UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche

Corso di Laurea Triennale in Ingegneria Informatica

Tesi di Laurea Triennale

Tecniche di Deep Learning nel Riconoscimento delle Emozioni da Segnali Vocali

Laureanda

Daniela Nardone

Matricola 577084

Relatore

Prof. Fabio Gasparetti

Anno Accademico 2023/2024

It is never too late to be what you might have been.

Introduzione

A partire dagli ultimi decenni, l'intelligenza artificiale (**IA**) rappresenta sempre maggiormente una delle innovazioni più trasformative dei nostri tempi grazie alla sua influenza in numerosi ambiti della vita umana.

Partendo dalla sanità e dall'istruzione e passando per il settore industriale e l'intrattenimento, l'IA ha trasformato radicalmente il modo in cui le macchine interagiscono con gli esseri umani, spaziando dalla risoluzione di problemi complessi fino all'adattamento alle esigenze delle persone.

Tra i molteplici campi di applicazione, il riconoscimento delle emozioni è diventato un'area di interesse cruciale.

In particolare, il riconoscimento delle emozioni a partire da segnali vocali, in inglese **Speech Emotion Recognition** (**SER**), si occupa di identificare e classificare le emozioni espresse attraverso la voce sfruttando caratteristiche del parlato, come il tono, il ritmo, il volume e l'intonazione, per analizzare lo stato emotivo dell'interlocutore.

Al fine di migliorare l'efficacia delle interazioni tra uomo e macchina risulta quindi essenziale studiare e creare interfacce con la capacità di riconoscere tali segnali emotivi: nello specifico, in settori come l'intelligenza artificiale, la sanità o l'istruzione, lo sviluppo di modelli di riconoscimento emotivo avanzati potrebbe rivoluzionare l'esperienza utente, rendendo pertanto i sistemi più adattivi e reattivi agli stimoli forniti dall'utente stesso.

La disciplina che si occupa di ciò, nota come **Affective Computing**, si posiziona nell'intersezione tra informatica, psicologia e neuroscienze e, basandosi sull'utilizzo di algoritmi avanzati e tecniche di apprendimento automatico, ha come obiettivo quello di creare sistemi di riconoscimento emotivo automatico che siano efficienti, precisi e sicuri.

Il lavoro presentato in questa Tesi si inserisce in questo contesto, con l'obiettivo di sviluppare e valutare modelli di riconoscimento delle emozioni da segnali vocali attraverso l'analisi di dataset vocali, l'uso di tecniche di estrazione delle features e la classificazione automatica.

In particolare, per la risoluzione del problema è stata adottato un approccio che prevede l'applicazione della libreria Librosa per l'estrazione dai segnali vocali delle caratteristiche rilevanti e l'utilizzo di una rete neurale convoluzionale (**CNN**) per l'addestramento sulle caratteristiche estratte.

Il modello è stato addestrato e valutato sul dataset **CREMA-D**.

Diversi esperimenti sull'architettura e sul dataset sono stati portati avanti, e successivamente discussi in dettaglio, al fine di ottimizzare le prestazioni del sistema.

La Tesi è strutturata come segue:

Il Capitolo 1 presenta lo stato dell'arte, riportando in breve le strategie di risoluzione presenti in letteratura sul medesimo dataset ed i relativi risultati.

Il Capitolo 2 svolge un'analisi approfondita delle principali tecnologie adottate per il progetto, esaminando le architetture CNN, e più nello specifico le architetture **1D CNN**, descrivendo la strategia di tuning degli iperparametri **Bayesian Optimization** ed infine descrivendo l'utilità e l'utilizzo della **Data Augmentation**.

Il Capitolo 3 fornisce una panoramica dettagliata del dataset utilizzato, illustrandone le principali caratteristiche. Inoltre, presenta le librerie più rilevanti impiegate nel corso del lavoro, evidenziandone il ruolo e l'importanza nell'analisi.

Il Capitolo 4 delinea l'approccio tecnico adottato spiegando dettagliatamente l'architettura del progetto, a partire dall'estrazione delle caratteristiche e dalla creazione della rete neurale, fino allo studio e all'analisi delle varie sperimentazioni svolte, con particolare attenzione alle fasi di addestramento e di definizione delle metriche di valutazione.

Il Capitolo 5, infine, fornisce i risultati e la valutazione sperimentale, presentando le metriche di valutazione ed i grafici di perdita ed accuratezza.

Indice

Elenco delle figure

Capitolo 1

Stato dell'arte

Il riconoscimento delle emozioni del parlato (SER) è un campo di ricerca in espansione, che si concentra sull'identificazione delle emozioni umane attraverso segnali vocali. Questo settore ha applicazioni in vari ambiti, tra cui assistenti vocali, analisi del comportamento umano e interfacce uomo-macchina.

Il Deep Learning ha trasformato l'elaborazione del segnale vocale, portando molti ricercatori a ottenere risultati significativi nel SER grazie a metodi di apprendimento profondo.

La prima soluzione rilevante, elaborata da [1] nel 2015, propone un approccio innovativo che sfrutta la capacità delle DCNN (Deep Convolutional Neural Network) di apprendere automaticamente caratteristiche invarianti di alto livello dai dati grezzi per classificare le emozioni nel parlato. In particolare, la ricerca utilizza gli spettrogrammi come input per la DCNN, integrando la PCA (Principal Component Analysis) come tecnica di pre-elaborazione. Tra i passaggi fondamentali della fase preparatoria, è inclusa la tecnica del Whitening, strettamente connessa alla PCA, che trasforma i dati in un sistema di coordinate in cui le componenti risultano non correlate e con varianza unitaria. Il modello

proposto si basa su un'architettura DCNN composta da due strati convoluzionali, due strati di pooling e due strati completamente connessi. Valutato sul dataset IEMOCAP, il modello raggiunge un'accuratezza del 40%. È interessante notare che l'applicazione del Whitening contribuisce significativamente a migliorare le performance: il modello addestrato sugli spettrogrammi sbiancati supera quello addestrato sugli spettrogrammi grezzi, confermando l'efficacia della tecnica di pre-elaborazione.

Tra il 2016 e il 2017, altre ricerche continuano ad affinare questo approccio, sfruttando principalmente i dataset IEMOCAP e EmoDB per generare spettrogrammi come input ai modelli di classificazione. Questi studi evidenziano i vantaggi di utilizzare rappresentazioni 2D del segnale audio, poiché eliminano la necessità di una laboriosa feature engineering manuale, riducendo significativamente i tempi e le risorse richieste per l'elaborazione dei dati. Questo cambio di paradigma, dalla progettazione manuale delle caratteristiche all'apprendimento automatico delle stesse attraverso modelli come le DCNN, si conferma come una tendenza chiave per migliorare l'efficienza e le prestazioni nel riconoscimento delle emozioni.

Nello studio proposto da [2], vengono analizzate le rappresentazioni 2D dei segnali vocali generate tramite la STFT (Short-Time Fourier Transform). Il modello combina diverse architetture, includendo CNN, reti neurali ricorrenti (RNN) ed una loro integrazione innovativa. In particolare, le CNN sono utilizzate per apprendere modelli locali e trasformarli in caratteristiche astratte, mentre le RNN, nello specifico le LSTM (Long Short Term Memory), si dimostrano particolarmente efficaci nell'elaborazione di dati sequenziali e nella cattura di dipendenze a lungo termine. L'innovazione chiave dello studio è rappresentata dal concetto di Time-Distributed CNN, che combina l'abilità delle reti convoluzionali nell'estrarre caratteristiche con la capacità delle LSTM di modellare la di-

namica temporale. Questa sinergia permette di ottenere esiti migliori rispetto all'utilizzo isolato delle CNN o delle LSTM. I risultati sperimentali confermano l'efficacia dell'approccio combinato: mentre le CNN ottengono una precision dell'87.74%, una recall dell'86.32% e un f1-score dell'86.06%, e le LSTM registrano rispettivamente valori del 79.87%, 78.83% e 78.31%, l'approccio Time-Distributed CNN raggiunge una precision dell'88.01%, una recall dell'86.86% e un f1-score dell'86.65%. Questo rappresenta un miglioramento significativo, dimostrando l'efficacia della combinazione tra capacità di estrazione delle caratteristiche e modellazione temporale.

A differenza delle soluzioni precedenti, il metodo presentato da [3] adotta un approccio articolato che inizia segmentando il segnale audio in finestre di 3 secondi, per poi generare spettrogrammi a spaziatura lineare che preservano la struttura armonica del parlato. Queste rappresentazioni 2D vengono utilizzate come input di un'architettura che combina i punti di forza delle reti neurali convoluzionali (CNN) e delle reti LSTM (Long Short-Term Memory). Una delle principali innovazioni dello studio è l'introduzione di una tecnica di filtraggio armonico per rimuovere le componenti non vocali dallo spettrogramma. Questo approccio garantisce al sistema una maggiore resistenza al rumore, permettendo un riconoscimento accurato delle emozioni anche in ambienti con elevati livelli di rumore di fondo. I risultati sperimentali mostrano che il modello combinato CNN-LSTM raggiunge un'accuratezza massima del 68.8%, superando il modello basato esclusivamente sulla convoluzione, che si ferma al 66.1%. Un ulteriore contributo significativo dello studio è l'introduzione di un processo di predizione a due fasi per affrontare lo squilibrio del dataset IEMOCAP. Nella prima fase, il modello classifica l'input in una delle quattro classi di emozioni presenti nel dataset. Se l'emozione rilevata è 'neutrale', si attiva una seconda fase che utilizza tre predittori binari per confrontare la neutralità con ciascuna delle altre emozioni, perfezio-

nando così il risultato finale. Questo approccio, sebbene specifico per risolvere il problema dello squilibrio dei dati, consente al modello di raggiungere un'accuratezza complessiva del 67.3%.

