una profondità intrinseca elevata ed è proprio tale profondità a renderli particolarmente soggetti ai problemi del vanishing ed exploding gradient in fase di training. Tra le soluzioni proposte per mitigare questo problema ci sono quelle del gradient clipping e del troncamento della retropropagazione: la prima soluzione consiste nel utilizzare una versione troncata del gradiente durante l'aggiornamento dei pesi, mentre la seconda prevede la suddivisione dell'input in sottosequenze. Entrambe le soluzioni introducono iperparametri che dovranno essere tarati.

In base agli input attesi e agli output restituiti è possibile classificare le RNN come segue.

- One-to-one, dove input e output hanno lunghezza fissa, corrispondono alle comuni FFNN.

- One-to-many, si tratta di reti che prendono input di lunghezza fissa, ma restituiscono una sequenza di dimensione variabile: è il caso delle RNN per l'image captioning [9] [10], ovvero modelli che prendono in input un'immagine per generarne una descrizione. La lunghezza di tale descrizione varia a seconda dell'immagine.

- Many-to-one, modelli che prendono una sequenza di dimensione variabile in input, ma restituiscono un output di dimensione fissa: esempi di reti di questo tipo sono le RNN per la sentiment analysis o il movie rating [11].

- Many-to-many, reti nelle quali input e output hanno dimensione variabile, a loro volta classificabili in base alla lunghezza dell'input rispetto a quella dell'output: questi possono avere la stessa lunghezza, come nel caso dei modelli ricorrenti per la classificazione dei singoli frame di un video, o lunghezze differenti, come nelle RNN per la Speech Recognition [12] [13].

Tra le formulazioni del problema della SER più popolari in letteratura si trovano quelle di classificazione statica e classificazione dinamica [14]: nel primo caso viene effettuata un'elaborazione dell'input, chiamata turn-based, dove una rete one-to-one predice l'emozione dalla traccia audio completa. Nel caso della modellazione dinamica l'elaborazione dell'input prende il nome di frame-based e la classificazione avviene a livello di frame tramite l'impiego di una rete many-to-one, che andrà poi ad aggregare le previsioni dei vari frames in modo da restituire come risultato un'unica emozione.

Una delle principali limitazioni delle RNN è quella della memoria a breve termine, in quanto l'hidden state permette di usare le informazioni dei soli input passati recenti. Un'architettura che permette di mitigare tale problema è quella delle Long Short-Term Memory (LSTM) che usano meccanismi interni, chiamati *gates*, per regolare il flusso delle informazioni da memorizzare, permettendo alla rete di scegliere quali informazioni mantenere e quali scartare.

Le reti neurali ricorrenti permettono, in ogni istante temporale, di analizzare la parte della sequenza che precede e che succede l'input attuale. Sono composte da due RNN comuni che processano la stessa sequenza in direzioni opposte: questo non introduce complessità, in quanto le reti vengono addestrate come le normali RNN, mentre l'output della rete bidirezionale è costruito aggregando i singoli output dei due sottomodelli.

1.3 Dataset

Le scelte fondamentali da valutare per affrontare il problema della SER sono riassunte dai punti seguenti e verranno approfondite in questo e nelle prossime sezioni:

- dataset appropriato sia in termini di cardinalità che di qualità dei dati,
- applicazione di eventuali tecniche di data augmentation,
- tipologia e numero di features da estrarre dai dati,

- modello da utilizzare per classificare le features

Sono già stati introdotti i limiti principali relativi ai dataset da impiegare per risolvere il problema in esame. Un ulteriore punto da tenere in considerazione è rappresentato dalla soggettività dell'emozione percepita: persone diverse possono associare emozioni diverse alla stesso traccia.

I dataset progettati per il riconoscimento delle emozioni sono classificabili in 3 categorie:

- Dataset naturali, composti raccogliendo estratti da programmi TV, da video YouTube e call center. Sono dataset di questo tipo RECOLA [15], CHEAVD [16], FAU-Aibo [17] e SUSAS [18]. L'utilizzo di questi dataset è complesso a causa della continuità delle emozioni, della variazione dinamica dell'emozione durante il discorso e di registrazioni non pulite, affette da rumore di fondo.

- Dataset seminaturali, creati da prestazioni di attori ai quali viene chiesto di leggere una scena contenente diverse emozioni. Esempi di dataset seminaturali sono IEMOCAP [19], AFEW [20] e Belfast [21]. Si tratta di emozioni create artificialmente, il che le rende più facilmente separabili, ma la varietà di tali emozioni è ridotta rispetto a quella raggiunta dai dataset simulati.

- Dataset simulati, creati con l'ausilio di relatori che leggono lo stesso messaggio con emozioni diverse, come è stato fatto per i dataset TESS, DES, EMO-DB [22] e RAVDESS [23]. La varietà di emozioni è molto ampia.

Dopo aver selezionato il dataset è fondamentale condurre una fase approfondita di esplorazione dei dati, impiegando tecniche statistiche e di visualizzazione per descrivere le caratteristiche dell'insieme, in maniera tale da capire dimensione, quantità e qualità dei dati a disposizione.

1.4 Data Augmentation

Quando si lavora con dataset di dimensione limitata si rivela estremamente utile adottare tecniche di data augmentation per raggiungere la quantità minima di dati necessaria per affrontare il problema in esame. Lo stesso discorso vale per i dataset sbilanciati, dove c'è una differenza elevata tra le cardinalità dei dati delle varie classi. Queste tecniche permettono di generare dati sintetici ex-novo o di produrre copie leggermente alterate di dati già esistenti, mitigando il problema della dimensione limitata o l'eventuale sbilanciamento.

L'allenamento di algoritmi di Machine Learning effettuato su dataset non sufficientemente ampi può portare al problema conosciuto come *overfitting*: il modello prodotto impara una funzione che descrive in maniera troppo precisa i dati di training, perché le informazioni in suo possesso non sono sufficienti per apprendere un comportamento più generale. Un modello del genere non sarà in grado di generalizzare, permetterà cioè di ottenere prestazioni ottime quando utilizzato su dati già incontrati in fase di addestramento, mentre ci sarà un netto peggioramento quando gli verranno sottoposti dati nuovi.

I modelli profondi, e quelli generativi in particolare, trovano impiego anche nella generazione di dati sintetici. É il caso, per esempio, delle Generative Adversarial Network(GAN), architetture composte da due sottomodelli, un generatore e un discriminatore. Il primo si occupa di produrre nuovi dati verosimili, mentre il secondo deve identificare i dati prodotti rispetto a quelli reali. L'addestramento di tali modelli è svolto in modo che il generatore impari a produrre dati sempre più simili a quelli reali, mentre il discriminatore deve diventare sempre più abile nell'identificare i dati falsi rispetto a quelli originali. Il modello generatore verrà poi utilizzato, una volta allenato, per produrre dati sintetici da aggiungere al dataset.

Per citare un esempio dalla letteratura riguardante il problema della SER, l'uso delle GAN si è rivelato più efficace rispetto alle tecniche di augmentation tradizionali nel bilanciamento dei dataset IEMOCAP e FEEL-25k [24]. Altri studi hanno dimostrato i limiti di queste architetture: i dati generati

hanno una distribuzione simile a quella dei dati del training set, usati per la generazione e questo fa sì che i dati sintetici si rivelino poco utili nella generazione di un modello capace di generalizzare [25], infatti le prestazioni saranno pessime se i dati del test set presentano una distribuzione differente da quella dei dati di training.

1.5 Ingegnerizzazione delle features

Le fasi di scelta ed esplorazione di un dataset adeguato sono seguite da un passo di ingegnerizzazione ed estrazione delle features dai file audio. Questo passaggio è reso fondamentale dalla necessità dei modelli di Machine Learning di ottenere in input informazioni che li rendano in grado di classificare efficacemente i file audio dai quali sono state estratte le features. Scegliere ed estrarre le informazioni giuste porterà il modello ad ottenere prestazioni migliori.

Tra la varietà di tipi di informazioni estraibili si è scelto di descrivere quelle che sono state impiegate durante lo svolgimento dell'esperimento, a sua volta descritto nel secondo capitolo del presente documento.

Le features seguenti sono tutte estratte partendo dallo spettrogramma della traccia audio, che consiste nella rappresentazione grafica dell'intensità del suono in funzione della frequenza e, di conseguenza, anche del tempo.

- Mel-Spectrogram features, ottenute convertendo lo spettrogramma nella scala Mel e campionando da tale grafico un certo numero di punti [26]. Sono ampiamente utilizzate nell'analisi dei suoni.

- Mel-Frequency Cepstral Coefficients (MFCC) features, le più usate nel contesto della SER [27]. Sono ottenute dal Mel Frequency Cepstrum (MFC), una rappresentazione dello spettro della potenza del suono.

- Chroma features, ottenute convertendo lo spettrogramma in un cromatogramma [28]; sono fortemente correlate alle classi di altezze e sono

particolarmente indicate per l'analisi musicale, permettendo di catturare in maniera efficace caratteristiche armoniche e melodiche di musica e strumenti.

- Tonnetz features, ottenute proiettando le Chroma features in uno spazio 6-dimensionale, dove ogni coppia di assi è usata per rappresentare un intervallo diverso: quinta giusta, terza minore e terza maggiore [29].

- Contrast features, rappresentano una stima dell'energia media del suono [30]. Un valore alto corrisponde a un segnale a banda stretta, mentre un valore basso corrisponde a un rumore a banda larga.

1.6 Algoritmo di addestramento

Come già accennato nelle sezioni precedenti, l'obiettivo che ci si pone quando si allena un classificatore è quello di produrre, mediante un algoritmo di addestramento, un modello capace di generalizzare, in grado quindi di raggiungere buone prestazioni anche sui restanti dati del dominio del problema, quelli non visti durante la fase di training.

A causa della complessità dei problemi per i quali vale la pena adottare soluzioni basate su Deep Learning, il dominio del problema ha una cardinalità infinita. Il dataset utilizzato rappresenta quindi un'approssimazione di tale dominio.

1.6.1 Training, validation e testing

Per addestrare il modello si opera una suddivisione del dataset in 3 sottoinsiemi disgiunti: il *training set*, il *validation set* e il *test set*.

Il training set è il sottoinsieme più numeroso ed è composto dai dati sui quali verrà allenato il modello. Durante la fase di training la rete impara le relazioni tra input e output, ovvero tra i dati del training set e le classi a loro associate. Se questo insieme è troppo limitato o composto da dati molto

simili tra loro, il modello prodotto non sarà in grado di generalizzare per via delle poche informazioni che ha potuto imparare dal training set.

Il validation set viene impiegato durante la fase di addestramento per valutare la bontà del modello in termini di prestazioni. Presentare alla rete dati non presenti nel training set permette di determinare se il modello si sta allenando in maniera efficace o se, a causa del overfitting, non è in grado di generalizzare sui nuovi dati. Un validation set limitato comporta un'elevata varianza delle metriche calcolate per valutare il modello, impedendo un efficace tuning degli iperparametri.

Infine, il test set viene utilizzato al termine dell'allenamento per valutare e visualizzare le prestazioni del modello. Sono le prestazioni sul test set che vengono usate per fare paragoni tra i diversi modelli che hanno affrontato lo stesso problema sullo stesso dataset.

È importante che la divisione tra i tre insiemi sia rispettata, cioè che siano effettivamente disgiunti, altrimenti la presenza nel test o nel validation set di dati già visti in fase di training farà sì che il modello riesca a classificare correttamente l'input, ma solo perché ha già imparato come etichettare quello specifico esempio. In questi casi il modello otterrà risultati positivi, ma la sua capacità di generalizzare sarà molto limitata. Le buone pratiche in materia di suddivisione del dataset suggeriscono di destinare l'80% dei dati per il training set e di dividere il restante 20% in parti uguali tra validation set e test set. Quella appena menzionata non è una regola, in quanto la suddivisione ottimale dipende da svariati fattori, alcuni dei quali sono il caso d'uso, la struttura del modello e la dimensione del dataset.

1.6.2 Parametri e iperparametri

Ruolo fondamentale quando si allena questo genere di modelli viene giocato dagli iperparametri. Mentre i parametri vengono imparati durante il training, gli iperparametri vengono valorizzati prima di eseguire l'algoritmo di addestramento.

La pratica prevede l'esecuzione di un algoritmo di training che produce un modello. Vengono poi valutate le prestazioni della rete ottenuta sul validation set e, nel caso queste non siano soddisfacenti, vengono cambiati gli iperparametri dell'algoritmo prima di eseguirlo nuovamente.

Al termine dell'esperimento, quando si sono validati un insieme di modelli prodotti impiegando differenti configurazioni di iperparametri, si sceglie quello che ha presentato le prestazioni migliori sul validation set e si valutano le sue prestazioni sul test set.

1.6.3 Cross Validation

La cross validation è una tecnica usata nel Machine Learning per mitigare il problema del overfitting. Consiste nel dividere il training set in un numero intero di sottoinsiemi disgiunti di uguale dimensione. Si procede poi iterando sui sottoinsiemi: ad ogni iterazione si valida il modello sul sottoinsieme selezionato dopo aver effettuato un passo di training sugli altri.

Oltre al overfitting, la cross validation permette di mitigare anche il problema del campionamento asimmetrico, che si verifica quando un sottoinsieme di dati che presentano caratteristiche comuni e peculiari rispetto ai restanti dati, viene inserito nel validation set e quindi escluso dal training set, rendendo il modello incapace di apprendere come sfruttare tali peculiarità in fase di classificazione.

1.7 Valutazione delle prestazioni

Una volta completata la fase di training è necessario effettuare una valutazione delle prestazioni del modello ottenuto e a tale scopo vengono identificate delle metriche che verranno monitorate durante il processo di allenamento. Dalle metriche selezionate deriva l'attendibilità del risultato ottenuto.