La ricerca proposta da [4] utilizza reti neurali convoluzionali 1D (1D-CNN) combinate con LSTM per elaborare direttamente le clip audio grezze e apprenderne le caratteristiche in modo automatico. La rete è strutturata con 4 blocchi modulari LFLB (Local Feature Learning Block), seguiti da uno strato LSTM e uno strato completamente connesso (Fully Connected). Gli LFLB, che sostituiscono i classici strati convoluzionali, sono composti da uno strato convoluzionale, uno strato di normalizzazione batch, una funzione di attivazione ReLU (unità lineare esponenziale) e uno strato di max-pooling. Questa configurazione permette agli LFLB di apprendere correlazioni locali, normalizzare le attivazioni e creare rappresentazioni gerarchiche delle caratteristiche estratte. Gli strati LSTM, invece, sono responsabili della modellazione delle dipendenze a lungo termine. Le prestazioni del modello sono state valutate sui dataset EmoDB e IEMOCAP attraverso due esperimenti distinti: uno basato sull'analisi dipendente dall'interlocutore (speaker-dependent analysis) e l'altro su quella indipendente (speaker-independent analysis). Per il dataset EmoDB, i risultati mostrano un'accuratezza del 92.34% per l'analisi speaker-dependent e dell'86.73% per quella speaker-independent. Per il dataset IEMOCAP, invece, i valori di accuratezza sono del 67.92% e del 79.72%, rispettivamente.

La soluzione proposta da [5] analizza le clip audio provenienti dai dataset RAVDESS, IEMOCAP ed EmoDB. A differenza di molti approcci precedenti, che si basano sulla conversione dei dati audio in rappresentazioni visive o sull'uso diretto dei dati grezzi, questo studio utilizza una combinazione di cinque rappresentazioni spettrali: MFCCs (Mel-Frequency Cepstral Coefficients), Mel-Spectrogram, Chromagram, Spectral Contrast

e Tonnetz Representation. Questo approccio consente di catturare informazioni più complete del suono, incluse caratteristiche relative al timbro, alla tonalità e all'armonia. La ricerca propone e confronta cinque modelli di rete neurale convoluzionale ad una dimensione (1D-CNN). Il modello base, denominato modello A, è progettato per classificare sette emozioni. Il modello B rappresenta una variante del modello A, in cui le emozioni "disgusto" e "noia" sono state rimosse, poiché risultavano essere le più frequentemente confuse, migliorando così la precisione. Il modello C, invece, è costituito da sette classificatori binari, uno per ogni emozione. Il modello D combina tre classificatori: il primo è un classificatore binario per l'emozione "disgusto", seguito da un secondo classificatore per l'emozione "noia" e, infine, dal modello B per le restanti emozioni. Infine, il modello E unisce i modelli C e D. I dataset RAVDESS e IEMOCAP sono stati valutati esclusivamente con il modello A, adattato per ciascun dataset. I risultati ottenuti sono stati un'accuratezza del 71.61% per RAVDESS e del 64.3% per IEMOCAP. Per il dataset EmoDB, sono stati testati tutti i modelli, ottenendo i seguenti risultati: il modello A ha raggiunto un'accuratezza del 82.86%, il modello B del 96.34%, il modello C dell'82.4%, il modello D dell'84.76% e il modello E dell'86.1%. Questi risultati evidenziano che il modello B, con la rimozione delle emozioni più confuse, ha ottenuto le migliori performance sul dataset EmoDB.

La ricerca proposta da [6] presenta un sistema a due fasi per l'analisi del parlato, che comprende una fase di estrazione delle caratteristiche seguita da una fase di classificazione. L'obiettivo è catturare una rappresentazione ricca di informazioni emotive nel parlato, utilizzando un vettore di caratteristiche a 42 dimensioni che combina parametri consolidati come MFCCs (Mel Frequency Cepstral Coefficients), ZCR (Zero-Crossing Rate), TEO (Teager Energy Operator) e introduce anche l'HNR (Harmonic to Noise Rate). Per ridurre la dimensionalità di questo vettore, viene impiegato un AutoEncoder (AE), una tecnica

che consente di comprimere il vettore di input mantenendo le informazioni più rilevanti. In questo modo si ottimizzano sia l'efficienza computazionale che l'accuratezza durante la fase di classificazione. Lo studio esplora due tipologie di AutoEncoder: l'AE di base, che prevede un solo strato nascosto, e l'AE a stack, che utilizza due o più strati nascosti. Per la fase di classificazione, vengono testati diversi kernel di SVM (Support Vector Machine), tra cui lineari, polinomiali e a base radiale (RBF). La valutazione del modello è stata effettuata sul dataset RML (Ryerson Multimedia Laboratory), ottenendo un'accuratezza del 74.07% con l'AutoEncoder e del 65.43% senza la riduzione delle caratteristiche.

Un approccio simile è adottato da [7], che utilizza come input un vettore di cinque caratteristiche, tra cui MFCCs, ZCR, TEO, Chromagram STFT e RMS (Root Mean Square). Il modello proposto è composto da cinque strati CNN monodimensionali, che producono in output le probabilità associate a ciascuna delle otto emozioni analizzate nello studio. Un aspetto innovativo di questo approccio è il miglioramento della capacità del modello di generalizzare e di operare efficacemente anche in ambienti rumorosi. La valutazione del modello viene condotta su tre dataset distinti, con i seguenti risultati: 99% di accuratezza per il dataset TESS, 96% per il dataset RAVDESS e 84% per il dataset CREMA-D.

Lo studio più recente di [8] (2024) propone un approccio basato su un classificatore MLP (Multi-Layer Perceptron) per il riconoscimento delle emozioni nel parlato, focalizzandosi sul dataset CREMA-D. L'MLP, una rete neurale feedforward, viene utilizzato per categorizzare le emozioni nei campioni vocali. Per migliorare le capacità di addestramento e generalizzazione del modello, vengono adottate tecniche di Data Augmentation, come l'iniezione di rumore, lo stretching, lo shifting e il pitching. Il modello raggiunge un'accuratezza complessiva del 61% sui dati di test.

A differenza delle soluzioni precedentemente descritte, l'approccio proposto in questa

Tesi si basa esclusivamente su una rete neurale convoluzionale monodimensionale (1D-CNN), applicata agli audio del dataset CREMA-D. Da questi sono state estratte specifiche combinazioni di caratteristiche, tra cui Mel-Frequency Cepstral Coefficients, Chroma, Mel-Spectrogram, Spectral Contrast, Tonnetz Representation, Zero-Crossing Rate, Harmonics-to-Noise Ratio, Short-Time Energy, Teager Energy Operator, Power Spectral Density, Jitter e Shimmer. Il modello è stato inizialmente addestrato su un primo set di cinque caratteristiche e successivamente su un set più ampio di dieci, in un esperimento condotto su sei emozioni, considerato come punto di partenza della sperimentazione. In seguito, per affrontare alcune criticità emerse nella valutazione iniziale, sono state sviluppate tre soluzioni alternative, basate su diverse combinazioni di caratteristiche o sull'esclusione delle emozioni risultate più ambigue.

Capitolo 2

Principali Tecnologie

Questo Capitolo esplora la struttura, gli elementi costitutivi e il funzionamento generale delle CNN, con particolare attenzione alle 1D-CNN. Verranno in seguito trattate le principali tecniche per la ricerca degli iperparametri ottimali, concentrandosi sull'ottimizzazione bayesiana. Inoltre, si discuterà l'importanza e l'applicazione della data augmentation nelle reti neurali, sottolineando come queste tecniche combinano le loro potenzialità per ottenere modelli più precisi ed efficienti.

2.1 Reti Neurali

Nel campo del Deep Learning, una **rete neurale artificiale** (ANN), anche nota semplicemente come rete neurale (NN), è un modello matematico ispirato al funzionamento del cervello umano composto da unità artificiali connesse tra loro, chiamate **neuroni**. I neuroni elaborano le informazioni contenute nella propria memoria locale oppure negli stimoli provenienti dall'esterno o da altri neuroni presenti nella rete. Ognuno di essi è connesso ai nodi dello strato successivo ed il loro valore di output diventa il valore di input per quelli

seguenti. Ogni connessione tra neuroni è associata a un peso (**weight**), un parametro che viene modificato durante l'addestramento. Il peso determina l'influenza di un input sul neurone successivo, secondo la formula:

$$output = (inputs \cdot weights) + bias$$

Oltre ai pesi, ogni neurone dispone di un **bias**, un valore costante aggiunto al calcolo basato sui pesi. Anche il bias viene ottimizzato durante l'addestramento: il suo ruolo è quello di spostare la funzione di attivazione nel piano dimensionale, migliorando la capacità del modello di adattarsi ai dati.

Generalmente i pesi vengono inizializzati casualmente, mentre i bias vengono impostati a zero. Durante l'addestramento, la rete aggiusta costantemente questi parametri al solo scopo di migliorare le proprie prestazioni. Questo processo avviene principalmente negli strati nascosti (**hidden layers**), che si trovano tra lo strato di input e quello di output. Questi strati sono detti tali perché il loro funzionamento interno non è direttamente osservabile dall'esterno.

Le reti neurali sono composte da livelli di nodi organizzati in:

- **Strato di input**, che riceve i dati grezzi.
- **Strati nascosti**, dove avviene la maggior parte delle elaborazioni.
- **Strato di output**, che produce il risultato finale.

Oltre al peso, ad ogni nodo è anche associata una soglia: un valore minimo che l'output deve superare per *attivare* il neurone, ossia per permettere la trasmissione dei dati tra i vari livelli; se l'output non supera la soglia, l'informazione non viene trasmessa al livello successivo.

Quando la soglia viene superata, si genera uno stato interno, chiamato **attivazione**, che rappresenta il risultato del calcolo effettuato sui dati in ingresso. Questo stato viene poi trasformato in un **segnale di risposta** tramite la **funzione di attivazione**, che determina l'intensità della risposta del neurone. Il segnale si propaga attraverso le connessioni, inviando l'informazione ai neuroni dello strato successivo. Questo processo continua fino a raggiungere lo strato di output, dove viene prodotto il risultato finale, reso disponibile all'esterno. Le funzioni di attivazione più comuni includono la **sigmoide**, la **tangente iperbolica (tanh)** e la **ReLU (Rectified Linear Unit)**, ognuna con caratteristiche specifiche che influenzano il comportamento del modello.

Al di là della funzione di attivazione, un altro aspetto cruciale nell'addestramento di una rete neurale riguarda i tipi di apprendimento che possono essere applicati. L'apprendimento in questo campo può essere diviso in tre categorie principali: supervisionato, non supervisionato e per rinforzo.

- Nel **supervised learning**, l'obiettivo è quello di prevedere un valore di uscita sulla base di osservazioni sui dati in ingresso. A tal fine, vengono utilizzati set di dati etichettati, cioè dove ogni esempio di input è associato a un'etichetta di uscita corrispondente. Il modello impara quindi a mappare gli input agli output in modo che possa fare previsioni anche su nuovi dati non visti durante l'addestramento, risolvendo così problemi di classificazione o regressione.

- Nel **unsupervised learning**, il modello non ha accesso a etichette di uscita. Invece, si concentra sull'analisi dei dati in ingresso per identificare pattern o raggruppamenti nascosti. Gli algoritmi in questo caso cercano di scoprire la struttura sottostante dei dati attraverso operazioni come clustering o riduzione della dimensionalità, senza la necessità di etichette.

- Il **reinforcement learning**, invece, si distingue significativamente dagli altri tipi di apprendimento. In questo caso, un agente impara a prendere decisioni interagendo con un ambiente dinamico: l'apprendimento quindi si basa su un meccanismo di tentativi ed errori. L'agente esplora una serie di azioni e riceve feedback in forma di ricompense o penalità, che lo guidano a migliorare il proprio comportamento. L'obiettivo dell'agente è quello di massimizzare la ricompensa totale nel tempo, bilanciando l'esplorazione di nuove azioni con lo sfruttamento di quelle già conosciute che portano ai risultati migliori. Questo tipo di apprendimento è un processo continuo di interazione con l'ambiente e di adattamento, dove ogni azione e il relativo feedback contribuiscono a migliorare la strategia complessiva.

2.2 Convolutional Neural Networks (CNN)

Le **reti neurali convoluzionali (CNN)** sono una tipologia di reti neurali ampiamente utilizzate nel campo del Deep Learning. La loro forza risiede nell'efficienza nel riconoscimento di pattern e nella gestione di grandi quantità di dati complessi.

Il nome deriva dall'operazione matematica di convoluzione, un elemento chiave del loro funzionamento. Questa operazione può essere paragonata al lavoro di uno scanner: il suo risultato è una funzione che rappresenta la sovrapposizione tra due funzioni, calcolata spostando una sull'altra nel tempo o nello spazio.

2.2.1 Caratteristiche principali

A differenza delle reti neurali tradizionali, le CNN utilizzano pesi e bias condivisi tra i neuroni di uno stesso strato: ciò implica che tutti i neuroni nascosti in un layer rilevano la medesima caratteristica, ma in posizioni diverse dell'input. Questo approccio riduce signi-

ficativamente il numero dei parametri da apprendere rendendo, così, la rete più efficiente e riducendo il rischio di overfitting. Allo scopo di ottimizzare le prestazioni, i valori dei pesi vengono regolati ad ogni strato.

Inoltre, le CNN sono in grado di identificare e riconoscere pattern rilevanti analizzando piccole porzioni di dati, che vengono successivamente generalizzati e ricercati in posizioni diverse dell'input. Questa caratteristica prende il nome di località ed invarianza spaziale e rende le reti particolarmente efficaci nell'estrazione di informazioni significative da dati strutturati, garantendo una maggiore robustezza e flessibilità nell'elaborazione.

Infine, risulta fondamentale che le CNN siano molto profonde e presentino centinaia di strati: ciò consente alla rete di apprendere un'ampia gerarchia di caratteristiche, partendo da quelle più semplici fino ad arrivare a quelle più complesse.

2.2.2 Struttura

Le CNN sono costituite da diversi tipi di strati, ognuno dei quali svolge un ruolo specifico. La struttura è così suddivisa:

1. Strati convolutivi;
2. Strati di pooling;
3. Strati completamenti connessi.

2.2.2.1 Strati convolutivi

Lo strato convolutivo è il primo livello, l'elemento costitutivo principale ed il punto in cui si verifica la maggior parte dei calcoli di una rete CNN. In questo livello vengono utilizzati dei **filtri (o kernel)**, ovvero delle piccole finestre che analizzano solo una porzione di dati alla volta, che scorrono su tutti i dati e producono una **mappa delle caratteristiche** (in

inglese **feature map**) che rappresenta la risposta di ogni filtro in relazione ad una regione specifica dell'input.

Ogni livello di convoluzione iniziale può essere seguito da un altro livello di convoluzione, quando ciò si verifica, la struttura della CNN può diventare gerarchica.

Dopo ogni convoluzione viene applicata una funzione di attivazione, come ad esempio la ReLU, che introduce non linearità nel processo di apprendimento, consentendo alla rete di apprendere rappresentazioni più complesse e varie degli input.

A livello implementativo, nella definizione dei layer convoluzionali vengono istanziati tre particolari iperparametri che vanno a condizionare considerevolmente la dimensione dell'output. Tali iperparametri sono il **numero di filtri**, lo **stride** e il **padding**. Il numero dei filtri definisce quante diverse caratteristiche il modello può estrarre dall'input in ogni layer convoluzionale influendo, così, sulla profondità dell'output. Lo stride, invece, è la distanza con cui il filtro si sposta attraverso l'input durante l'operazione di convoluzione. Un valore di stride maggiore implica che il filtro salta più campioni, mentre uno stride minore fa sì che il filtro si sposti più lentamente, coprendo piccole porzioni di input per ogni operazione. D'altra parte, infine, il padding viene adoperato per aggiungere valori extra ai bordi dell'input. Il suo scopo principale è garantire che i filtri possano essere applicati in modo efficace su tutta l'area dell'input evitando la perdita di informazioni durante l'operazione di convoluzione. Esistono vari tipi di padding, ma quelli maggiormente utilizzati sono:

- **Padding valido**, con il quale non viene aggiunto alcun padding ai bordi. La dimensione dell'output diminuisce poiché il filtro viene applicato solo a porzioni del segnale che hanno una dimensione sufficiente per eseguire correttamente l'operazione di convoluzione completa.

- **Stesso padding**, che ha come obiettivo quello di mantenere la stessa dimensione dell'output rispetto all'input.

- **Padding zero**, che semplicemente imposta a zero tutti gli elementi che ricadono al di fuori della matrice di input, producendo un output più grande o di dimensioni uguali. Viene utilizzato, per lo più, per evitare la perdita di informazioni sui bordi.

In sintesi, lo scopo dei livelli convolutivi è quello di estrarre le caratteristiche più significative dai dati in ingresso preservando, ad ogni modo, le informazioni spaziali o temporali.

2.2.2.2 Strati di pooling

A seguito di un layer di convoluzione vengono introdotti gli strati di pooling, definiti anche di sottocampionamento. Questi livelli eseguono una riduzione dimensionale delle mappe di caratteristiche, che saranno poi l'input dei layer successivi, rendendo così la rete meno sensibile alla posizione delle caratteristiche estratte. In modo simile al livello precedente, l'operazione di pooling utilizza un filtro che scorre sull'intero input. Tuttavia questo filtro non ha pesi da apprendere, ma applica una funzione di aggregazione ai valori compresi nel suo campo ricettivo, generando i valori che andranno a comporre l'array di output. Questa operazione viene svolta in due principali maniere:

- **Average pooling**: quando la finestra-filtro si sposta sull'input, viene calcolato il valore medio all'interno del campo in analisi che viene poi trasmesso all'array di output. Questo approccio di pooling risulta efficiente quando si presenta la necessità di ridurre il rumore nei dati e pertanto rende il modello più robusto agli errori minori.