Il monitoraggio delle metriche può essere fatto sui valori grezzi calcolati tramite la formula della specifica metrica, ma riportare tali valori su un grafico permette di estrapolare in maniera più semplice informazioni sul-

l'andamento della metrica nel corso dell'allenamento [31]. In questa sezione vengono introdotte alcune metriche utili per condurre tale valutazione: la loss, l'accuratezza e la f1 measure. L'elenco non vuole essere esaustivo, ma si limita a introdurre quelle utilizzate nel corso dell'esperimento, trattato nel prossimo capitolo.

La loss è il valore calcolato usando la loss function selezionata e indica quanto la previsione effettuata dal modello è lontana rispetto al valore atteso. Essendo lo scopo dell'allenamento quello di minimizzare il valore della loss, l'andamento della metrica durante il training è idealmente decrescente, indice del fatto che il modello sta aggiornando efficacemente i pesi. La valutazione dell'andamento della loss permette di supporre, se sono presenti molte oscillazioni tra i valori registrati, l'inadeguatezza del algoritmo di ottimizzazione o la necessità di variare gli iperparametri.

Un ulteriore metodo di valutazione delle prestazioni della classificazione è rappresentato dallo studio della matrice di confusione. Questa è una tabella con numero di righe e di colonne pari al numero delle possibili classi del problema e dove le righe rappresentano le classi reali, mentre sulle colonne si trovano le classi effettivamente predette. Questa rappresentazione permette di identificare il tipo degli errori che sta commettendo il modello: non solo è possibile distinguere le celle di confusione, dove il modello non si comporta adeguatamente, ma risulta anche immediato intuire se l'eventuale problema in termini di prestazioni sia dovuto al training set poco equilibrato o se sia invece legato a un difetto del modello.

L'accuracy misura le prestazioni del modello in termini di quante delle previsioni effettuate sono corrette ed è espressa come percentuale. A differenza della loss, è sensato aspettarsi che l'andamento dell'accuratezza durante il training sia crescente, indice del fatto che dopo ogni iterazione di training il modello riesce a classificare correttamente più esempi rispetto a prima dell'iterazione stessa.

Trattandosi di un problema di classificazione, è possibile valutare le prestazioni usando la F1 Measure. Questa metrica permette di combinare i

valori di precision e recall: la prima è il rapporto tra il numero di esempi correttamente classificati rispetto al numero totale di esempi classificati, mentre la seconda è calcolata come rapporto tra il numero di esempi correttamente classificati e il numero totale di esempi della classe predetta. Mentre la precision misura quanto spesso il modello si sbaglia quando predice una certa classe, la recall indica quante delle previsioni di una certa classe sono corrette.

Combinare precision e recall nella F1 Measure permette di ordinare in maniera assoluta un insieme di modelli, mentre lasciandole separate si potrebbero avere situazioni scomode nelle quali un modello presenta una precision maggiore rispetto a un altro che invece presenta una recall maggiore. Nel valutare l'andamento della F1 Measure bisogna tener conto del fatto che un valore elevato indica che i valori di precision e recall sono a loro volta elevati e viceversa un valore di F1 Measure basso è indice di valori di precisione recall bassi. Un valore medio della metrica indica che solo una delle due componenti è carente, ma nel complesso in un addestramento efficace ci si aspetta un andamento della curva della metrica crescente.

1.8 Lavori simili

Nei primi anni gli studi si sono concentrati sull'impiego di modelli tradizionali per la classificazione e su features meno sofisticate.

Support Vector Machine (SVM) e Hidden Markov Model (HMM) hanno permesso di raggiungere prestazioni apprezzabili sul dataset Danish Emotional Speech (DES) [32]. Gli alberi decisionali, generati tramite l'algoritmo C4.5, sono stati utilizzati sul dataset Berlin per superare lo stato dell'arte dei tempi in cui è stato condotto lo studio [33]. Le regressioni logistiche binomiali sono state paragonate agli alberi decisionali in termini di prestazioni raggiunte su un dataset di tracce audio in lingua Malayalam, verificando un'accuratezza raggiunta dagli alberi decisionali maggiore rispetto a quella raggiunta dalle regressioni logistiche [34].

Le prestazioni di modelli quali Gaussian Mixture Model(GMM) e K-NN sul Berlin Emotional Speech dataset(BES) sono state paragonate, scoprendo che, nonostante l'accuratezza di GMM si sia rivelata maggiore su tutte le altre emozioni, K-NN ha ottenuto risultati migliori nella classificazione delle tracce audio che esprimevano felicità [35]. Durante un altro studio comparativo sono state paragonate le prestazioni di SVM e K-Nearest Neighbors su un database di tracce in lingua tedesca e sul dataset SAVEE, in inglese. Il numero delle features estratte è stato ridotto tramite l'impiego della Principal Component Analysis(PCA) e date in input ai modelli. L'algoritmo K-NN ha dato risultati migliori su entrambi i dataset [36].

Mentre inizialmente le informazioni principali usate per descrivere i dati erano la frequenza fondamentale, le frequenze formanti, velocità del parlato, jitter, shimmer e diverse features spettrali, alcuni studi si sono concentrati più sulla fase di ingegnerizzazione delle features ai fini di estrarre informazioni più significative che permettessero ai modelli di classificazione di raggiungere prestazioni migliori o di essere allenati in tempi più brevi a causa del ridotto numero di input da elaborare. Altri studi applicano tecniche di riduzione della dimensionalità, come la PCA, per decrementare il numero di features che verranno date in input al classificatore [37], mentre altri cercando di identificare le features che meglio descrivono la traccia audio, permettendo ai modelli una classificazione più efficace [38] [39].

Negli ultimi anni sono venute alla luce tecniche sempre più sofisticate per l'estrazione dalle tracce audio delle informazioni utili alla classificazione e a tal proposito si citano alcune features che hanno portato a risultati interessanti. Teaged Energy Operator(TEO) si è rivelata efficace nell'individuazione di emozioni quali rabbia, ansia, disgusto e tristezza: le features TEO sono state usate in congiunzione con quelle MFCC, estratte dal dataset Emo-DB e classificate attraverso GMM [40]. Le features Harmonic to Noise Rate(HRN) hanno permesso di ottenere buoni risultati sul dataset RML quando utilizzate insieme a MFCC, Zero Crossing Rate(ZCR) e TEO: la classificazione, fatta attraverso SVM, è stata preceduta da una fase di riduzione

della dimensionalità delle features ad opera di un auto-encoder [41].

Mentre la maggior parte degli studi sulla SER si concentrano sul mappare una traccia audio verso uno spazio finito, composto da un'enumerazione delle possibili emozioni riscontrabili, esistono tecniche che mappano invece tale traccia verso un punto di uno spazio tridimensionale [42], dove gli assi esprimono valenza, dominanza e attivazione, trasformando la questione in un problema di regressione. Questi approcci alternativi sono basati su studi psicologici che dimostrano che è possibile rappresentare un'emozione come un punto su uno spazio multidimensionale, anziché come valore categorico [43] [44].

Le modellazioni della SER come problema di classificazione statica o dinamica sono state messe a confronto in diversi studi: è emerso che, nei casi dove l'input non è una traccia audio breve, ma un discorso di media-lunga durata, la modellazione frame-based risulta più robusta in quanto non deve gestire la segmentazione dell'input ed è più sensibile ai cambiamenti emotivi tra un segmento e l'altro [45]. Altri studi hanno tuttavia dimostrato la superiorità della modellazione statica del problema [46] [47].

Con l'avvento del Deep Learning il problema della SER, come altri, è stato affrontato impiegando le principali architetture profonde, dalle architetture non ricorrenti come MLP e CNN alle architetture ricorrenti, quali RNN e LSTM. Si è visto, grazie agli studi in materia, come ogni architettura si porti dietro i propri pregi e benefici. Diversi modelli profondi sono stati usati per ottenere prestazioni migliori dello stato dell'arte su diversi benchmark: è il caso di Generalized Discriminant Analysis(GerDA) [48], un'architettura basata su una rete profonda, che ha superato i risultati fino ad allora migliori, ottenuti tramite SVM, su un insieme di 9 diversi datasets.

Un'architettura basata sulle celle LSTM è quella di Dual-Sequence LSTM (DS-LSTM) [49], che usata in un modello a due livelli ha permesso di ottenere buone prestazioni sul dataset IEMOCAP. La cella DS-LSTM processa simultaneamente due serie temporali, ottenute calcolando due spettrogrammi in scala Mel con frequenze temporali differenti, mentre il modello si compone di

due sottomodelli differenti: il primo impiega la cella DS-LSTM per analizzare le features citate poc'anzi e il secondo usa una LSTM comune per analizzare le features MFCC. L'output è calcolato a partire dalla concatenazione degli output dei sottomodelli.

Un'altra architettura basata sulle LSTM è quella delle 1D e 2D CNN LSTM [50]: questi due modelli sono composti da 4 Local Feature Learning Blocks (LFLB), seguiti da una cella LSTM e da un livello completamente connesso. I LFLB sono a loro volta composti da un livello di convoluzione e da uno di max pooling. Le due architetture sono caratterizzate dal tipo di convoluzione che applicano nei LFLB: i livelli di convoluzione possono infatti essere 1D o 2D. Le reti sono state collaudate sui databases EmoDB e IEMOCAP ed è emerso che l'accuratezza raggiunta dal modello 2D CNN LSTM risulta migliore sia di quella raggiunta dalla versione 1D che di quella del benchmark.

Ad oggi lo stato dell'arte sul dataset RAVDESS[1] è dettato dalla Temporal-aware bI-direction Multi-scale Network(TIM-Net) [51], un'architettura dove le features vengono processate da due flussi separati, subendo diverse operazioni di convoluzione ad opera di blocchi convoluzionali chiamati Temporal-Aware e facendo skip-connection della concatenazione dei risultati dei blocchi sui diversi flussi di elaborazione. L'aggregazione di questi output viene infine classificata da un unico livello denso.

[1] I dati sono stati ricavati da PaperWithCode: https://paperswithcode.com/sota/speech-emotion-recognition-on-ravdess

Capitolo 2

Metodologia, tecnologie e dataset

Nel secondo capitolo verrà descritto l'esperimento effettuato per valutare l'efficacia delle tecniche di Deep Learning nel contesto dello specifico problema affrontato, quello della Speech Emotion Recognition.

Nella prima sezione sarà fornita un'introduzione all'esperimento, partendo dagli obiettivi e dai passi seguiti, per concludere con gli strumenti utilizzati, in termini di librerie, per agevolare le operazioni necessarie. Nella sezione successiva si introdurrà RAVDESS, il dataset usato per condurre la prova. Viene descritta la composizione dei dati a disposizione, l'analisi svolta sui questi prima di iniziare con l'esperimento vero e proprio, le tecniche di data augmentation applicate per aumentare il la cardinalità del dataset e gli accorgimenti presi per mitigare il problema della degradazione delle prestazioni dovuto all'applicazione di tecniche con un costo computazionale degno di nota.

Successivamente si discuterà sulla divisione operata del dataset in training, validation e test set, evidenziando anche un difetto metodologico che porta alcune delle reti proposte in letteratura e sui forum ad ottenere prestazioni elevate in fase di valutazione del modello, ma grazie a una violazione dei principi della cross validation che inficia i risultati ottenuti. Verrà poi

descritto il procedimento seguito per allenare i vari modelli profondi e le metriche calcolate e memorizzate durante il corso dell'addestramento, citando infine alcune delle considerazioni fatte in merito al numero di cicli di training, al numero di dati passati in input alla rete ad ogni ciclo e all'effetto della data augmentation su questi due valori.

Saranno infine introdotti i modelli usati nel corso dell'esperimento, i principali livelli e le pratiche comuni adottate per mitigare i principali problemi che emergono quando si addestrano modelli profondi, quali overfitting, exploding gradient e vanishing gradient. I modelli descritti comprenderanno le reti convoluzionali, le reti ricorrenti standard, bidirezionali e stacked, e i multi classificatori.

2.1 Introduzione all'esperimento

L'obiettivo dell'esperimento è quello di valutare le prestazioni di vari modelli profondi applicati al problema della SER. Per condurre tale valutazione è stata esaminata principalmente la qualità dei modelli in termini di accuratezza di classificazione, ovvero la percentuale di esempi correttamente classificati rispetto alla totalità degli esempi processati. L'accuratezza interessante durante il corso della prova è quella relativa alla classificazione del validation set, mentre quella sul test set verrà calcolata come passo finale per abilitare il confronto con i risultati di altri esperimenti simili.

Lo studio è stato condotto selezionando un dataset composto da sole tracce audio, effettuando una fase di analisi ed elaborazione iniziale di questi dati; successivamente allo studio dei dati a disposizione sono state selezionate ed estratte dalle tracce audio le features, le informazioni ritenute più caratteristiche e discriminanti, necessarie per addestrare i modelli profondi in esame. Le prestazioni delle reti sono state raccolte e paragonate tra loro, per selezionare infine il modello che si è rivelato qualitativamente migliore. Tale modello è stato usato per classificare il test set e l'accuratezza raggiunta

è stata paragonata con le prestazioni dell'attuale stato dell'arte sul dataset RAVDESS.

2.1.1 Librerie usate

Librosa [52] fornisce gli strumenti fondamentali necessari ai sistemi di information retrieval che lavorano su dataset di tracce audio. Di questa libreria sono state sfruttate le funzioni di analisi audio: in particolare sono stati impiegati i metodi per l'estrazione delle features, per l'aumento delle serie temporali e per la visualizzazione ed analisi delle tracce audio contenute nel dataset.

La libreria open source Pandas è stata impiegata per elaborare ed analizzare i dati. Le strutture e le operazioni offerte hanno agevolato la manipolazione del dataset. L'oggetto Dataframe è stato usato per memorizzare le informazioni delle tracce audio del dataset, incluso il percorso su file system. Anche le features sono state salvate su un'istanza del medesimo oggetto e questo ha permesso di effettuare con un singolo comando operazioni che di norma richiederebbero la scrittura di una procedura algoritmica corrispondente. Anche il reshaping delle serie temporali, modifica della dimensione dei dati necessaria affinché possano essere usati come input per una rete neurale, è stato eseguito tramite l'apposito metodo fornito dalla libreria.

Le reti neurali profonde sono state costruite usando le funzioni offerte da Keras, libreria open source costruita sulla piattaforma TensorFlow che espone un'interfaccia ad un elevato livello di astrazione per la costruzione dei modelli. Keras permette di minimizzare il numero di azioni necessarie nei casi d'uso più comuni nel campo del Machine Learning e del Deep Learning e produce messaggi di errore chiari che agevolano la fase di debugging.

La libreria scikit-learn è un software distribuito con licenza libera, costruito su Numpy, SciPy e matplotlib per offrire supporto ad algoritmi di Machine Learning come classificatori, regressori o tecniche di clustering. Il modulo `metrics` espone diversi metodi per il calcolo di metriche e funzioni di score. Da questo modulo sono state adoperate le funzioni per la visualiz-

zazione della matrice di confusione, per il calcolo del F1 Measure di tutti i modelli e per quello dell'accuratezza dei multi classificatori. Per visualizzare la heatmap è stata impiegata la funzione omonima di Seaborn, libreria per la visualizzazione di dati statistici basata su matplotlib.

L'oggetto StandardScaler del modulo `preprocessing` di scikit-learn è stato impiegato per standardizzare le featues portando la media a 0 e la deviazione standard a 1, in modo da stabilizzare l'addestramento. Un'ulteriore standardizzazione a livello dei singoli batch è stata applicata in alcune reti usando un livello di BatchNormalization, per mitigare il problema del internal covariate shift. Il processo di standardizzazione tramite StandardScaler prevede due passi, uno di allenamento sulla totalità dei dati e uno di trasformazione sui dati selezionati. Nel primo passo viene fornito l'intero insieme delle features in modo che l'oggetto apprenda e memorizzi media e deviazione standard, mentre nel secondo passo viene applicata la standardizzazione sui soli dati forniti in input. Nel corso dell'esperimento i passi sono stati effettuati in maniera consecutiva per standardizzare tutte le features del dataset.

Imbalanced-learn è un'altra libreria open source utilizzata all'interno del progetto, che si appoggia a Scikit-learn e offre strumenti per gestire la classificazione di dataset sbilanciati, quindi dove la cardinalità dei dati appartenenti a una classe è molto minore rispetto alle altre. Della libreria è stata usata la funzione `SMOTENC`, contenuta nel modulo `over_sampling`, come metodo ulteriore per aumentare i dati del training set.

2.2 RAVDESS

Il dataset usato per condurre l'esperimento è RAVDESS Emotional speech audio, una sottoporzione del Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). Rispetto al RAVDESS puro, che comprende audio e video, parlato e musica, per un totale di quasi 25 GB, la sottoporzione usata in questa trattazione, che per comodità verrà chiamata solamente

RAVDESS da qui in poi, contiene solo audio parlato, per un totale di 450 MB.

Il dataset contiene 1440 file con estensione WAV: si tratta di 60 tracce per 24 attori di professione (12 maschi e 12 femmine). In ogni traccia un attore pronuncia una frase con un accento neutro del Nord America esprimendo un'emozione differente con un livello di intensità differente. Le possibili frasi che troviamo nei file audio sono due, "Kids are talking by the door" e "Dogs are sitting by the door", mentre i livelli di intensità delle emozioni, anch'essi due, sono normale e forte. Le possibili emozioni espresse sono 8: calma, felicità, tristezza, rabbia, paura, disgusto, sorpresa e un'emozione neutra che ammette un solo livello di intensità. Le 30 combinazioni possibili di frase, intensità ed emozione diventano 60 in quanto per ogni combinazione è presente una prima e una seconda ripetizione.

Le informazioni appena elencate sono codificate nel titolo delle singole tracce da indici, mentre una legenda del significato di tali indici è riportata sulla pagina Kaggle del dataset[1]. Un'informazione non estraibile dal nome dei files è quella dell'età degli attori, ma si legge nell'articolo dove viene illustrata la creazione del RAVDESS che l'età media degli attori è di 26 anni, per un range di valori compresi tra i 21 e i 33, con una deviazione standard di 3,75.

2.2.1 Analisi dei dati

Il dataset ha subito una fase di elaborazione che ha permesso di costruire, partendo dal titolo dei files audio, una tabella contenente i dati necessari per condurre l'analisi e, successivamente, per effettuare l'estrazione delle features e la classificazione. Per l'estrazione delle serie temporali dalle tracce audio è stata usata la funzione `load` di Librosa senza modificare la frequenza di campionatura di default, ovvero 20.050 HZ, che fa sì che vengano campionati 20.050 valori in virgola mobile ogni secondo: essendo gli audio lunghi in media

[1]Per la legenda dei titoli dei files del dataset RAVDESS si rimanda alla pagina https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio

4 secondi, i vettori dei campioni relativi alle singole tracce comprendono mediamente 88.200 valori.

Trattandosi di file audio sono state adottate tecniche di visualizzazione atte a produrre grafici che mostrassero la forma d'onda, lo spettro d'onda e lo spettrogramma.

- La forma d'onda costituisce una rappresentazione della quantità, del tipo e dell'ampiezza delle frequenze presenti nel suono analizzato. Graficamente rappresenta la frequenza nel tempo. Una forma d'onda è illustrata a titolo esemplificativo nella figura 2.1.

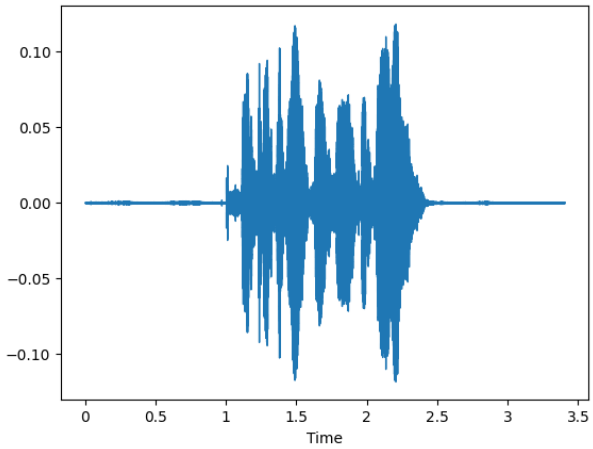

Figura 2.1: Esempio del grafico della forma d'onda di una traccia del dataset

- Lo spettro d'onda rappresenta la quantità di vibrazione ad ogni valore di frequenza: si tratta di un grafico con i valori di frequenza sulle ascisse e quelli di potenza o pressione sulle ordinate. Viene riportato a titolo di esempio uno spettro d'onda nella figura 2.2.

- Lo spettrogramma consiste invece nella rappresentazione grafica dell'intensità di un suono in funzione del tempo e della frequenza. Un esempio di spettrogramma è riportato nella figura 2.3.

Questo processo ha permesso di identificare pattern caratteristici delle specifiche emozioni, come la bassa velocità, l'elevata presenza di armoniche,

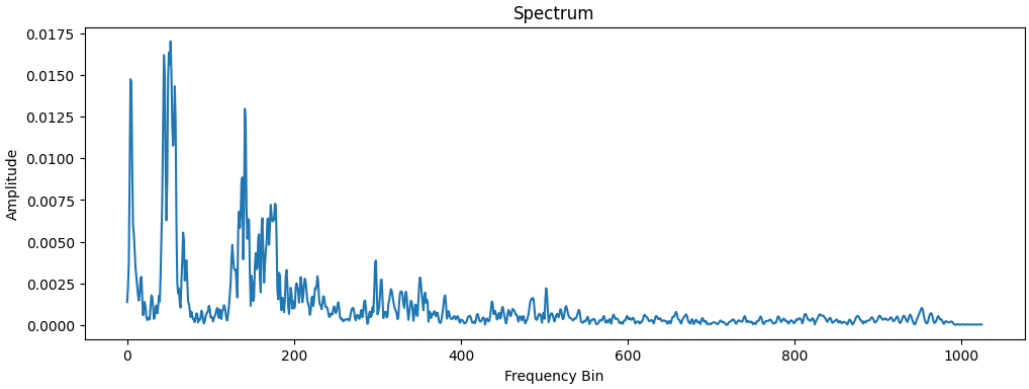

Figura 2.2: Esempio del grafico dello spettro d'onda di una traccia del dataset

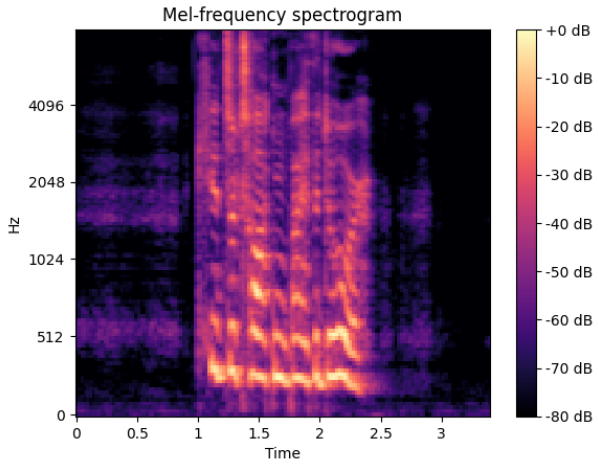

Figura 2.3: Esempio dello spettrogramma di una traccia del dataset

la scarsa variazione di tonalità del disgusto rispetto all'alta velocità, la scarsa presenza di armoniche e l'ampia variazione di tonalità della gioia.

L'analisi orientata al dataset nel suo insieme, anziché sulle singole tracce che lo compongono, ha confermato l'assenza di valori mancanti, assenza che nel caso specifico sarebbe stata riconducibile alla mancata raccolta dell'informazione più che alla non applicabilità dell'attributo ad alcuni elementi. Altra informazione fondamentale raccolta in fase di analisi del dataset è quella del leggero sbilanciamento delle classi rispetto all'emozione neutra, dovuta all'assenza del doppio livello di intensità, mentre la cardinalità degli esempi

appartenenti alle classi risulta costante per tutte le altre emozioni del dataset.

2.2.2 Augmentation

Per aumentare la dimensione del dataset sono state adottate diverse tecniche di data augmentation. Fondamentalmente è stato fatto ricorso a tecniche basate sulle funzioni offerte dalla libreria Librosa, che permettono di effettuare operazioni specifiche sugli audio per applicare filtri particolari, e funzioni standard del pacchetto NumPy, che alterano i valori della serie temporale estratta per aggiungere effetti meno specifici.

Quelle riportate di seguito sono le tecniche di augmentation utilizzate durante l'esperimento. Ognuna delle tecniche descritte è stata usata su ognuno dei dati del dataset, anche se le copie sintetiche sono state usate solo nel training set. Tuttavia l'operazione di augmentation ha permesso di portare la dimensione complessiva del dataset da 1.400 unità a 10.080, incrementando quindi la cardinalità di 7 volte rispetto al valore di partenza.

- `static noise` consiste nell'aggiunta di un rumore di fondo all'audio di input. È stata implementata modificando i valori dalla serie temporale con dei valori casuali generati in base a un fattore che specifica la potenza del rumore;

- `shift` applica uno spostamento all'audio del file a destra o a sinistra di un valore casuale compreso tra 0 a 1 secondo tramite la funzione `roll` di NumPy;

- `stretch` effettua il time stretching della traccia: il processo riguarda il cambio della velocità del segnale, ma non ne influenza l'intonazione. Graficamente effettua un appiattimento della forma d'onda e all'aumentare della forza dello stretching, risulterà aumentata anche la lunghezza dei suoni, mentre l'acutezza verrà diminuita. È stata implementata usando la funzione `time_stretch` di Librosa;

- `pitch` effettua il pitch shfit della traccia audio: questo comporta un aumento o riduzione della frequenza della nota musicale. L'effetto sulla forma d'onda è simile a quello apportato dall'operazione di time stretching, ma l'audio ottenuto ha un suono diverso. È stata implementata usando il metodo `pitch_shift` di Librosa;

- applicando la funzione di `dynamic change` verrà maggiorata la frequenza della nota più alta, modificando la forma d'onda, mentre il file audio risulterà inalterato all'udito. L'implementazione è avvenuta moltiplicando i valori della serie temporale per un valore casuale campionato da una distribuzione uniforme;

- `speed and pitch` cambia drasticamente la traccia audio comprimendola, senza però cambiare la durata della traccia. Il metodo adopera unicamente funzioni di NumPy.

In termini pratici viene usata la funzione di caricamento di Librosa per ottenere dal file audio la serie temporale di valori in floating point. Sulla serie temporale estratta vengono poi applicate le tecniche di augmentation. Infine, dalla lista delle serie temporali, reali e sintetiche, vengono estratte le features che costituiranno l'input dei modelli.

Un'ulteriore tecnica impiegata per aumentare ulteriormente il dataset è la Synthetic Minority Over-sampling TEchnique (SMOTE) [53]. Questa si fonda su un principio riconducibile a quello della tecnica Nearer Neighbors e permette la generazione di dati artificiali in maniera trasparente rispetto allo spazio dei dati, ma lavorando direttamente nello spazio delle features. SMOTE nasce con lo scopo di popolare artificialmente un dataset sbilanciato, dove ci sono quindi classi molto meno numerose di altre, in maniera da ridurre questa differenza. Per fare ciò la classe meno numerosa viene espansa introducendo nuovi dati.