- **Max pooling**: quando viene applicato sull'input, il filtro seleziona il valore massimo nella regione in analisi e lo invia all'array di output. In genere questa tipologia di pooling è quella maggiormente adoperata per via della sua capacità nel conservare le informazioni più importanti.

In generale, sebbene l'operazione di pooling comporti la perdita di molte informazioni, offre una serie di vantaggi alla rete neurale come, ad esempio, ne riduce la complessità, ne migliora l'efficienza e limita il rischio di adattamento alla composizione e posizione dei dati.

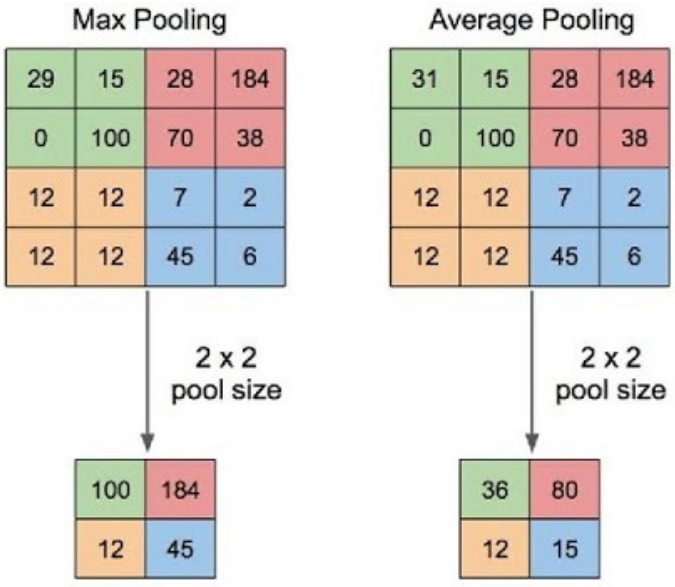


Figura 2.1: Differenza tra Max Pooling ed Average Pooling

2.2.2.3 Strati completamente connessi

Verso la fine della rete, dopo vari strati convoluzionali e di pooling, le mappe di caratteristiche vengono appiattite in un vettore unidimensionale, secondo un'operazione chiamata

flattening, e passate attraverso uno o più strati completamente connessi, che funzionano come una rete neurale tradizionale. Il nome di questo livello ne descrive le caratteristiche: difatti, i layer fully connected possono essere visti come dei vettori di neuroni. L'output di ciascun neurone è inviato a tutti i neuroni del layer successivo, con diversi valori di peso. A questo punto, il modello apprende le combinazioni di alto livello delle caratteristiche estratte dai filtri per effettuare la fase finale, ovvero una classificazione o una regressione. A differenza degli strati precedentemente descritti, nei livelli completamente connessi viene solitamente utilizzata la funzione di attivazione **softmax**, per la classificazione multiclasse, che genera una distribuzione di probabilità sulle diverse categorie. Questa funzione consente di classificare gli input in modo appropriato, assegnando a ciascuna classe una probabilità compresa tra 0 e 1. Viene quindi generato un vettore di dimensione K che contiene le probabilità per ciascuna classe: K, in questo caso, indica il numero di classi prevedibili.

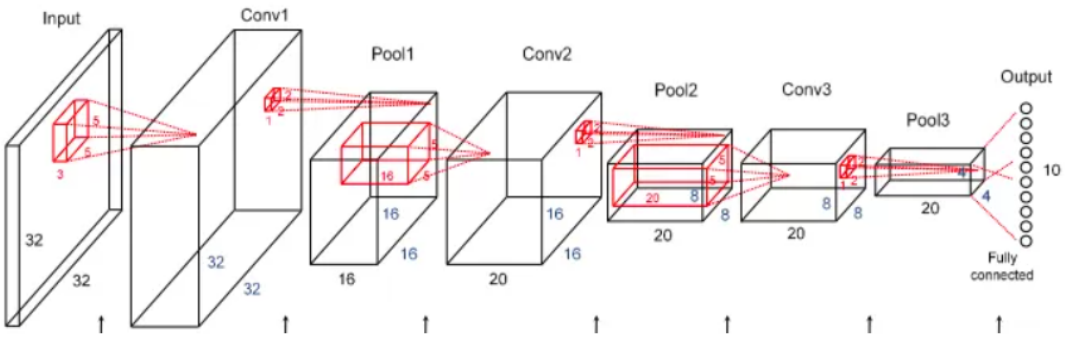


Figura 2.2: Struttura CNN

2.2.3 1D-CNN

Una **rete neurale convoluzionale ad una dimensione (1D-CNN)** è una variante della CNN progettata specificamente per elaborare dati sequenziali monodimensionali, come serie temporali o dati di testo. Grazie a questa loro capacità di elaborare dati sequenziali in modo rapido ed efficace, le 1D-CNN trovano applicazione in numerosi settori, tra cui:

- **Riconoscimento delle emozioni da segnali vocali**: analisi di feature acustiche per distinguere le emozioni.
- **Elaborazione di segnali audio**: riconoscimento vocale e classificazione di eventi acustici.
- **Analisi di serie temporali**: previsione finanziaria, monitoraggio sensoriale e rilevamento anomalie.
- **Medicina e bioinformatica**: diagnosi da segnali EEG/ECG.
- **NLP**: classificazione di testi e sentiment analysis.
- **Industria e cybersecurity**: rilevamento guasti in impianti, monitoraggio interferenze elettromagnetiche e sicurezza informatica.

2.2.3.1 Struttura

La struttura generale delle reti neurali convoluzionali 1D è simile a quella delle CNN tradizionali, con la principale differenza che le operazioni di convoluzione e pooling avvengono lungo un'unica dimensione. Gli altri strati, invece, rimangono invariati.

L'elemento chiave di una 1D-CNN, quindi, è lo **strato convoluzionale monodimensionale (Conv1D)**, in cui i kernel scorrono lungo la sequenza di ingresso, calcolando il

prodotto scalare tra il filtro e i dati in ogni posizione. Questo processo genera una feature map che mette in evidenza schemi o caratteristiche locali tra elementi adiacenti.

Successivamente, lo **strato di pooling (MaxPooling1D o AveragePooling1D)** riduce la dimensione dell'output, selezionando le informazioni più rilevanti e migliorando l'efficienza computazionale del modello.

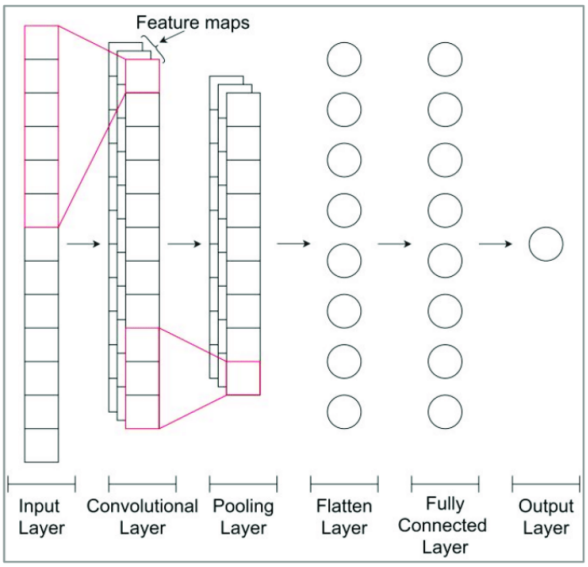


Figura 2.3: Struttura 1D-CNN

2.2.3.2 Vantaggi e limitazioni

Nel contesto del Deep Learning, la selezione della rete neurale più adatta alla natura dei dati è fondamentale per ottenere risultati significativi. Nel caso delle reti neurali convoluzionali 1D, queste offrono diversi vantaggi, ma presentano anche alcune limitazioni che devono essere considerate.

Un primo vantaggio importante di queste reti è la loro efficienza computazionale: le operazioni di convoluzione su una sola dimensione riducono il numero di parametri rispetto alle reti 2D, rendendo queste architetture particolarmente adatte per applicazioni in tempo reale, dove la velocità di elaborazione è cruciale. Inoltre, risultano essere molto versatili, poiché sono in grado di rilevare automaticamente pattern significativi in vari tipi di segnali monodimensionali, come ad esempio quelli audio o temporali. Questa capacità di individuare le caratteristiche rilevanti in modo autonomo le rende particolarmente utili in compiti complessi. Le 1D-CNN sono anche flessibili, in quanto possono essere integrate con altre architetture, come le RNN, LSTM o Transformer. Questi modelli ibridi permettono di migliorare la capacità di apprendere le dipendenze a lungo termine, risolvendo in parte i limiti delle reti 1D-CNN pure.

Tuttavia, presentano anche alcune limitazioni: in particolare, richiedono dati strutturati e non sono altrettanto flessibili quando si trattano sequenze di lunghezza variabile. Inoltre, possono avere difficoltà nel catturare dipendenze a lungo termine, specialmente quando la sequenza di dati è molto lunga. In questi casi, le informazioni di contesto potrebbero andare perse a causa della natura locale della convoluzione. Infine, le loro performance sono fortemente influenzate dalla scelta degli iperparametri, come il tipo di filtri, la dimensione del kernel e lo stride: una selezione inadeguata di questi parametri può compromettere l'efficacia del modello in maniera significativa.