L'algoritmo di campionatura segue questi passi: viene preso un punto casuale tra quelli appartenenti alla classe sbilanciata, vengono individuati i vicini di tale punto che appartengono alla stessa classe e si itera su questi,

partendo dal più vicino. Si calcola la distanza euclidea tra il punto di partenza e il suo vicino, quindi si moltiplica tale distanza per un numero casuale tra 0 e 1 e si somma il risultato al punto di partenza, ottenendo un nuovo punto nello spazio delle features che si trova sulla retta che congiunge i due appena menzionati. Si ripete l'algoritmo, cambiando punto di partenza e cambiando il vicino selezionato, finché non si è campionato il numero di elementi desiderato, per questo motivo vi è la necessità di passare come parametro il numero di nuovi esempi che si desidera produrre.

A differenza della tecnica SMOTE originale, è stata usata la funzione `SMOTENC`, dalla libreria open source Imbalanced-learn, che permette di generare dati sintetici anche per features categoriche o numeriche. Il metodo permette di selezionare la strategia di campionamento attraverso il parametro `sampling_strategy`, che consente di indicare per quale classe è necessario campionare i nuovi punti. Nel caso in esame è stato lasciato il valore di default, che indica all'algoritmo di campionare dati appartenenti alla classe minoritaria, fino a bilanciare il dataset, per poi procedere scegliendo casualmente la classe per la quale campionare il punto seguente ad ogni iterazione.

A causa dell'elevata quantità di tempo necessaria per passare dal dataset RAVDESS alla lista delle serie temporali aumentate e infine al dataframe delle features, si è scelto di effettuare questo procedimento una sola volta e di mantenere in memoria due file CSV: il primo contiene le features e le classi estratte dal dataset non aumentato, mentre il secondo contiene le stesse informazioni, ma estratte dal dataset aumentato. In questo modo si semplifica anche l'operazione di popolamento dei sottoinsiemi nel corso dell'addestramento: il training set verrà popolato dal CSV relativo al dataset aumentato, mentre validation e test set saranno costruiti prendendo i dati dall'altro CSV.

2.3 Divisione in sottoinsiemi

Essendo il dataset composto da tracce audio recitate da 24 attori diversi, si è optato per una suddivisione veicolata dall'attore al quale fa riferimento la traccia. L'insieme degli attori è stato diviso in 3 sottoinsiemi disgiunti: i primi due, composti da 4 attori ciascuno, sono stati usati per popolare test set e validation set mentre le tracce associate ai restanti attori sono state usate per costruire il training set. La cross validation è stata condotta quindi su un insieme comprensivo delle tracce di 16 attori. Si è praticamente effettuata una 16-fold cross validation, durante la quale il modello è stato allenato iterativamente usando un fold differente ad ogni iterazione come validation set e i restanti 15 come training set.

Un difetto metodologico comunemente riscontrato riguarda l'applicazione della data augmentation prima di effettuare la cross validation: questo fa sì, al momento della suddivisione dei dati in folds, che esempi aumentati simili ad esempi reali finiscano in folds diversi. Questo errore rappresenta una violazione dei principi alla base della cross validation, atti a rendere realistici i risultati ottenuti, quindi tali risultati, pur essendo ottimi, non sono significativi.

Per evitare l'errore si è fatto in modo che i dati aumentati venissero inclusi solo nel training set. Per ottimizzare le prestazioni, riducendo il carico computazionale dovuto alla ripetuta applicazione delle operazione di creazione dei dati sintetici, sono state salvate due copie del dataset delle features estratte: una aumentata e una no. In questo modo i dati necessari per comporre test set e validation set sono stati campionati dal dataset non aumentato, mentre quelli del training set dal dataset aumentato.

2.4 Training

L'allenamento dei vari modelli è avvenuto tramite cross-validation. La tecnica procede iterativamente, selezionando ad ogni iterazione uno degli attori del training set e allenando il modello su tutti gli altri. Terminato

l'allenamento sul insieme dei dati di training, alla rete vengono quindi forniti in input i dati relativi all'attore scartato dal training set e sui risultati della classificazione di questi vengono calcolate le metriche per valutare le prestazioni.

Il procedimento appena descritto è stato ripetuto numerose volte, in quanto i modelli profondi possono necessitare di un numero elevato di passi, nei quali aggiornano i pesi in base alla loss calcolata, per convergere a un minimo locale. Tuttavia incrementare arbitrariamente il numero di cicli di cross validation può non essere sempre la soluzione migliore. Può infatti capitare che l'eccessivo numero di cicli di addestramento porti il modello a diventare troppo specifico rispetto ai dati del training set.

Un altro aspetto da considerare è quello della cardinalità dell'insieme di dati passato come input al modello ad ogni passo di training. Tale insieme è detto mini-batch, o solamente batch per brevità, e la sua cardinalità indica il numero di dati che la rete elabora prima di aggiornare i propri pesi. La cardinalità dei batch è un altro iperparametro da valutare. In questa trattazione è stata usata sempre la dimensione di default del batch, ovvero 32 unità.

Operando una suddivisione del dataset in training, validation e test set veicolata dall'indice dell'attore che ha recitato le tracce, la data augmentation andrà ad aumentare il numero di esempi dati al modello in ogni passo di cross validation, che verranno suddivisi in batch, dopo ognuno dei quali verranno aggiornati i pesi della rete. Questo comporta sicuramente un aumento del tempo richiesto per svolgere ogni ciclo di cross validation, ma influenzerà anche il numero di cicli necessari per convergere.

Loss, accuratezza e F1 Measure sono state monitorate e in base a queste metriche sono stati aggiornati gli iperparametri per cercare di ottenere prestazioni migliori. I valori delle metriche sono stati calcolati, salvati e stampati in console al concludersi di ogni passo di ogni ciclo di cross validation e al termine dell'allenamento è stato creato un grafico che riportasse i valori memorizzati durante i cicli di addestramento, in maniera tale da esa-

minare l'andamento delle metriche durante il training e individuare eventuali problematiche emerse.

Il monitoraggio di loss e accuratezza permette di valutare l'efficacia dell'allenamento che si sta effettuando e in generale è possibile identificare le seguenti casistiche:

- la loss aumenta e l'accuratezza diminuisce, indice di un probabile difetto metodologico che fa sì che il modello elabori i dati senza imparare correttamente;
- la loss diminuisce ma l'accuratezza non aumenta: questo significa che il modello aggiorna correttamente i propri pesi, ma le prestazioni sul validation set non migliorano, il che suggerisce un caso di overfitting della rete sui dati di training;
- l'accuratezza aumenta e la loss diminuisce, quindi il modello sta lavorando correttamente, imparando dai dati del training set e aggiornando i pesi in maniera adeguata.

2.5 Modelli usati

Sono stati usati diversi modelli durante il corso dell'esperimento, ma per classificarli in maniera macroscopica è possibile identificare CNN e RNN. I modelli sono stati costruiti usando le funzioni offerte dalla libreria Keras e nel seguito della sezione sono descritti i componenti principali usati nelle varie reti profonde.

L'oggetto Input di Keras è stato usato per specificare la dimensione dei dati in ingresso della rete. Il costruttore istanzia un tensore della dimensione specificata e permette di disaccoppiare la struttura del modello dalla dimensione dell'input. La dimensione dell'input è pari al numero di features estratte dal file audio.

Le reti hanno come coda un livello denso, completamente connesso, implementato dall'oggetto Dense di Keras. Trattandosi di un problema di clas-

sificazione multiclasse la funzione di attivazione del livello denso finale è softmax, che mappa gli input in un range compreso tra 0 e 1 in maniera tale che la somma degli output sia unitaria. La dimensione dell'output corrisponde alla cardinalità dell'insieme delle classi, ovvero al numero di possibili emozioni del dataset.

Il livello denso finale è preceduto da una funzione Flatten che linearizza l'output multidimensionale proveniente dai livelli precedenti, convertendolo in forma monodimensionale, in modo che possa essere preso ed elaborato dal livello completamente connesso.

2.5.1 Pratiche comuni

Alcune pratiche sono state adottate per fronteggiare problemi comuni dell'addestramento di reti neurali quali overfitting, vanishing ed exploding gradient.

Come algoritmo di ottimizzazione è stato adottato ADAptive Moment estimation(ADAM), che combina le proprietà di altri due algoritmi: momentum e RMSprop. Come il momentum, ADAM accelera la convergenza dell'algoritmo di discesa stocastica del gradiente aggiungendo un iperparametro che pesa l'aggiornamento dei pesi tenendo conto anche dell'aggiornamento effettuato nel passo precedente. In questo modo se il gradiente punta nella stessa direzione iterazione dopo iterazione, l'aggiornamento dei pesi sarà sempre maggiore, mentre quando il gradiente cambia di segno tra due passi si eseguirà uno spostamento minore nella nuova direzione assunta da quest'ultimo.

Da RMSprop ADAM prende l'idea di impiegare un learning rate differente per ogni parametro, ovvero per ogni peso. I singoli valori dei learning rates verranno adattati al parametro, effettuando aggiornamenti maggiori a quelli aggiornati più frequentemente e viceversa saranno applicati aggiornamenti meno drastici ai pesi che vengono aggiornati più spesso.

Un accorgimento da adottare quando si usa ADAM è quello di salvare lo stato interno del ottimizzatore insieme ai pesi del modello, altrimenti,

se dopo il caricamento di tali pesi si effettuerà una nuova fase di training usando quegli stessi pesi, il modello opererà in maniera differente. Keras offre un metodo `save_model` che permette di salvare un modello scegliendo tra diversi formati. Il formato HDF5 supporta il salvataggio automatico dello stato dell'ottimizzatore.

In diverse reti è stato introdotto il *dropout* in uno o più livelli hidden. Questo metodo di regolarizzazione ha lo scopo di evitare l'overfitting riducendo l'apprendimento interdipendente dei neuroni. Viene introdotto un iperparametro che indica la probabilità che un neurone ha di essere ignorato nel passo attuale dell'allenamento e questo fa sì che la reti diventi più robusta, addestrandosi usando informazioni parziali. Per questo il dropout riduce la quantità di tempo necessaria per concludere ogni passo di allenamento, ma la rete richiederà più iterazioni per convergere.

Usare una probabilità di disattivazione troppo bassa ha un effetto minimale sul training, mentre usare un valore troppo elevato impedisce alla rete di imparare efficacemente. Le buone pratiche nel campo dell'addestramento di reti profonde suggeriscono un valore compreso tra il 20% e il 50%, dove il primo è anche consigliato come punto di partenza. Il meccanismo del dropout viene attuato solo in fase di training, mentre in test e validation la rete opererà con tutti i neuroni attivi per sfruttare al massimo le proprie capacità.

Durante l'allenamento di diverse reti ricorrenti è capitato di osservare il problema del exploding gradient: in pratica il valore della loss calcolato al termine dell'iterazione risultava indefinito, NaN. Per mitigare il problema è stata adottata la tecnica del gradient clipping, impostando il parametro `clipvalue` esposto da ogni ottimizzatore della libreria Keras per definire il valore massimo e minimo che può assumere il gradiente durante l'aggiornamento dei pesi tramite retropropagazione.

Per mitigare in maniera preventiva il problema del vanishing gradient è stata impostata ReLU come funzione di attivazione dei livelli ricorrenti. Essendo ReLU inferiormente limitata all'origine, allevia il problema della saturazione dei neuroni durante la retropropagazione dell'errore in quanto la

sua derivata è 1 per input positivi e 0 per input negativi: questa attivazione sparsa riduce il numero di moltiplicazioni per valori prossimi a 0 durante l'aggiornamento dei pesi, permettendo l'allenamento di reti molto profonde senza che si verifichi il problema della scomparsa del gradiente.

Su alcuni livelli hidden è stata impiegata la *batch normalization* per mitigare il problema del *internal covariate shift*. Il *covariate shift* è un problema dovuto all'appartenenza a regioni differenti dello spazio delle features dei dati appartenenti a batch diversi che porta al rallentamento dell'addestramento della rete a causa dei miglioramenti minimali del modello a seguito dell'aggiornamento dei pesi. La soluzione per la problematica appena illustrata è quella di mescolare i dati prima di comporre i batch.

Nelle reti profonde lo stesso problema può verificarsi nei livelli interni prendendo il nome di internal covariate shift e causando disagi più difficilmente rilevabili. La batch normalization è una soluzione proposta per risolvere tale problema e consiste nell'introduzione di due parametri addestrabili, un fattore di scale e uno di shift, che permettono al livello di decidere se effettuare l'operazione omonima. L'operazione effettuata porta il batch in una forma normalizzata, ovvero ad avere media nulla e deviazione standard unitaria. Oltre al problema appena introdotto, la batch normalization riduce la necessità di un'accurata inizializzazione dei pesi della rete e abilita all'utilizzo di learning rates più elevati, in grado di velocizzare l'addestramento riducendo il numero di iterazioni necessarie per convergere.

2.5.2 Reti convoluzionali

Nelle CNN sono stati usati livelli di convoluzione e di pooling rispettivamente per estrarre le informazioni e per aggregarle. Questi modelli sono generalmente composti da blocchi che comprendono un certo numero di livelli di convoluzione seguiti da uno di pooling.

Di norma i livelli di convoluzione operano su input tridimensionali applicando un filtro, detto *kernel*, che non è altro che una piccola matrice di valori che viene fatta scorrere sull'input, calcolando per ogni posizione il prodotto

scalare tra i valori del filtro e i valori dell'input in esame. Nel caso delle reti ricorrenti non si lavora su input tridimensionali, ma su serie temporali, quindi l'operazione è svolta utilizzando livelli di *convoluzione 1D*, dove un filtro monodimensionale viene fatto scorrere sull'unica dimensione della serie di input, calcolando il prodotto scalare tra i valori del filtro e i valori selezionati dell'input.

I livelli di *pooling* vengono usati per aggregare le informazioni estratte dai livelli convoluzionali. L'idea è quella di mantenere le caratteristiche dominanti riducendo la dimensione dell'input. Viene fatta scorrere una finestra sull'input che ne aggrega i valori calcolando la media, nei livelli di *average pooling*, o il valore massimo, in quelli di *max pooling*. Tra i benefici apportati da questi livelli si citano la riduzione del numero di parametri, della sensibilità al rumore, del rischio di overfitting e della potenza computazionale richiesta nelle operazioni successive. In particolare, l'operazione di max pooling permette di scartare attivazioni dovute al rumore, rendendo più robusta la rete.

Come per i livelli convoluzionali, lavorando con le serie temporali è necessario che l'operazione di aggregazione avvenga su un input monodimensionale e per questo sono stati impiegati livelli di *pooling 1D*. In questi livelli viene fatta scorrere una finestra sulla sequenza di input e per ogni posizione viene calcolato il risultato dell'aggregazione dei valori in base al metodo di pooling selezionato.

2.5.3 Reti ricorrenti

Le reti ricorrenti permettono di elaborare serie temporali mantenendo un'astrazione di memoria durante il corso dell'elaborazione, atta a memorizzare informazioni sugli input già processati.

I livelli ricorrenti usati sono le celle RNN e quelle LSTM. Entrambi i livelli operano elaborando un elemento della serie temporale per volta, calcolando l'output dell'istante corrente in base a questo input e al hidden state dell'istante precedente. Ambedue i livelli citati offrono però due metodologie

di utilizzo differenti, in termini di funzionamento e di prestazioni: nel primo caso l'output della cella viene definito come il valore prodotto da questa dopo aver elaborato l'ultimo elemento della serie temporale, mentre nel secondo caso viene restituita la lista degli output della cella dopo l'elaborazione di ogni elemento della serie temporale. Nel primo caso l'output è composto da uno scalare per ognuna delle classi, mentre nel secondo è una lista di vettori con una cardinalità della lista pari al numero di elementi costituenti la serie temporale.

Sono stati provati modelli con un numero di celle ricorrenti in sequenza superiore a uno, facendo stacking di RNN e LSTM per aumentare la complessità dell'elaborazione e della rete, cercando di ottenere un modello più efficiente. Per fare stacking di livelli ricorrenti è necessario che questi restituiscano l'intera sequenza delle previsioni nei singoli istanti temporali, in questo modo ogni cella in ogni istante emetterà un nuovo hidden state, passandolo alla cella stessa nell'istante temporale successivo e alla cella seguente nell'istante attuale. Questo permette alla rete di raggiungere capacità di rappresentazione superiori, ma modelli di questo tipo diventano velocemente preda di un overfitting a un grado così elevato che neanche il dropout riesce a mitigare.

2.5.4 Reti bidirezionali

I modelli bidirezionali consentono alla rete di basare la propria classificazione non solo sull'istante temporale attuale e su quelli passati, ma anche su quelli futuri. Sono implementati impiegando due reti ricorrenti per elaborare la serie temporale in entrambi i versi ed effettuando un'aggregazione sugli output prodotti nei singoli istanti temporali.

Il metodo Bidirectional di Keras permette di istanziare un livello bidirezionale, prendendo in input il livello ricorrente da usare per elaborare l'input nelle due direzioni, dall'inizio alla fine e viceversa, e la modalità di fusione degli output parziali ottenuti ad ogni istante temporale. Così sono stati creati i livelli bidirezionali delle reti usate durante l'esperimento, mentre co-

me sottomodelli si sono usate celle RNN o LSTM semplici, con un numero limitato di neuroni. Il metodo permette di specificare una cella diversa per effettuare l'elaborazione dell'input nel senso opposto a quello standard, ma nel corso dell'esperimento sono state impiegate unicamente reti bidirezionali simmetriche.

Sono state effettuate alcune considerazioni sulla metodologia di aggregazione degli output delle singole celle ricorrenti usate nei livelli bidirezionali. Somma, media e prodotto sono operazioni che restituiscono un output della stessa dimensione delle sequenze in input, mentre la concatenazione somma le due liste, raddoppiando la cardinalità dell'output. Mentre somma e prodotto soffrono di più per la presenza di eventuali outliers, la media permette di alleviare la presenza di valori che si discostano fortemente dagli altri, per questo è stata scelta come metodo di aggregazione dei risultati delle singole celle ricorrenti.

2.5.5 Multi classificatori

Usare più classificatori per etichettare i dati e aggregare le loro previsioni può portare a un aumento delle prestazioni significativo se si tiene conto di alcune questioni legate alla correlazione degli errori dei sottomodelli. I requisiti necessari affinché un multi classificatore porti a risultati migliori rispetto ai classificatori semplici sono l'indipendenza, cioè gli errori dei singoli classificatori non devono essere correlati, e che le prestazioni dei singoli classificatori siano superiori a quelle di un classificatore dummy, che effettua le sue previsioni scegliendo in maniera casuale una classe dal relativo dominio.

Per questo motivo è stata tentata la strada dei multi classificatori: il modello usato è composto da più sottomodelli, uno per ognuna delle emozioni del dataset, che sono stati allenati concentrandosi solo sul riconoscimento di una specifica emozione rispetto a tutte le altre. Questo approccio permette ai singoli classificatori di affrontare un sottoproblema binario, che risulta più semplice del problema totale, e di aggregare le previsioni per produrre l'output del multi classificatore.

I sottomodelli differiscono dai modelli precedentemente descritti solo per il numero di neuroni nel livello di output, che passa da un numero pari a quello delle possibili emozioni nel dataset a 1. A differenza delle classificazioni fatte dai modelli esposti fino ad ora, dove dato un input veniva fatta un'unica previsione restituendo un vettore delle confidenze relative ad ognuna delle classi, nel caso della multi classificazione verranno fatte 8 previsioni differenti per ogni singolo dato, ognuna relativa a una specifica emozione. Il singolo classificatore restituirà un livello di confidenza che indica la probabilità, secondo il modello, che il dato appartenga alla classe in esame.

L'aggregazione dei risultati dei sottomodelli è stata effettuata con un approccio basato sul livello di confidenza: è stata selezionata la classe il cui modello ha restituito il valore di confidenza maggiore rispetto agli altri. Tali valori rappresentano la certezza che il modello ha che il dato appartenga alla classe in esame, quindi è ragionevole scegliere la classe che è stata votata con il valore di probabilità maggiore rispetto alle altre.

Ulteriori accorgimenti possono venir adottati per migliorare le prestazioni dei multi classificatori, ma si rimanda al capitolo 3 per gli ulteriori dettagli.

Capitolo 3

Risultati sperimentali

Nel terzo e ultimo capitolo verranno riportati i risultati dell'esperimento effettuato e già introdotto nel capitolo precedente. Dopo aver descritto il contesto dell'esperimento e le metodologie e tecnologie impiegate, verranno analizzati i risultati relativi a ognuna delle tipologie di modelli usati e si cercherà di identificare le possibili cause che hanno portato a un livello di qualità differente rispetto alle altre reti valutate nel corso della medesima prova pratica.

Nella prima sezione vengono riassunti i valori misurati e valutati per identificare la qualità dei singoli modelli. La metrica principale adoperata è l'accuratezza, nello specifico l'accuratezza sul validation set, mentre per valutare la qualità del processo di allenamento sono state usate le curve del grafico di loss e accuratezza. Si è deciso di aggiungere una sezione dedicato all'inizializzazione dei pesi delle reti a seguito delle difficoltà riscontrate in fase di allenamento delle reti LSTM: queste si sono infatti rivelate molto sensibili ai valori dei pesi impostati dalla funzione di inizializzazione al momento della compilazione del modello.

Nella terza sezione sono elencati i modelli usati nel corso dell'esperimento. Tale elenco è corredato da un'introduzione alla struttura dei modelli, da tabelle che riportano i valori di accuratezza ottenuti a seguito dell'addestramento effettuato e dalle considerazioni che è stato possibile trarre dal-

l'analisi dei dati appena menzionati. Gli esperimenti sui modelli sono stati condotti su due differenti insiemi di dati, la prima ottenuta come risultato dell'estrazione delle features da RAVDESS e la seconda invece ricavata applicando le stesse tecniche di estrazione, ma sulla versione aumentata del dataset, prodotta applicando le tecniche di data augmentation già descritte. Si esamineranno anche le differenze tra i risultati ottenuti dagli stessi modelli allenati sulla versione aumentata e non aumentata del dataset. I modelli descritti nella sezione sono le reti ricorrenti semplici, costruite con celle RNN e LSTM, le stesse reti, ma con l'intera sequenza delle previsioni in output, le reti bidirezionali, le reti ricorrenti con stacking di celle RNN e LSTM, le reti convoluzionali e i multi classificatori.

Nella quarta sezione verrà eletto il modello che si è dimostrato qualitativamente migliore rispetto agli altri esaminati nel corso della trattazione. Le prestazioni e la struttura di questa rete saranno messe a confronto con quelle di TIM-Net, ad oggi considerato lo stato dell'arte della SER su RAVDESS. Infine, nell'ultima sezione si illustreranno quelli che potrebbero essere gli sviluppi futuri per migliorare i risultati ottenuti.

3.1 Metriche usate

L'accuratezza è la metrica principale usata per valutare le prestazioni del modello ottenuto in seguito alla fase di allenamento ed è calcolata come rapporto tra il numero di elementi classificati correttamente rispetto al numero di elementi classificati. Durante lo svolgimento dell'esperimento il valore di accuratezza considerato interessante è stato quello raggiunto sul validation set, mentre per valutare la qualità del modello prodotto rispetto ad altri modelli in letteratura è stato necessario basarsi sul valore raggiunto sul test set. Basandosi sul valore di questa metrica si sono modificati gli iperparametri nel tentativo di migliorare la qualità del modello. Gli iperparametri più incisivi sulle prestazioni si sono rivelati essere il learning rate, il numero di pesi addestrabili e il numero di cicli di cross validation effettuati, ma il tuning

non ha permesso di guadagnare più di qualche punto percentuale in termini di accuratezza.

Per quanto riguarda il valore della metrica rispetto alla qualità del modello, un'accuratezza elevata sul training set rispetto ad un valore molto minore ottenuto sul validation set è stato considerato indice del fatto che la rete impiegata era troppo complessa, portando al problema del overfitting. Le soluzioni tentate per risolvere la problematica sono state solitamente quelle di ridurre il learning rate o il numero di neuroni nei livelli hidden del modello e valutare l'eventuale aumento di qualità della rete. Per evitare il presentarsi di tale problematica si è introdotta una percentuale di dropout in alcuni dei livelli interni.

Un altro valore tenuto monitorato durante l'addestramento è stato quello della loss. Per le reti particolarmente complesse si è notato che tale valore, dopo un certo numero di passi di training, presentava valori indefiniti(NaN). Partendo dall'ipotesi che si trattasse di un caso di exploding gradient, la soluzione efficace è stata quella di usare gradient clipping nel ottimizzatore. La funzione di loss più utilizzata durante lo svolgimento dell'esperimento è stata la binary cross entropy, calcolata con la formula seguente in funzione del vettore delle classi predette ($\hat{y}$) e quello delle classi reali (y):

$$-y \cdot \log(\hat{y}) - (1 - y) \cdot \log(1 - \hat{y})$$

Valutando nel grafico l'andamento delle curve di accuratezza e loss è stato possibile identificare punti oltre i quali la loss subiva un forte aumento, mentre l'accuratezza decrementava di colpo. Questo problema è stato mitigato introducendo del dropout, solitamente un 20%, e facendo batch normalization in uno o più dei livelli hidden. In questo modo si è incrementata la stabilità del training, risolvendo il problema appena citato. La F1-measure è stata inserita in un grafico a parte per non peggiorare la visibilità del primo ed è stata calcolata usando la funzione `f1_score` di scikit-learn: è stato usato il modello per effettuare una previsione sui dati di test e le classi predette

sono state passate insieme alle classi reali come input alla funzione per il calcolo della metrica.

Per ogni modello sono state valutata anche la matrice di confusione e la heatmap dei risultati ottenuti dalla classificazione del validation set. La prima permette di ottenere più informazioni sull'esito di tale classificazione, ma la seconda si presta a una visualizzazione più rapida e permette di carpire le informazioni necessarie in maniera più intuitiva. La figura 3.1 illustra un esempio di heatmap prodotta usando la libreria Seaborn: questa modalità di rappresentazione permette in maniera lampante di capire che il modello ha classificato in maniera accurata i dati che esprimevano l'emozione sorpresa, confermato dal colore chiaro nella cella in basso a destra, mentre le prestazioni sull'emozione paura sono state pessime, infatti la colonna corrispondente a tale emozione è completamente nera. Inoltre è possibile notare alcuni comportamenti anomali come la classificazione della tristezza come calma e della rabbia e del disgusto come sorpresa.

3.2 Pesi ed inizializzazione

È già stata discussa l'importanza della quantità dei pesi della rete nel corso dell'addestramento: un numero elevato di pesi addestrabili richiederà una quantità di tempo maggiore per concludere ogni passo di training. Oltre alla questione del tempo, va considerato anche l'aumento del numero di moltiplicazioni effettuate durante l'applicazione dell'algoritmo di retropropagazione dell'errore tramite discesa stocastica del gradiente. L'elevato numero di moltiplicazioni rende la rete più sensibile ai problemi del vanishing ed exploding gradient. Ulteriore aspetto da valutare è quello della complessità del modello, infatti una rete con più pesi imparerà rappresentazioni più sofisticate dei dati, rischiando però, se si esagera con l'addestramento, di non riuscire poi a generalizzare sui nuovi dati.