2.3 Ottimizzazione degli iperparametri

L'**ottimizzazione degli iperparametri** è il processo di selezione dei valori ottimali per gli iperparametri di un modello di Deep Learning, al fine di migliorare le sue prestazioni e la sua capacità di generalizzazione. Gli iperparametri sono variabili di configurazione

esterne che governano il processo di addestramento e l'architettura del modello e devono essere impostati prima dell'inizio dell'addestramento. Poiché ogni set di dati ed ogni modello necessitano di un diverso insieme di iperparametri, la loro selezione è un processo sperimentale. Questa operazione può essere effettuata manualmente o attraverso metodi automatici, eseguendo più prove con diversi valori e analizzando i risultati sulla base di metriche come l'accuratezza o la funzione di perdita. La cross-validation è spesso utilizzata per stimare le prestazioni di generalizzazione del modello e scegliere la configurazione ottimale.

Il suo obiettivo è quello di trovare un equilibrio ottimale tra bias e varianza. Il bias rappresenta la discrepanza tra le previsioni del modello e la realtà: un modello con un bias elevato fatica a individuare le relazioni chiave nei dati, risultando poco accurato. La varianza, invece, misura la sensibilità del modello ai nuovi dati: un modello con varianza elevata è eccessivamente complesso e si adatta troppo ai dati di addestramento, compromettendo la sua capacità di generalizzare su dati non visti. Un modello ideale, quindi, deve avere un bias sufficientemente basso per essere accurato e una varianza contenuta per garantire coerenza nei risultati. Una buona ottimizzazione degli iperparametri consente di raggiungere questo equilibrio, migliorando le prestazioni del modello ed ottimizzando l'uso delle risorse computazionali durante l'addestramento.

Di seguito vengono elencati alcuni degli iperparametri più importanti per l'addestramento delle reti neurali:

- **Tasso di apprendimento**: controlla la velocità con cui un modello aggiorna i propri pesi durante l'addestramento. Un valore elevato accelera l'apprendimento, ma può rendere instabili le prestazioni o impedire al modello di convergere verso una soluzione ottimale. Un valore troppo basso, invece, garantisce una maggiore stabilità

ma richiede più tempo per l'addestramento.

- **Decadimento del tasso di apprendimento**: imposta la velocità con cui il tasso di apprendimento di una rete diminuisce nel tempo, consentendo al modello di apprendere più rapidamente.

- **Dimensione del batch**: rappresenta il numero di campioni utilizzati per aggiornare i pesi in ogni iterazione. Un batch piccolo può rendere l'ottimizzazione più instabile ma favorisce una maggiore generalizzazione, mentre un batch più grande migliora la stabilità e l'efficienza computazionale ma può ridurre la capacità del modello di adattarsi ai dati.

- **Numero di hidden layers**: determina la profondità della rete neurale. Una rete più profonda è in grado di apprendere rappresentazioni più complesse, ma aumenta anche il rischio di overfitting e il costo computazionale.

- **Numero di neuroni per livello**: definisce l'ampiezza della rete neurale. Un numero maggiore di nodi può migliorare la capacità del modello di apprendere relazioni complesse, ma richiede più risorse computazionali e aumenta il rischio di overfitting se non regolato correttamente.

- **Momentum**: aiuta a stabilizzare l'ottimizzazione, riducendo le oscillazioni nella discesa del gradiente. Permette al modello di mantenere una direzione costante negli aggiornamenti dei pesi, migliorando la velocità di convergenza e riducendo il rischio di rimanere bloccato in minimi locali.

- **Epoche**: rappresentano il numero di volte in cui l'intero set di dati di addestramento viene presentato alla rete durante l'addestramento. Un numero troppo basso potreb-

be impedire al modello di apprendere adeguatamente, mentre un numero troppo alto potrebbe causare overfitting.

- **Funzione di attivazione**: introduce non linearità nel modello, permettendogli di apprendere rappresentazioni complesse dei dati.

Esistono diverse tecniche di ottimizzazione degli iperparametri per svolgere il processo di regolarizzazione:

- **Grid Search**: In questo metodo si definisce un insieme di iperparametri e una metrica di valutazione delle prestazioni. L'algoritmo esplora sistematicamente tutte le possibili combinazioni degli iperparametri specificati, addestrando e valutando un modello per ciascuna configurazione. I modelli vengono confrontati in base alle loro prestazioni, e quello con i risultati migliori viene selezionato per l'addestramento finale. Sebbene questa tecnica garantisca l'individuazione della configurazione ottimale, risulta poco efficiente dal punto di vista computazionale, poiché richiede di testare un numero elevato di combinazioni.

- **Random Search**: Questa tecnica è simile alla Grid Search ma, anziché esplorare tutte le possibili combinazioni di iperparametri, seleziona casualmente un sottoinsieme di configurazioni ad ogni iterazione. Durante le diverse iterazioni, i modelli vengono confrontati tra loro per identificare quello con la combinazione di iperparametri che offre le migliori prestazioni. Rispetto alla precedente, la Random Search risulta più veloce ma allo stesso tempo meno precisa, poiché potrebbe non riuscire a trovare la configurazione ottimale a causa della selezione casuale degli iperparametri.

- **Bayesian Optimization**: si distingue dalle due tecniche precedenti perché sfrutta le valutazioni passate per guidare la ricerca verso le configurazioni più promettenti,

accelerando così il processo di ottimizzazione. Al contrario, Random Search e Grid Search esplorano lo spazio degli iperparametri in modo uniforme e indipendente dai risultati ottenuti nelle iterazioni precedenti.

2.3.1 Ottimizzazione Bayesiana

L'**ottimizzazione bayesiana** è una tecnica basata sul **teorema di Bayes** per trovare la giusta combinazione di iperparametri. Questo teorema consente di calcolare la probabilità che un determinato evento accada data la probabilità che un altro evento correlato al primo avvenga. È possibile, quindi, combinare due eventi secondo la seguente formula:

$$P(A \cap B) = P(A|B) \cdot P(B)) = P(B|A) \cdot P(A)$$

Da cui si ricava che:

$$P(A|B) = P(B|A) \cdot P(A)/P(B)$$

Dove:

- P(A) è la funzione di densità di probabilità dell'evento A;
- P(B) è la funzione di densità di probabilità dell'evento B;
- P(A|B) è la probabilità che l'evento A accada dopo che è accaduto l'evento B;
- P(B|A) è la probabilità che l'evento B accada dato che è accaduto A.

N.B. P(A|B) e P(B|A) vengono dette probabilità condizionate.

Quando questo teorema viene applicato, l'algoritmo costruisce un modello probabilistico basato su un insieme di iperparametri con l'obiettivo di ottimizzare una specifica metrica. Ad ogni iterazione, sulla base dei risultati dei test precedenti, l'ottimizzazione

bayesiana seleziona in modo probabilistico un nuovo set di iperparametri che ha maggiori probabilità di migliorare le prestazioni del modello. Poiché risulta essere più efficiente rispetto alla funzione obiettivo originale, questo modello probabilistico, noto come **surrogato della funzione obiettivo**, viene aggiornato e affinato ad ogni nuova valutazione. Man mano che il surrogato diventa più accurato nella previsione dei migliori iperparametri, il processo di ottimizzazione diventa più rapido, riducendo il numero di test necessari. Grazie a questo approccio strategico, l'ottimizzazione bayesiana è significativamente più efficiente rispetto ad altri metodi, evitando di testare combinazioni di iperparametri poco promettenti e concentrando le risorse sulle configurazioni più vantaggiose.

2.3.1.1 Fasi

L'operazione di ottimizzazione bayesiana si sviluppa attraverso diverse fasi, ognuna delle quali è finalizzata a individuare la configurazione ottimale di parametri da valutare. Il processo è così composto:

1. **Inizializzazione**: Si parte con una fase iniziale di esplorazione per raccogliere dati sull'andamento della funzione obiettivo in diverse regioni dello spazio degli iperparametri. Questo passaggio aiuta a ottenere un set di punti di partenza utili per costruire il modello surrogato.

2. **Costruzione del modello surrogato**: Utilizzando i dati raccolti, ovvero le coppie (*iperparametro, punteggio della funzione obiettivo)*, si costruisce un modello probabilistico che approssima la funzione obiettivo reale.

3. **Selezione del prossimo punto da esplorare**: Viene utilizzata una **funzione di acquisizione** per selezionare l'insieme di iperparametri x* da esplorare successivamente. La funzione di acquisizione ha il compito di bilanciare l'esplorazione e lo

sfruttamento. L'esplorazione consiste nel selezionare quegli iperparametri per i quali il modello ha una maggiore incertezza riguardo ai risultati, mentre per sfruttamento si intende la selezione di quegli iperparametri che hanno maggiore probabilità di migliorare la funzione obiettivo.

4. **Valutazione della funzione obiettivo**: Una volta selezionato l'insieme di iperparametri x*, la funzione obiettivo viene valutata in questo punto e ne viene registrato il punteggio. Questo passaggio è fondamentale per determinare l'efficacia della configurazione degli iperparametri selezionata.

5. **Aggiornamento del modello surrogato**: Dopo aver ottenuto il risultato della valutazione della funzione obiettivo, il modello surrogato viene aggiornato con i nuovi dati. Questo passaggio consente al modello di affinare le sue previsioni e migliorare la selezione dei futuri iperparametri.

6. **Conclusione**: I passaggi precedenti vengono eseguiti in modo iterativo e il processo prosegue finché non si raggiunge il numero massimo di iterazioni o finché non viene trovato un valore ottimale soddisfacente.

2.3.1.2 Vantaggi

Grazie alla sua capacità di bilanciare esplorazione e sfruttamento, la Bayesian Optimization viene ritenuta una delle tecniche più avanzate per l'ottimizzazione degli iperparametri. Ecco i principali vantaggi di questa metodologia:

- **Efficienza**: Riduce significativamente il numero di valutazioni della funzione obiettivo necessarie per trovare una soluzione ottimale.

- **Scalabilità**: È applicabile anche a problemi con spazi di ricerca ad alta dimensionalità.
- **Adattabilità**: Si adatta dinamicamente alle informazioni raccolte durante il processo, migliorando progressivamente la ricerca.
- **Robustezza**: È resistente al rumore nei dati e alla presenza di ottimi locali, riuscendo a trovare soluzioni di alta qualità anche in ambienti complessi.

2.4 Data augmentation

I modelli di Deep Learning sfruttano grandi quantità di dati diversificati per generare previsioni accurate in vari contesti. In questo ambito, la **data augmentation** riveste un ruolo fondamentale. Con il termine data augmentation si fa riferimento a un insieme di tecniche che mirano a manipolare e trasformare i dati preesistenti, ampliando così la dimensione del dataset originale. L'obiettivo di queste tecniche è migliorare l'ottimizzazione e la capacità di generalizzazione del modello. È importante sottolineare che i nuovi campioni generati mantengono la stessa etichetta o classe del dato originale.

Esistono diversi metodi di data augmentation, e le tecniche specifiche da utilizzare dipendono dalla natura dei dati con cui si sta lavorando. È importante notare che la data augmentation viene solitamente applicata durante la fase di pre-elaborazione del set di dati di addestramento.

2.4.1 Funzionamento

L'aumento dei dati consiste nel trasformare, modificare o editare i dati esistenti per generare varianti. Di seguito viene fornita una panoramica del processo:

- **Esplorazione del set di dati**: La prima fase dell'aumento dei dati prevede l'analisi di un set di dati esistente per comprenderne le caratteristiche principali. In base al tipo di dati e agli obiettivi desiderati, vengono selezionate le tecniche di aumento più adatte.

- **Aumento dei dati esistenti**: Una volta individuata la tecnica più adatta, si applicano le trasformazioni sui dati. Durante questo processo, le regole di etichettatura vengono mantenute, assicurando che i dati aumentati conservino le stesse etichette dei dati originali, per garantire la coerenza.

- **Integrazione dei dati**: In questa fase, i dati aumentati vengono combinati con i dati originali, creando un set di dati di addestramento più ampio per il modello.

2.4.2 Vantaggi

L'adozione della tecnica di data augmentation offre numerosi vantaggi, contribuendo in modo significativo a migliorare le performance dei modelli e a superare le difficoltà legate alla raccolta di un numero adeguato di dati. Di seguito vengono esaminati i principali benefici di questa tecnica:

- **Miglioramento delle prestazioni**: Le tecniche di aumento dei dati arricchiscono il set di dati esistente creando numerose variazioni. Ciò consente di disporre di un dataset più ampio per l'addestramento, esponendo il modello a una maggiore diversità di caratteristiche. I dati aumentati favoriscono una migliore generalizzazione su dati mai visti prima, migliorando le prestazioni del modello in scenari reali.

- **Riduzione della dipendenza dai dati**: La raccolta e la preparazione di grandi volumi di dati per l'addestramento sono spesso costose e dispendiose in termini di

tempo. Le tecniche di aumento dei dati rendono più efficaci set di dati di dimensioni contenute, riducendo così la necessità di dataset ampi per l'addestramento.

- **Prevenzione dell'overfitting**: L'aumento dei dati contribuisce a ridurre il rischio di overfitting durante l'addestramento. L'overfitting si verifica quando un modello si adatta troppo ai dati di addestramento, rendendo difficoltosa la generalizzazione a nuovi dati. Con un set di dati più ampio e vario, l'aumento dei dati aiuta il modello a evitare questo problema, migliorando la sua capacità di affrontare situazioni più generali e diversificate.

- **Protezione della privacy**: Quando si lavora con dati sensibili, le tecniche di aumento possono essere utilizzate per generare dati che preservano le caratteristiche statistiche degli originali. Questo approccio consente di addestrare modelli di Deep Learning su dati modificati, proteggendo la privacy degli individui e limitando l'accesso alle informazioni originali.

Capitolo 3

Dataset e Librerie

Il Capitolo 3 di questa Tesi offre un'analisi dettagliata del dataset e delle librerie utilizzate per lo sviluppo del modello.

3.1 Dataset

Il dataset impiegato nello studio è il **CREMA-D** (Crowd Sourced Emotional Multimodal Actors Dataset), che contiene 7.442 clip registrate da 91 attori (48 uomini e 43 donne) di età compresa tra 20 e 74 anni.

Gli attori provengono da un'ampia varietà di origini etniche, tra cui afroamericana, asiatica, caucasica, ispanica ed etnia non specificata. Questa diversità etnica contribuisce alla rappresentatività e all'universalità del dataset.

La suddivisione degli attori per età ed etnia è visualizzata in Figura 3.1.

Le clip includono sei emozioni: **rabbia, disgusto, paura, felicità, neutralità e tristezza**, espresse in quattro livelli di intensità: **basso, medio, alto e non specificato**.

Nelle registrazioni, gli attori recitano **12 frasi** dal contenuto semanticamente neutrale.

Età	# attori
20-29 anni	34
30-39 anni	23
40-49 anni	16
50-59 anni	12
60-69 anni	5
Oltre 70 anni	1

Razza \ Etnia	Non ispanici	Ispanici	Totale
Caucasica	53	8	**61**
Afro-Americana	21	1	**22**
Asiatica	7	0	**7**
Non specificata	0	1	**1**
Totale	**81**	**10**	**91**

Figura 3.1: Generalità degli attori

Le frasi originali in inglese, con le relative traduzioni in italiano, sono riportate nella Tabella 3.1.

Originale	Traduzione
It's 11 o'clock.	Sono le undici.
That is exactly what happened.	È esattamente quello che è successo.
I'm on my way to the meeting.	Sto andando alla riunione.
I wonder what this is about.	Mi chiedo di cosa si tratti.
The airplane is almost full.	L'aereo è quasi pieno.
Maybe tomorrow it will be cold.	Forse domani farà freddo.
I would like a new alarm clock.	Vorrei una nuova sveglia.
I think I have a doctor's appointment.	Credo di avere un appuntamento dal medico.
Don't forget a jacket.	Non dimenticare una giacca.
I think I've seen this before.	Penso di aver già visto questo.
The surface is slick.	La superficie è scivolosa.
We'll stop in a couple of minutes.	Ci fermeremo tra un paio di minuti.

Tabella 3.1: Frasi Dataset CREMA-D

Le sessioni di registrazione, della durata media di tre ore, si sono svolte in un ambiente insonorizzato, con illuminazione professionale. Gli attori erano seduti davanti a uno schermo verde per facilitare la post-produzione. Le clip finali sono state estratte manualmente

dai video grezzi e, successivamente, convertite in formato MP4.

Sebbene il dataset originario sia multimodale (audio-video), questo studio esamina esclusivamente lo stream audio, senza, però, analizzare le differenze tra le clip per livelli di intensità.

3.2 Librerie

Per l'elaborazione dei dati precedentemente menzionati, è stato fatto ampio uso delle librerie offerte da Python, come illustrato in Figura 3.2. In particolare, un ruolo fondamentale è stato svolto da **Librosa** che si è rivelata essenziale per l'analisi e l'estrazione delle caratteristiche audio.

```
# Importazione delle librerie necessarie

import os
import matplotlib.pyplot as plt
import librosa
import librosa.display
import numpy as np
import scipy.stats as stats
import pandas as pd
import seaborn as sns
from sklearn.utils import shuffle
from collections import Counter
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import OneHotEncoder
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

Figura 3.2: Librerie impiegate

3.2.1 Keras

Keras è una libreria open-source per il Deep Learning scritta in Python, progettata per semplificare la creazione, l'addestramento e la valutazione delle reti neurali. Il suo obiettivo

principale è consentire una sperimentazione rapida, riducendo la complessità nella costruzione dei modelli grazie a un'interfaccia chiara e intuitiva. Grazie alla sua modularità, permette agli sviluppatori di assemblare facilmente diversi strati di rete neurale, sia in modo sequenziale che attraverso architetture più avanzate. Questa semplicità non solo facilita l'apprendimento per i principianti, ma consente anche agli esperti di testare rapidamente nuove configurazioni.

Keras non si limita solo alla creazione e addestramento dei modelli, ma copre l'intero flusso di lavoro del Machine Learning, dall'elaborazione dei dati fino all'ottimizzazione degli iperparametri ed alla distribuzione del modello. Questa caratteristica lo rende una piattaforma completa per chi lavora con il Deep Learning, sia a scopo di ricerca che in produzione. La sua estensibilità permette di personalizzare i modelli con layer, funzioni di attivazione, funzioni di perdita e ottimizzatori personalizzati, adattandolo alle esigenze specifiche di ogni progetto.

Un aspetto fondamentale di Keras è la sua compatibilità con diverse piattaforme e dispositivi, permettendone l'esecuzione su CPU, GPU e persino TPU. Questa flessibilità garantisce scalabilità e prestazioni elevate, adattandosi sia a piccoli esperimenti su macchine locali che a sistemi di addestramento su larga scala. Inoltre, Keras supporta molteplici backend, tra cui **TensorFlow**, **Theano** e **Microsoft Cognitive Toolkit (CNTK)**, offrendo agli sviluppatori la possibilità di scegliere l'infrastruttura di calcolo più adatta alle proprie esigenze. Uno dei suoi maggiori punti di forza è l'integrazione con TensorFlow, uno dei framework più potenti e diffusi nel Machine Learning. Grazie alla sua semplicità d'uso e alla rapidità con cui permette di sperimentare nuove architetture, Keras è diventato una delle librerie più apprezzate nel Deep Learning, trovando ampio impiego sia in ambito accademico che industriale.

Di seguito vengono riportate le principali librerie di Keras utilizzate per la costruzione del modello.