I problemi elencati sono mitigabili tramite una corretta regolazione degli iperparametri: valore del learning rate, numero di livelli, di pesi e di cicli

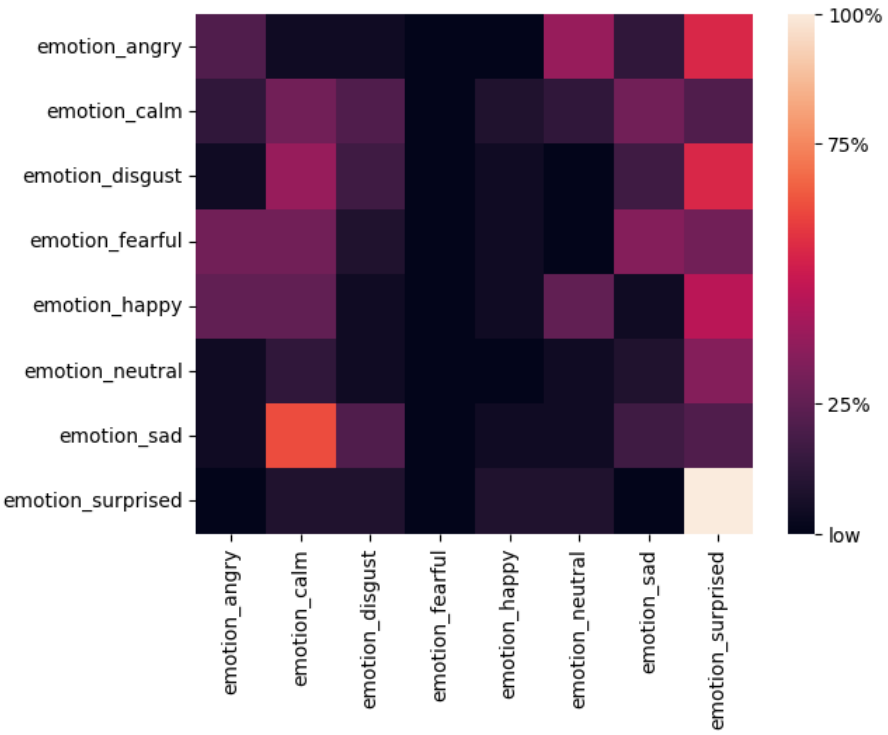

Figura 3.1: Esempio di heatmap prodotta con Seaborn

di training. Un problema riscontrato durante lo svolgimento della prova sperimentale riguarda invece l'inizializzazione dei valori dei pesi addestrabili. Alcuni modelli si sono rivelati particolarmente sensibili all'inizializzazione dei tali valori, presentando comportamenti molto differenti già dalle prime iterazioni di addestramento, mentre in alcuni casi si verificava quello che è stato classificato come problema del exploding gradient, restituendo valori di loss indefiniti. Questo è il caso specifico delle LSTM, che si sono rivelate molto sensibili a questa problematica, richiedendo di compilare nuovamente il modello più volte fino ad ottenere una configurazione di valori dei parametri che non desse difficoltà di questo tipo già dal primo passo di allenamento.

La classe alla quale Keras delega l'inizializzazione dei pesi dipende dal singolo livello in esame, ma la più comune, usata dai livelli convoluzionali e dalle celle RNN e LSTM, è la classe `GlorotUniform`, che popola i valori dei pesi campionando punti casuali da una distribuzione uniforme. Tale distribu-

zione è limitata nell'intervallo $[-limit, limit]$, dove i parametri sono definiti secondo la seguente formula in funzione delle variabili fan_in e fan_out, che rappresentano rispettivamente il numero di unità di input e di output nel tensore dei pesi :

$$\sqrt{\frac{6}{(fan\_in + fan\_out)}}$$

Una soluzione interessante proposta per risolvere il problema dell'inizializzazione dei pesi prevede l'aggiunta di un rumore adattivo ai primi stati della rete con l'obiettivo di decrementare la loss calcolata nei primi istanti temporali, portando alla conseguente riduzione dell'impatto dell'inizializzazione non corretta e permettendo al modello di tarare i propri pesi in base ai dati ricevuti in input senza ottenere un overflow della loss calcolata[54].

3.3 Prestazioni dei modelli

La presente sezione ha lo scopo di illustrare i risultati ottenuti a seguito dell'addestramento di vari modelli profondi. Ogni sottosezione comprende una differente categoria di rete, mentre le tabelle racchiudono generalmente le stesse informazioni: uno mnemonico che identifichi il modello, il numero di parametri addestrabili della rete e il valore di accuratezza raggiunto sul validation set.

I modelli sono stati addestrati in sequenza, cambiando al termine di ogni sessione di allenamento gli iperparametri per rendere il fine tuning di questi automatico e salvando al contempo il modello appena prodotto. I modelli sono stati valutati in gruppo successivamente. Il training dei modelli più lenti in termini di velocità di addestramento è avvenuto su un cluster munito di GPU Turing pilotate con driver Nvidia v.740 e librerie di computazione CUDA 11.4, mentre gli altri allenamenti sono stati effettuati su Google Colaboratory.

3.3.1 Reti ricorrenti semplici

Le prime prove si sono concentrate sulla valutazione di modelli semplici, composti da un unico livello ricorrente, RNN o LSTM. Come ottimizzatore è stato usato ADAM con il learning rate di default, che è stato ridotto per allenare le LSTM con numero di neuroni elevato, mentre come loss è stata usata la binary cross-entropy. Si è usata inizialmente una versione non aumentata del dataset e si sono effettuati cicli di cross validation finché l'accuratezza mostrava un trend crescente.

Successivamente il dataset è stato aumentato, portando la sua cardinalità da 1.400 a 10.080, e si sono ripetute le prove precedenti per registrare il nuovo livello di accuratezza raggiunto. A causa dell'aumento del numero di esempi nel training set è stata riscontrata una diminuzione del numero di cicli di cross validation necessari per convergere, mentre nel caso delle LSTM il learning rate ha dovuto subire un ulteriore riduzione per concludere il training.

La tabella 3.1 riporta il tipo di cella ricorrente usato come livello interno del modello seguito dal numero di neuroni in tale livello, il numero di parametri addestrabili totale della rete e l'accuratezza massima raggiunta sul validation set dopo l'addestramento sul training set aumentato e non. I modelli sono composti dalla cella ricorrente indicata in tabella dotata di un numero di neuroni pari al valore tra parentesi nella stessa colonna. Le celle usano ReLU come funzione di attivazione, batch normalization e un 20% di dropout. La rete usa ADAM come ottimizzatore, binary cross-entropy come loss di classificazione e softmax come funzione di attivazione del livello completamente connesso.

Dai risultati è stato identificato un miglioramento delle prestazioni delle reti con celle RNN con pochi neuroni a seguito dell'aumento del dataset, mentre le celle più complesse presentano un grado di overfitting maggiore già dalle prime iterazioni di training. Nel caso delle celle LSTM l'aumento delle prestazioni risulta meno omogeneo. Si citano inoltre le difficoltà riscontrate nel corso dell'addestramento delle celle LSTM con molti neuroni: queste celle sembrano soffrire fortemente dell'assenza di un'inizializzazione appro-

Cella (neuroni)	Parametri addestrabili	Accuratezza(sul validation set)	
		Training set non aumentato	Training set aumentato
RNN(8)	168	0,1875	0,2375
RNN(16)	456	0,2333	0,3500
RNN(32)	1.416	0,2917	0,2500
RNN(64)	4.872	0,3000	0,2875
LSTM(8)	408	0,3500	0,2624
LSTM(16)	1.320	0,3083	0,3833
LSTM(32)	4.680	0,3583	0,2750
LSTM(64)	17.544	0,3625	0,3833

Tabella 3.1: Tabella comparativa dei modelli RNN e LSTM base allenate sul training set aumentato e non.

priata dei pesi, infatti, nonostante le precauzioni prese applicando la batch normalization e il gradient clipping, già dopo la prima iterazione di training la loss calcolata è risultata essere spesso NaN, sintomo di un overflow del valore dovuto probabilmente al problema dell'exploding gradient. Scalare il learning rate di due ordini di grandezza ha permesso di mitigare il problema, ma ha prolungato fortemente il tempo necessario per raggiungere i risultati riportati in tabella a seguito del training.

Alla luce di tali difficoltà si è deciso di procedere nelle prove successive utilizzando sia il dataset aumentato che quello non aumentato per le reti con numero limitato di parametri, come nel caso dei modelli bidirezionali e delle reti ricorrenti stacked, mentre si è adoperato solo il dataset non aumentato per le reti con decine o centinaia di migliaia di parametri addestrabili, ovvero le reti ricorrenti con l'intera sequenza delle previsioni in output e le CNN.

È stata inoltre notata, visualizzando la matrice di confusione e la heatmap, una tendenza del modello a classificare l'emozione tristezza come calma e la paura come tristezza. Questo errore può dipendere dalla durata insuf-

ficiente dell'addestramento che non ha permesso alla rete di imparare una suddivisione qualitativamente migliore dello spazio delle features, risultando in una non corretta classificazione di alcune classi piuttosto che altre, dove invece le prestazioni si sono rivelate migliori, come nel caso dell'emozione disgusto, che è stata correttamente individuata in una buona percentuale dei dati del validation set.

3.3.2 Reti ricorrenti con intera sequenza in output

Per costruire le reti successive si sono modificate le celle dei modelli precedenti perché non restituissero in output una sola previsione per ogni classe, ma affinché fornissero una lista di previsioni per ogni elemento della sequenza temporale, previsioni ottenute usando l'input dell'istante corrente e le informazioni memorizzate nel hidden state sugli input precedenti. L'output della cella è diventato quindi una matrice, che ha richiesto uno step di linearizzazione tramite il metodo `Flatten` di Keras per essere convertito in una lista di valori. Questa differenza nell'output della cella ha fatto lievitare fortemente i pesi addestrabili della rete, influendo anche sulle prestazioni dell'addestramento in termini di tempo necessario per concludere ogni iterazione.

Nella tabella 3.2 è riportato il tipo di cella ricorrente, con parametro `return_sequence` positivo, usato come livello interno del modello seguito dal numero di neuroni in tale livello, dal numero di parametri addestrabili totale della rete e dall'accuratezza raggiunta sul validation set dopo l'addestramento sul training set non aumentato.

Le reti con cella RNN hanno giovato maggiormente dell'informazione sulle previsioni dei singoli istanti temporali, presentando valori di accuratezza che superano il 40%, rispetto al massimale del 30% raggiunto nella prima prova. Nelle reti LSTM il miglioramento è minimo, probabilmente per via della selezione che tali celle fanno sull'informazione da mantenere nel hidden state, che permette loro di restituire un output basato anche su informazioni significative dei primi elementi della serie temporale. Le celle RNN non

Cella (neuroni)	Parametri addestrabili	Accuratezza(sul validation set)
RNN(8)	12.456	0,4116
RNN(16)	25.032	0,4375
RNN(32)	50.568	0,4333
LSTM(8)	12.696	0,3333
LSTM(16)	25.896	0.3375
LSTM(32)	53.832	0,3417

Tabella 3.2: Tabella comparativa dei risultati dei modelli ricorrenti con l'intera sequenza in output sul dataset non aumentato.

supportano questa astrazione di memoria a lungo termine, quindi basare la classificazione del modello sull'intera serie delle previsioni effettuate dalla cella porta a un miglioramento più accentuato.

3.3.3 Reti ricorrenti bidirezionali

Nelle reti successive è stato usato un livello bidirezionale invece di una cella ricorrente standard. Il livello usa la stessa cella RNN o LSTM per elaborare l'input in entrambe le direzioni. La coppia di output in ogni istante temporale viene aggregata facendo la media dei valori. Il numero di pesi risulta quasi raddoppiato in ognuno dei modelli, ma le celle usate contano pochi neuroni, evitando l'eccessiva dilatazione dei tempi necessari per l'addestramento.

Nella tabella 3.3 sono riportati i modelli bidirezionali utilizzati: nella prima colonna, tra parentesi, si trova il numero di neuroni della cella ricorrente usata per costruire il livello bidirezionale. Le altre colonna comprendono il numero di parametri addestrabili e l'accuratezza raggiunta sul validation set dopo l'allenamento del modello sul training set aumentato e non.

I modelli con celle RNN hanno migliorato leggermente le loro prestazioni, fatta eccezione per il caso con cella RNN a 16 neuroni allenata sul data-

Cella (neuroni)	Parametri addestrabili	Accuratezza(sul validation set)	
		Training set non aumentato	Training set aumentato
BRNN(8)	232	0,2292	0,2875
BRNN(16)	712	0,3417	0,2375
BLSTM(8)	712	0,2417	0,2500
BLSTM(16)	2.440	0,2125	0,2708

Tabella 3.3: Tabella comparativa dei risultati dei modelli bidirezionali sul dataset aumentato e non.

set aumentato, migliore nella versione non bidirezionale probabilmente per via di un'inizializzazione fortuita dei pesi rispetto a quanto accaduto nella controparte bidirezionale. Per quel che riguarda le celle LSTM, è evidente un peggioramento del grado di accuratezza raggiunto: ciò può dipendere dal fatto che l'informazione derivante dall'elaborazione inversa della serie temporale non si è rivelata utile ai fini della classificazione, oppure dalla durata non sufficiente dell'addestramento. Studi in letteratura evidenziano un aumento delle epoche di training necessarie a modelli di questo tipo per convergere[55].

I risultati lasciano supporre che il problema non si presti particolarmente alla classificazione con modelli bidirezionali, probabilmente a causa del tipo di features usate. È possibile che features diverse forniscano informazioni più adatte ai fini dell'elaborazione effettuata da questi modelli.

3.3.4 Reti ricorrenti con stacking

Dopodiché sono stati provati alcuni modelli facendo stacking di celle ricorrenti. L'aumento del numero di parametri addestrabili nei modelli dipende non solo dall'aumentata profondità della rete, ma anche dal fatto che le celle RNN e LSTM devono restituire in output alla cella successiva l'intera sequenza delle previsioni effettuate in ogni istante temporale.

La tabella 3.4 riporta il tipo delle celle usate nel modello e, tra parentesi, il numero di neuroni di ognuna di queste celle. Tali informazioni sono seguite dal numero di parametri addestrabili e dal grado di accuratezza raggiunto dal modello sul validation set dopo essere stato allenato sul dataset aumentato e non.