```
from kerastuner import BayesianOptimization
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from keras.optimizers import Adam
```

3.2.2 Sklearn

Scikit-learn, noto anche come **sklearn**, è una delle librerie più utilizzate per il Machine Learning in Python, grazie alla sua semplicità d'uso, efficienza e ampia gamma di funzionalità. Si tratta di una libreria open-source costruita su *NumPy*, *SciPy* e *Matplotlib*, il che le consente di gestire operazioni di algebra lineare ed array in modo ottimizzato, garantendo al contempo una perfetta integrazione con altre librerie per l'analisi dei dati e la visualizzazione.

Scikit-learn offre strumenti per molteplici compiti di Machine Learning, sia supervisionato che non supervisionato. Tra le sue funzionalità principali rientrano la classificazione, la regressione, il clustering e la riduzione della dimensionalità. Include algoritmi consolidati rendendola, così, una scelta versatile per molte applicazioni. Inoltre, offre strumenti avanzati per la selezione dei modelli, l'ottimizzazione degli iperparametri e la valutazione delle prestazioni, consentendo agli sviluppatori di costruire modelli efficaci con facilità.

Le principali librerie di scikit-learn utilizzate per la progettazione sono le seguenti:

```
from sklearn.utils import shuffle
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

3.2.3 Librosa

Librosa è una libreria open-source sviluppata in Python, progettata per l'analisi e l'elaborazione di file audio, con particolare attenzione alle applicazioni di Machine Learning e Deep Learning. Questa libreria è ampiamente utilizzata per lavorare con file audio nei formati WAV, MP3 e FLAC, e fornisce numerosi strumenti per l'estrazione di caratteristiche fondamentali dai segnali audio, come i Mel Frequency Cepstral Coefficients (MFCC), gli spettrogrammi, le caratteristiche cromatiche (chroma), il tempo, la tonalità (pitch) e molte altre.

Librosa non solo permette di estrarre queste caratteristiche, ma anche di visualizzarle e manipolarle. Ad esempio, è possibile visualizzare la forma d'onda di un segnale audio, calcolare la sua trasformata di Fourier o ottenere uno spettrogramma. Inoltre, la libreria include funzioni per il rilevamento dei beat e del tempo, strumenti particolarmente utili nell'analisi musicale, e per la separazione armonico-percussiva, che consente di distinguere gli elementi melodici da quelli ritmici di un brano.

Una delle principali caratteristiche di Librosa è la possibilità di modificare direttamente l'audio, come il cambiamento di pitch o di tempo, attraverso tecniche come il pitch shifting e il time stretching. Queste funzionalità risultano utili per la manipolazione di brani musicali o per adattare i segnali audio a specifiche esigenze. La libreria supporta anche l'analisi strutturale dei brani, con strumenti per la segmentazione e per il riconoscimento degli onset, ovvero i punti di inizio di eventi significativi nel segnale audio.

Grazie alla sua API ad alto livello e alla facilità di integrazione con altre librerie Python, come Matplotlib per la visualizzazione dei dati, Librosa si presenta come uno strumento estremamente versatile per l'analisi audio. La sua flessibilità la rende fondamentale per chiunque lavori in ambito di riconoscimento vocale, classificazione musicale, ricerca

musicale e elaborazione del segnale.

3.2.4 Altre librerie

Le restanti librerie sono strumenti essenziali per la gestione dei dati, l'analisi statistica e la visualizzazione. **OS** e **Sys** permettono di interagire con il sistema operativo e gestire il flusso di esecuzione del programma, mentre **JSON** facilita la manipolazione di dati in formato JSON. **NumPy** e **Pandas** sono fondamentali per la gestione e l'elaborazione di dati numerici e tabellari, mentre **Seaborn** e **Matplotlib** consentono di creare visualizzazioni efficaci. **SciPy**, con il modulo *signal*, fornisce strumenti avanzati per l'elaborazione dei segnali, tra cui l'individuazione di picchi nei dati, mentre, con il modulo *stats*, offre funzioni statistiche avanzate. **Counter**, dalla libreria *collections*, è utile per analizzare la frequenza degli elementi in una sequenza. Infine, **IPython.display.Audio** permette di riprodurre file audio direttamente in ambiente interattivo.

Capitolo 4

Approccio Tecnico

Nel seguente Capitolo sono illustrate in modo approfondito le diverse fasi del processo, dalla pre-elaborazione dei dati alla progettazione della struttura delle reti neurali impiegate, per poi concludere con la descrizione delle metriche adottate per la valutazione delle performance.

4.1 Data Preprocessing

Dopo aver importato tutte le librerie necessarie, si è proceduto al caricamento dei file audio in formato .wav e alla loro successiva elaborazione.

4.1.1 Data Augmentation

La prima fase del preprocessamento dei dati prevede l'impiego di tecniche di **ADA (Audio Data Augmentation)**, finalizzate ad ampliare e diversificare il dataset mediante l'introduzione di variazioni e perturbazioni nei file audio originali. Le tecniche adottate nel progetto sono descritte nelle funzioni riportate nella Figura 4.1. In particolare, vengono

utilizzate le tecniche di **inserimento del rumore**, **stretching**, **pitching** e **shifting**.

```python
# Funzioni per la data augmentation

def noise(filename):
    data, sr = librosa.load(filename, duration=3, offset=0.5)
    noise_amp = 0.035*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(filename, rate=0.8):
    data, sr = librosa.load(filename, duration=3, offset=0.5)
    return librosa.effects.time_stretch(y=data, rate=rate)

def shift(filename):
    data, sr = librosa.load(filename, duration=3, offset=0.5)
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(filename, n_steps=7):
    data, sr = librosa.load(filename, duration=3, offset=0.5)
    return librosa.effects.pitch_shift(y=data, sr=sr, n_steps=n_steps)
```

Figura 4.1: Funzioni di ADA implementate

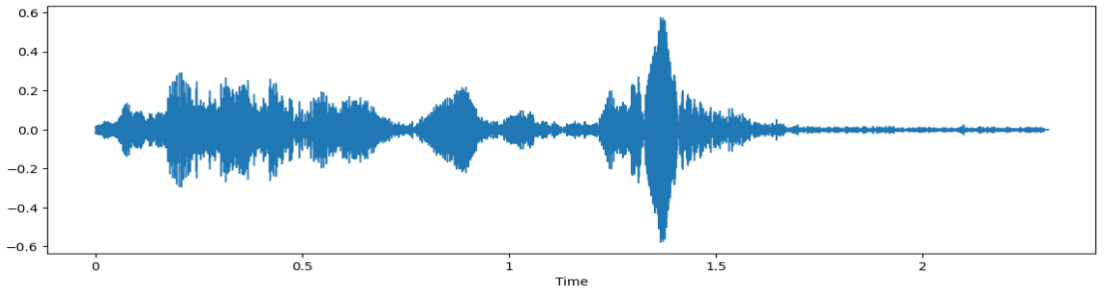
4.1.1.1 Inserimento del rumore

Il primo tipo di modifica applicata all'audio è stata l'aggiunta di rumore. In Figura 4.2 sono riportate le waveforms dello stesso file audio prima e dopo l'inserimento del rumore.

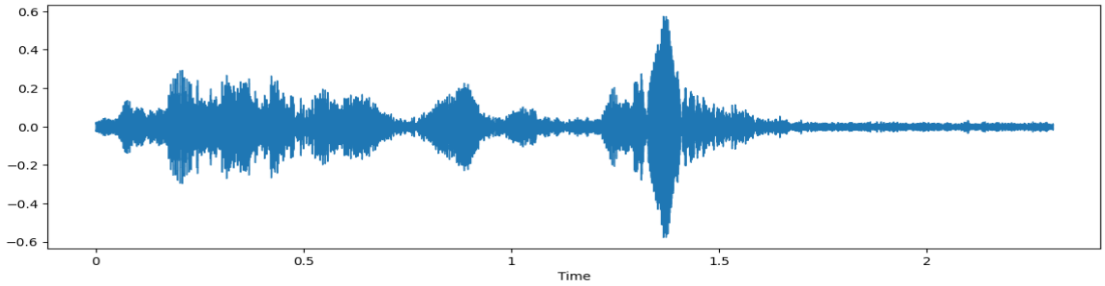
L'aggiunta di rumore può risultare utile perché contribuisce a rendere il modello più robusto rispetto ai disturbi ambientali ed alle variazioni delle condizioni di registrazione. A questo scopo, è stata utilizzata la funzione **noise**, appositamente progettata per questa operazione. La funzione genera un segnale rumoroso moltiplicando l'audio originale per un valore casuale e successivamente somma il segnale ottenuto a quello iniziale, il tutto sfruttando la libreria NumPy.