Cella (neuroni)	Parametri addestrabili	Accuratezza(sul validation set)	
		Training set non aumentato	Training set aumentato
RNN(8/8)	304	0,1958	0,2000
RNN(16/16)	984	0,1833	0,2167
LSTM(8/8)	952	0,3667	0,2708
LSTM(16/16)	3.432	0,2667	0,3500

Tabella 3.4: Tabella comparativa dei risultati dei modelli costruiti facendo stacking di livelli ricorrenti e allenandoli sul dataset aumentato.

Dai risultati si evince che il livello di complessità introdotto dall'aggiunta della seconda cella ricorrente in coda alla prima non ha portato a un miglioramento significativo dei risultati, ma nella quasi totalità delle prove effettuate si è osservato un grado di accuratezza minore rispetto alla controparte con singola cella ricorrente. Queste singole celle si portano dietro un grado di profondità intrinseco elevato e sembra che lo stacking renda i modelli molto sensibili al overfitting, permettendogli di imparare rappresentazioni dei dati in input che portano a prestazioni elevate sul training set, ma a prestazioni mediocri sui dati non incontrati in fase di training.

L'analisi della matrice di confusione e della heatmap ha permesso di rilevare un'elevata accuratezza in corrispondenza di emozioni quali calma e sorpresa, mentre la classificazione dell'emozione neutra ha dato risultati pessimi.

3.3.5 Reti convoluzionali

Le reti convoluzionali sono state costruite usando blocchi costituiti da due livelli consecutivi di convoluzione monodimensionale seguiti da un livello di max pooling 1D. Sono state costruite 2 reti differenti impiegando rispettivamente 1 e 2 dei blocchi appena descritti. In seguito è stata replicata la prova aumentando il numero di filtri applicati nel primo livello convoluzionale di ogni blocco per valutare l'efficacia di tale aumento rispetto al incremento di profondità della rete.

I livelli di convoluzione monodimensionale impiegati prevedono l'applicazione dei 128 o 256 filtri di dimensione 5, fattore di stride unitario e l'opzione che permette di aggiungere padding automaticamente all'input per fare in modo che nessun elemento a fine sequenza venga troncato durante l'applicazione dei filtri. Ognuno di questi livelli usa ReLU come funzione di attivazione, batch normalization e un 10% di dropout. I livelli di convoluzione sono seguiti da un livello di max pooling unidimensionale senza stride, senza padding e con dimensione della finestra di pooling 8, che indica il numero di elementi dell'input da aggregare. Il livello denso finale usa softmax come funzione di attivazione, mentre la rete usa la binary cross-entropy come loss di classificazione e ADAM come ottimizzatore.

La tabella 3.5 presenta nella prima colonna uno mnemonico della rete che identifica il numero di blocchi convoluzionali contenuti in tale modello e, tra parentesi, il numero di filtri impiegati nei livelli di convoluzione di tali blocchi. A tali informazioni seguitano il numero di parametri addestrabili e l'accuratezza raggiunta in fase di validazione a seguito dell'addestramento sul dataset non aumentato.

L'aumento di profondità a seguito dell'aggiunta di blocchi convoluzionali ha portato ad un aumento delle prestazioni, anche se minimo, in tutti e tre i modelli. L'aumento del numero di filtri ha giovato alla qualità della rete più di quanto abbia fatto l'aumento del numero di livelli convoluzionali presenti in ogni blocco. Dalla disamina della matrice di confusione e della heatmap si è notata una tendenza dei modelli a classificare i dati che presen-

Rete	Parametri addestrabili	Accuratezza(sul validation set)
CNNx1(128,128)	107.400	0,3958
CNNx2(128,128)	249.992	0,4291
CNNx1(256,128)	190.088	0,4042
CNNx2(256,128)	498.184	0,4500
CNNx1(128,128,128)	189.448	0,4092
CNNx2(128,128,128)	414.088	0,4208

Tabella 3.5: Tabella comparativa dei risultati dei modelli convoluzionali sul dataset non aumentato.

tano l'emozione rabbia, disgusto o paura con l'emozione sorpresa. Nel caso particolare dell'emozione paura, nessun dato è mai stato etichettato come appartenente alla classe da nessuno di questi modelli, probabilmente perché le features estratte sono state riconosciute come meno informative per tale classe rispetto alle altre.

3.3.6 Multi classificatore

Il multi classificatore è stato costruito allenando singoli modelli su una semplificazione binaria del problema. Usando il dataset così come descritto fino ad ora per fare una classificazione binaria bisogna considerare lo sbilanciamento del dataset conseguente al fatto di prendere in considerazione una sola emozione per volta. Tutti gli altri dati apparterranno alla classe negativa, rendendo il dataset sbilanciato e l'accuratezza calcolata finora diverrà non significativa. Le reti classificheranno i dati come appartenenti alla classe negativa perché risulta essere la classe più numerosa e così facendo raggiungeranno un grado di accuratezza elevato, ma per via di un errore metodologico.

Per ovviare a tale problematica sono state vagliate due possibilità, citate

entrambe per completezza: la prima prevede il campionamento, dall'insieme dei dati di classe negativa, di un numero di elementi pari alla cardinalità dell'insieme dei dati della classe positiva. In questo modo i dati sui quali si allena il modello risultano bilanciati in termini di appartenenza alle due classi possibili, ma utilizzare una porzione ridotta dei dati richiederà un maggior numero di passi di addestramento per convergere.

La seconda opzione, quella adottata praticamente, prevede l'impiego di una metrica alternativa all'accuratezza per valutare le prestazioni del modello nonostante lo sbilanciamento delle classi. La metrica usata è l'accuratezza bilanciata, calcolata sulla base dell'implementazione proposta da scikit-learn per il metodo `balanced_accuracy_score`: l'accuratezza bilanciata viene calcolata come media dei valori di recall ottenuti per ognuna delle classi. Nel caso in esame il problema prevede due solo classi possibili, una positiva e una negativa, così il valore della metrica può essere calcolato come metà della somma tra true positive rate e true negative rate, essendo rispettivamente i valori di recall rispetto alla classe positiva e negativa.

Sì è dovuta implementare la metrica da zero per via dell'incompatibilità dell'output dei modelli di Keras, che restituiscono dei tensori, e l'input della funzione di scikit-learn per il calcolo dell'accuratezza bilanciata, che invece accetta array NumPy. L'implementazione è stata fatta usando le operazioni del modulo `backend` di Keras, in modo da poter operare direttamente sui tensori restituiti in output dal modello. I sottomodelli usati per costruire il multi classificatore sono reti con una singola cella LSTM da 8 neuroni, ReLU come funzione di attivazione, batch normalization e un 10% di dropout. Nel livello completamente connesso è stata sostituita la funzione di attivazione, impostando softmax. Il modello usa ADAM come ottimizzatore, l'accuratezza bilanciata come metrica e la binary cross-entropy come loss di classificazione.

Nonostante gli accorgimenti presi in termini di metriche, il modello ha raggiunto un'accuratezza del 39,58%, inferiore rispetto alla totalità dei modelli convoluzionali esaminati. Dall'analisi della matrice di confusione e della

heatmap si deduce che il modello classifica efficacemente le emozioni calma, paura e sorpresa, mentre sembra carente nell'individuazione di tutte le altre. In particolare il modello sembra scambiare spesso gioia e disgusto per rabbia, nonostante si sia insistito sull'addestramento dei sotto modelli responsabili della classificazione delle prime due classi.

3.4 Esito

La rete che ha raggiunto i risultati migliori sul validation set è stata infine testata sul test set. La rete in questione è il modello convoluzionale a due blocchi, entrambi con due livelli di convoluzione, uno da 256 filtri e l'altro da 128, e un livello di max pooling. Tale modello ha raggiunto un'accuratezza del 45,00% sul validation set e una del 36,06% sul test set. Risultati simili, anche se leggermente peggiori, sono stati raggiunti dalle reti con celle RNN con l'intera sequenza delle previsioni in output. Le reti ricorrenti hanno presentato il maggior grado di difficoltà in fase di addestramento, causato dalla forte sensibilità al problema del vanishing gradient e dall'elevata incapacità di generalizzare raggiunta dai modelli già dopo poche iterazioni di addestramento, nonostante gli accorgimenti presi per mitigare la problematica. Un altro limite è stato imposto dal elevata quantità di tempo necessaria per allenare reti ricorrenti con un numero elevato di neuroni.

Come già menzionato nel primo capitolo, lo stato dell'arte della SER sul dataset RAVDESS ad oggi è fatto da TIM-Net, con un'accuratezza del 92,08%. La rete, come gli altri modelli che furono in cima alla classifica prima dell'avvento di TIM-Net, ha alla base le CNN, anche se l'implementazione risulta piuttosto sofisticata. Questo sembra suggerire la superiorità dei modelli convoluzionali rispetto a quelli ricorrenti, probabilmente perché i secondi non risultano essere ancora una tecnologia abbastanza matura per ottenere gradi di accuratezza similari ai modelli convoluzionali.

Dalle analisi condotte sulle matrici di confusione e sulle heatmap ottenute a seguite della classificazione del validation set effettuata dai modelli si è

potuto notare un comportamento costante, non dipendente dal modello in esame: l'emozione sorpresa è stata riconosciuta correttamente quasi per tutti i dati del validation set, ma a causa dello sbilanciamento della classificazione verso questa classe, infatti anche rabbia, disgusto e gioia sono state classificate spesso come sorpresa. Essendo questo un comportamento comune, che prescinde il tipo di modello utilizzato, si può supporre che dipenda da una mancanza nelle features estratte delle informazioni necessarie a discriminare le tre emozioni erronee rispetto alla sorpresa o dalla necessità delle reti di un addestramento più duraturo per imparare a modellare una funzione efficace nella discriminazione di tali emozioni.

3.5 Sviluppi futuri

I margini di miglioramento delle reti illustrate sono molteplici. Partendo dalla fase di estrazione delle features, potrebbero essere identificate features più informative rispetto all'emozione espressa nell'audio e si potrebbero addirittura identificare features specifiche per le singole emozioni. In termini di elaborazione di tali features si possono introdurre nei modelli costrutti non utilizzati in questa trattazione, ma che si sono rivelati funzionali secondo a quanto attesta la letteratura. In particolare in questa sezione verranno citati gli strumenti usati nel modello TIM-Net, concentrandosi su quelli non adoperati nel corso della prova sperimentale precedentemente esposta. Anche utilizzando gli strumenti elencati nei capitoli precedenti è possibile costruire modelli più complessi, che possono portare a un aumento del grado di accuratezza. Infine, non è da escludere l'eventualità che l'introduzione di tecniche sofisticate di data augmentation possa permettere un incremento delle prestazioni delle reti.

È possibile adottare alcuni accorgimenti per incrementare ulteriormente le prestazioni di un multi classificatore. L'obiettivo è quello di rendere i singoli classificatori meno dipendenti tra loro, ovvero si cerca di ridurre la correlazione tra i loro errori usando classificatori diversi sullo stesso training

set o lo stesso classificatore su training set diversi.

Si possono quindi usare classificatori diversi o features diverse, estratte con algoritmi differenti, mentre allenare il modello su un sottoinsieme delle features estratte potrebbe non essere la soluzione adatta al caso in esame, in quanto si potrebbero perdere informazioni relative agli istanti temporali passati. Cambiare gli iperparametri della rete porterà l'algoritmo di allenamento a produrre classificatori differenti. Un'altra opzione, che non esclude quelle già citate, è quella di adottare una delle tecniche chiamate bagging e boosting: nel primo caso si allenerà il modello su porzioni differenti del training set, mentre nel secondo approccio si insisterà nell'allenamento sui batch che hanno portato a valori di accuratezza minori, ottenendo in entrambi i casi modelli differenti. Superare i limiti dei modelli generativi imposti dalla distribuzione dei dati sintetici ottenuti simile a quella dei dati usati per la generazione potrebbe abilitare l'impiego di tecniche di data augmentation più sofisticate e basate sul Deep Learning, usando modelli come GANs o auto encoders per produrre dati utili al modello per generalizzare sugli esempi nel validation e nel test set.

Le prestazioni raggiunte da TIM-Net sono dovute all'elaborazione complessa dell'input fatta da questo modello. Rispetto agli strumenti illustrati nel presente documento vengono introdotti i concetti di dropout spaziale e di skip connection. Il dropout spaziale è un meccanismo che rispecchia il funzionamento del dropout standard, ma ad essere disattivate non sono singoli elementi del layer, ma intere feature maps. In questo modo viene promossa l'indipendenza tra le feature maps a costo di un rallentamento dell'apprendimento della rete maggiore rispetto a quello comportato dal dropout standard. La skip connection invece permette all'output di un livello di saltare alcuni dei livelli successivi: questo permette di allenare reti molto profonde riducendo la degradazione delle prestazioni ed evitando il problema del vanishing gradient. Nel caso di TIM-Net la skip connection porta l'output di alcuni livelli direttamente ai livelli completamente connessi: l'output della sequenza di livelli convoluzionali e quelli provenienti dalle skip connections vengono

quindi aggregati e classificati per produrre il risultato del modello.

Con i dovuti accorgimenti è possibile introdurre livelli di complessità senza peggiorare le prestazioni della rete, sia in termini di capacità di generalizzazione che di tempo richiesto per l'addestramento del modello. Altro punto sul quale si potrebbe concentrare l'attenzione è quello dell'abbattimento dei tempi necessari per effettuare l'allenamento delle reti ricorrenti con un numero elevato di neuroni, limitazione che si è rivelata forte nel corso dell'esperimento, nonostante l'impiego di GPU. A seguito di un eventuale riduzione dei tempi di addestramento rimarrebbero poi da valutare le prestazioni di tali modelli, che rischiano di essere molto soggetti al problema del overfitting a causa sia della profondità intrinseca di tali reti che dell'ampiezza dei livelli ricorrenti in termini di numero di neuroni.

Conclusioni

In questo volume di tesi è stato descritto l'esito dello studio condotto sul problema della Speech Emotion Recognition impiegando algoritmi di Deep Learning. Si è introdotto il contesto dell'esperimento in maniera sommaria, per poi scendere nei dettagli dei singoli aspetti considerati durante lo svolgimento della parte pratica dello studio, descritta anch'essa nel capitolo dedicato. Si è partiti da un'introduzione teorica nel primo capitolo, con l'intenzione di fornire gli strumenti necessari per capire le scelte operate e descritte nel seguito del documento. Nei capitoli successivi c'è stato un graduale passaggio agli aspetti più pratici dei temi trattati fino ad arrivare all'esposizione del esperimento effettuato e all'analisi dei risultati ottenuti, tentando di identificare le cause degli errori commessi dalle reti e gli eventuali margini di miglioramento.

É stato descritto il dataset impiegato, RAVDESS, la fase di analisi condotta su questo, la data augmentation fatta per incrementare il numero dei dati a disposizione e la conseguente estrazione delle features che si sono reputate adatte per consentire la classificazione dei dati ad opera dei modelli profondi. Il processo di addestramento di tali modelli è stato descritto e analizzato in termini di parametri e iperparametri che lo influenzano maggiormente. Sono stati forniti i dettagli fondamentali sulle principali tipologie di reti profonde adottabili per questo problema di classificazione multi classe e infine sono state introdotte le tecniche di valutazione delle prestazioni dei modelli appena citati.

Successivamente sono state esposte le metodologie, le tecnologie e gli ac-

corgimenti adottati per effettuare la prova sperimentale. A seguito di tale prova sono stati riportati e analizzati i risultati ottenuti, valutando le prestazioni dei modelli in termini di accuratezza raggiunta sul validation set, e infine sul test set, di andamento delle curve di loss, accuratezza e F1-Measure e di comportamento dei modelli in termini di classificazione delle specifiche classi analizzando le matrici di confusione e le heat maps.

I dati emersi dalle prove effettuate dimostrano che il problema non risulta complesso solo per l'essere umano, ma anche le reti neurali profonde riscontrano difficoltà nella classificazione delle emozioni espresse nelle tracce audio. Dal paragone effettuato tra le prestazioni dei modelli ottenuti nella fase sperimentale e quelle del modello che rappresenta lo stato dell'arte della SER su RAVDESS, TIM-Net, è stato possibile identificare la necessità di introdurre un grado di elaborazione dei dati superiore per ottenere risultati soddisfacenti. Il grado di accuratezza massimo raggiunto dai modelli esaminati è del 36,06%, con una forte propensione a classificare le tracce con l'emozione disgusto. Si è potuto inoltre affermare che i modelli convoluzionali riescono a raggiungere prestazioni migliori sul problema in esame, ipotizzando che ciò dipenda da una maggior maturità raggiunta da tali modelli rispetto alle reti ricorrenti.

Bibliografia

[1] Björn W. Schuller. Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends. *Commun. ACM*, 61(5):90–99, apr 2018. ISSN 0001-0782. doi: 10.1145/3129340.

[2] Min Chen, Ping Zhou, and Giancarlo Fortino. Emotion communication system. *IEEE Access*, 5:326–337, 2017. doi: 10.1109/ACCESS.2016.2641480.

[3] Erik Cambria, Amir Hussain, Catherine Havasi, and Chris Eckl. Sentic computing: Exploitation of common sense for the development of emotion-sensitive systems. *Development of Multimodal Interfaces: Active Listening and Synchrony: Second COST 2102 International Training School, Dublin, Ireland, March 23-27, 2009, Revised Selected Papers*, pages 148–156, 2010.

[4] Wu Li, Yanhui Zhang, and Yingzi Fu. Speech emotion recognition in e-learning system based on affective computing. In *Third international conference on natural computation (ICNC 2007)*, volume 5, pages 809–813. IEEE, 2007.

[5] GM Hancock, PA Hancock, and CM Janelle. The impact of emotions and predominant emotion regulation technique on driving performance. *Work*, 41(Supplement 1):3608–3611, 2012.

[6] Edward Sucksmith, Carrie Allison, Simon Baron-Cohen, Bhismadev Chakrabarti, and Rosa A Hoekstra. Empathy and emotion reco-

gnition in people with autism, first-degree relatives, and controls. *Neuropsychologia*, 51(1):98–105, 2013.

[7] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of big data*, 2 (1):1–21, 2015.

[8] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning, 2022.

[9] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.

[10] Hanzhang Wang, Hanli Wang, and Kaisheng Xu. Evolutionary recurrent neural network for image captioning. *Neurocomputing*, 401:249–256, 2020.

[11] Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. An lstm approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018)*, pages 214–223, 2018.

[12] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE, 2015.

[13] Takaaki Hori, Jaejin Cho, and Shinji Watanabe. End-to-end speech recognition with word-based rnn language models. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 389–396. IEEE, 2018.

[14] Dimitrios Ververidis and Constantine Kotropoulos. Emotional speech recognition: Resources, features, and methods. *Speech communication*, 48(9):1162–1181, 2006.

[15] Fabien Ringeval, Andreas Sonderegger, Juergen Sauer, and Denis Lalanne. Introducing the recola multimodal corpus of remote collaborative and affective interactions. In *2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG)*, pages 1–8. IEEE, 2013.

[16] Ya Li, Jianhua Tao, Linlin Chao, Wei Bao, and Yazhu Liu. Cheavd: a chinese natural emotional audio–visual database. *Journal of Ambient Intelligence and Humanized Computing*, 8:913–924, 2017.

[17] Stefan Steidl. *Automatic classification of emotion related user states in spontaneous children's speech*. Logos-Verlag Berlin, Germany, 2009.

[18] John HL Hansen, Sahar E Bou-Ghazale, Ruhi Sarikaya, and Bryan Pellom. Getting started with susas: a speech under simulated and actual stress database. In *Eurospeech*, volume 97, pages 1743–46, 1997.

[19] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42:335–359, 2008.

[20] Abhinav Dhall, Roland Goecke, Simon Lucey, Tom Gedeon, et al. Collecting large, richly annotated facial-expression databases from movies. *IEEE multimedia*, 19(3):34, 2012.

[21] Ellen Douglas-Cowie, Nick Campbell, Roddy Cowie, and Peter Roach. Emotional speech: Towards a new generation of databases. *Speech communication*, 40(1-2):33–60, 2003.

[22] Lin Zhang, Steffen Walter, Xueyao Ma, Philipp Werner, Ayoub Al-Hamadi, Harald C Traue, and Sascha Gruss. "biovid emo db": A multimodal database for emotion analyses validated by subjective ratings. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2016.

[23] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[24] Aggelina Chatziagapi, Georgios Paraskevopoulos, Dimitris Sgouropoulos, Georgios Pantazopoulos, Malvina Nikandrou, Theodoros Giannakopoulos, Athanasios Katsamanis, Alexandros Potamianos, and Shrikanth Narayanan. Data augmentation using gans for speech emotion recognition. In *Interspeech*, pages 171–175, 2019.

[25] Lu Yi and Man-Wai Mak. Improving speech emotion recognition with adversarial data augmentation network. *IEEE transactions on neural networks and learning systems*, 33(1):172–184, 2020.

[26] Nilanjan Dey and Amira S Ashour. *Direction of arrival estimation and localization of multi-speech sources*. Springer, 2018.

[27] Soumya Sen, Anjan Dutta, and Nilanjan Dey. *Audio processing and speech recognition: concepts, techniques and research overviews*. Springer, 2019.

[28] Soumya Sen, Anjan Dutta, Nilanjan Dey, Soumya Sen, Anjan Dutta, and Nilanjan Dey. Speech processing and recognition system. *Audio Processing and Speech Recognition: Concepts, Techniques and Research Overviews*, pages 13–43, 2019.

[29] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 21–26, 2006.

[30] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings. IEEE International Conference on Multimedia and Expo*, volume 1, pages 113–116. IEEE, 2002.

[31] Wei Dai and Daniel Berleant. Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics. In *2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI)*, pages 148–155. IEEE, 2019.

[32] Yi-Lin Lin and Gang Wei. Speech emotion recognition based on hmm and svm. In *2005 international conference on machine learning and cybernetics*, volume 8, pages 4898–4901. IEEE, 2005.

[33] Julia Sidorova. Speech emotion recognition with tgi+. 2 classifier. In *Proceedings of the Student Research Workshop at EACL 2009*, pages 54–60, 2009.

[34] Agnes Jacob. Modelling speech emotion recognition using logistic regression and decision trees. *International Journal of Speech Technology*, 20(4):897–905, 2017.

[35] Rahul B Lanjewar, Swarup Mathurkar, and Nilesh Patel. Implementation and comparison of speech emotion recognition system using gaussian mixture model (gmm) and k-nearest neighbor (k-nn) techniques. *Procedia computer science*, 49:50–57, 2015.

[36] Mohammed Jawad Al Dujaili, Abbas Ebrahimi-Moghadam, and Ahmed Fatlawi. Speech emotion recognition based on svm and knn classifications fusion. *International Journal of Electrical and Computer Engineering*, 11(2):1259, 2021.

[37] Fatemeh Noroozi, Marina Marjanovic, Angelina Njegus, Sergio Escalera, and Gholamreza Anbarjafari. Audio-visual emotion recognition in video clips. *IEEE Transactions on Affective Computing*, 10(1):60–75, 2017.

[38] Ruhul Amin Khalil, Edward Jones, Mohammad Inayatullah Babar, Tariqullah Jan, Mohammad Haseeb Zafar, and Thamer Alhussain. Speech emotion recognition using deep learning techniques: A review. *IEEE Access*, 7:117327–117345, 2019.

[39] Turgut Özseven. A novel feature selection method for speech emotion recognition. *Applied Acoustics*, 146:320–326, 2019.

[40] Surekha Reddy Bandela and T Kishore Kumar. Stressed speech emotion recognition using feature fusion of teager energy operator and mfcc. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5. IEEE, 2017.

[41] Pavol Harár, Radim Burget, and Malay Kishore Dutta. Speech emotion recognition with deep learning. In *2017 4th International conference on signal processing and integrated networks (SPIN)*, pages 137–140. IEEE, 2017.

[42] Dongrui Wu, Thomas D. Parsons, Emily Mower, and Shrikanth Narayanan. Speech emotion estimation in 3d space. In *2010 IEEE International Conference on Multimedia and Expo*, pages 737–742, 2010. doi: 10.1109/ICME.2010.5583101.

[43] James A Russell. Core affect and the psychological construction of emotion. *Psychological review*, 110(1):145, 2003.

[44] Harold Schlosberg. Three dimensions of emotion. *Psychological review*, 61(2):81, 1954.

[45] Juan Pablo Arias, Carlos Busso, and Nestor Becerra Yoma. Energy and f0 contour modeling with functional data analysis for emotional speech detection. In *Interspeech*, pages 2871–2875, 2013.

[46] Björn Schuller, Bogdan Vlasenko, Florian Eyben, Gerhard Rigoll, and Andreas Wendemuth. Acoustic emotion recognition: A benchmark comparison of performances. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 552–557. IEEE, 2009.

[47] Bogdan Vlasenko, Björn Schuller, Andreas Wendemuth, and Gerhard Rigoll. Frame vs. turn-level: emotion recognition from speech considering static and dynamic processing. In *Affective Computing and Intelligent Interaction: Second International Conference, ACII 2007 Lisbon, Portugal, September 12-14, 2007 Proceedings 2*, pages 139–147. Springer, 2007.

[48] André Stuhlsatz, Christine Meyer, Florian Eyben, Thomas Zielke, Günter Meier, and Björn Schuller. Deep neural networks for acoustic emotion recognition: Raising the benchmarks. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5688–5691. IEEE, 2011.

[49] Xinghao Jiang, Ke Xu, and Tanfeng Sun. Action recognition scheme based on skeleton representation with ds-lstm network. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7):2129–2140, 2019.

[50] Jianfeng Zhao, Xia Mao, and Lijiang Chen. Speech emotion recognition using deep 1d & 2d cnn lstm networks. *Biomedical signal processing and control*, 47:312–323, 2019.

[51] Jiaxin Ye, Xin-Cheng Wen, Yujie Wei, Yong Xu, Kunhong Liu, and Hongming Shan. Temporal modeling matters: A novel temporal emotional modeling approach for speech emotion recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[52] Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal ana-

lysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25, 2015.

[53] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[54] Hans-Georg Zimmermann, Christoph Tietz, and Ralph Grothmann. Forecasting with recurrent neural networks: 12 tricks. *Neural Networks: Tricks of the Trade: Second Edition*, pages 687–707, 2012.

[55] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292. IEEE, 2019.

[56] Qi rong Mao, Xiaojia Wang, and Yongzhao Zhan. Speech emotion recognition method based on improved decision tree and layered feature selection. *Int. J. Humanoid Robotics*, 7:245–261, 2010.

[57] Iulia Lefter, Leon JM Rothkrantz, Pascal Wiggers, and David A Van Leeuwen. Emotion recognition from speech by combining databases and fusion of classifiers. In *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic, September 6-10, 2010. Proceedings 13*, pages 353–360. Springer, 2010.

[58] Mohammed Jawad and Abbas Ebrahimi-Moghadam. Speech emotion recognition: A comprehensive survey. *Wireless Personal Communications*, 129:1–37, 03 2023. doi: 10.1007/s11277-023-10244-3.

[59] Akash Shaw, Rohan Kumar Vardhan, and Siddharth Saxena. Emotion recognition and classification in speech using artificial neural networks. *International Journal of Computer Applications*, 145(8):5–9, 2016.