Analizzando le Figure 4.2a e 4.2b, si nota che la prima rappresenta un segnale più uniforme e regolare, mentre nella seconda sono presenti variazioni brusche nella forma d'onda, indicative di interferenze o alterazioni rispetto al segnale originale. Inoltre, le

oscillazioni nella seconda figura appaiono più frastagliate e irregolari, segno della presenza di disturbi. Al contrario, il segnale nella prima figura si distingue per oscillazioni più fluide e ben definite, caratteristiche tipiche di un audio pulito.



(a)



(b)

Figura 4.2: Differenza tra un audio senza modifiche ed un audio con rumore

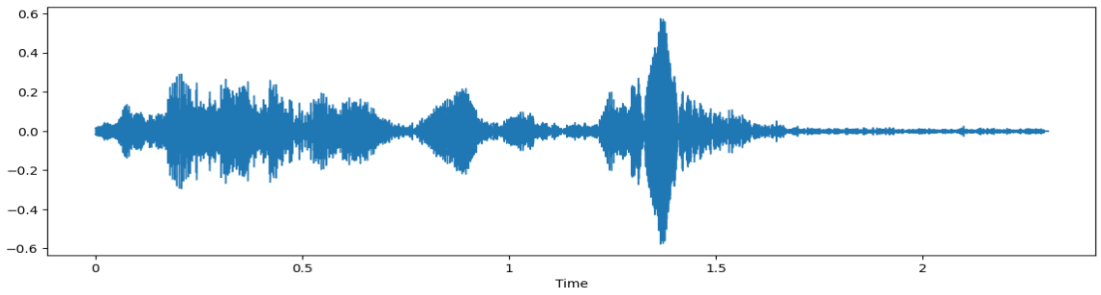
4.1.1.2 Stretching

Il passaggio successivo ha riguardato la modifica della durata del file audio, mantenendone invariata la tonalità. In Figura 4.3 sono riportate le waveforms di un audio originale e della sua versione modificata mediante stretching temporale.

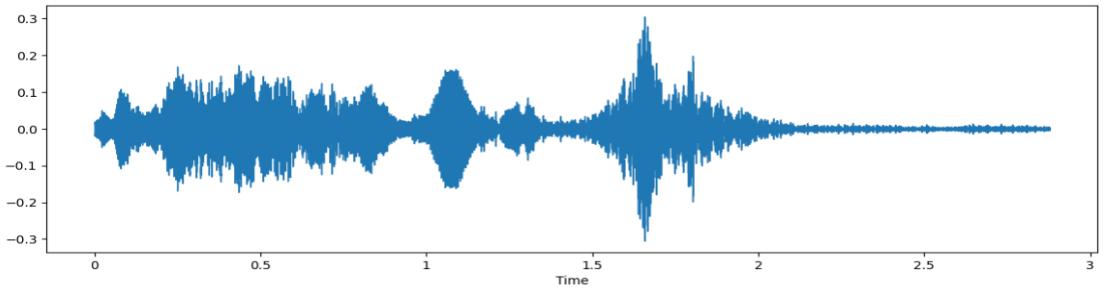
L'alterazione della durata è un'operazione particolarmente utile, in quanto consente di simulare variazioni nel ritmo e nella velocità di un discorso. A tale scopo, è stata

utilizzata la funzione **stretch**, che sfrutta la libreria Librosa e, in particolare, il modulo *time_stretch*. Il parametro **rate** controlla il fattore di stretching: valori inferiori a 1 rallentano l'audio, aumentandone la durata, mentre valori superiori a 1 lo velocizzano, riducendone la lunghezza temporale. Nel caso specifico, il parametro è stato impostato a 0.8, determinando un rallentamento del 20% rispetto alla velocità originale.

Analizzando le Figure 4.3a e 4.3b, si osserva che la seconda waveform risulta allungata nel tempo e, al contempo, compressa lungo l'asse delle ampiezze. Questo effetto si verifica perché la funzione agisce esclusivamente sulla durata dell'audio, senza alterarne la tonalità.



(a)



(b)

Figura 4.3: Differenza tra un audio senza modifiche ed un audio con stretching temporale

4.1.1.3 Shifting

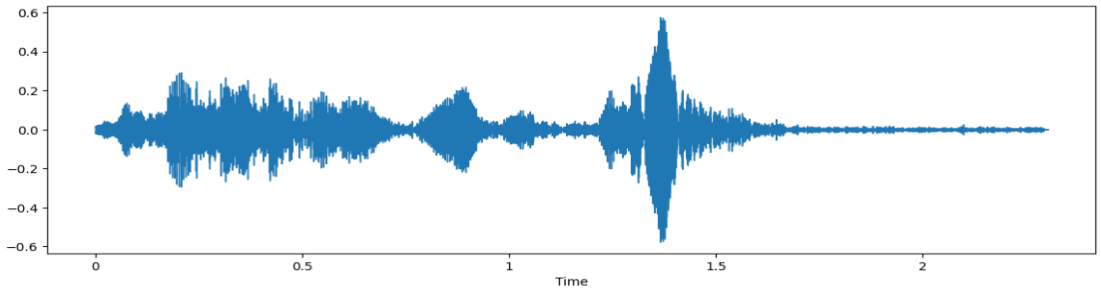
In seguito, si è applicata una traslazione temporale agli audio. In Figura 4.4 vengono messe a confronto le waveforms di un audio originale e della sua versione modificata mediante la funzione **shift**, progettata appositamente per effettuare lo shifting temporale. Questa tecnica è particolarmente utile perché permette di simulare ritardi o variazioni nel timing di un segnale audio.

L'operazione prevede lo spostamento del segnale nel tempo di un numero casuale di campioni e viene realizzata utilizzando la libreria NumPy. Nel caso specifico, il valore di shift viene determinato generando un numero casuale tra -5 e 5, che, moltiplicato per 1000 campioni, definisce lo **shift_range**. Se questo valore è positivo, l'audio viene traslato verso destra, anticipando i suoni. Se il valore è negativo, il segnale viene spostato a sinistra, determinando un ritardo nei suoni.

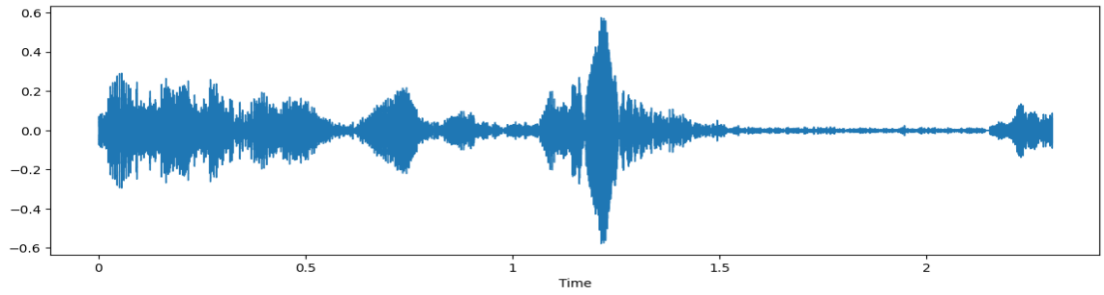
Lo shifting viene eseguito in modo circolare, il che significa che i campioni traslati all'inizio o alla fine dell'audio non vengono eliminati, ma ricompaiono dal lato opposto del segnale. Analizzando le Figure 4.4a e 4.4b, si nota che né la durata né l'ampiezza del segnale risultano alterate. In particolare, dalla seconda figura si può osservare che i campioni iniziali dell'audio originale sono stati spostati verso la fine, confermando che, in questo caso specifico, il valore di *shift_range* era positivo.

4.1.1.4 Pitching

L'ultima modifica applicata all'audio riguarda il cambio di tonalità. In Figura 4.5 vengono confrontate le waveforms di un audio originale e della sua versione modificata tramite la funzione **pitch**. Questa funzione, progettata specificamente per questa operazione, sfrutta la libreria Librosa e, in particolare, il modulo *pitch_shift*.



(a)

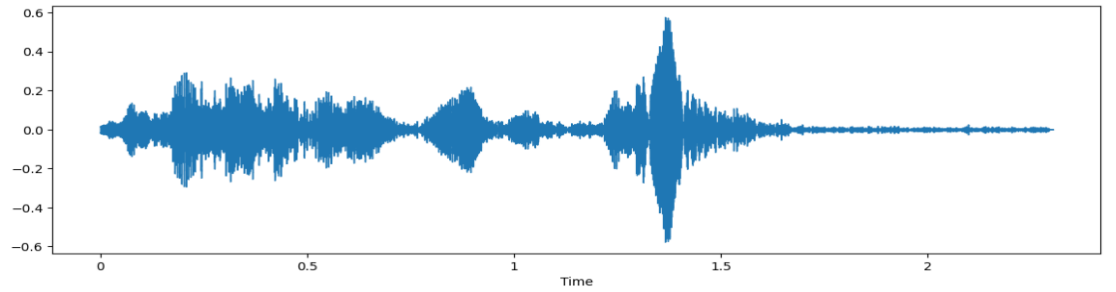


(b)

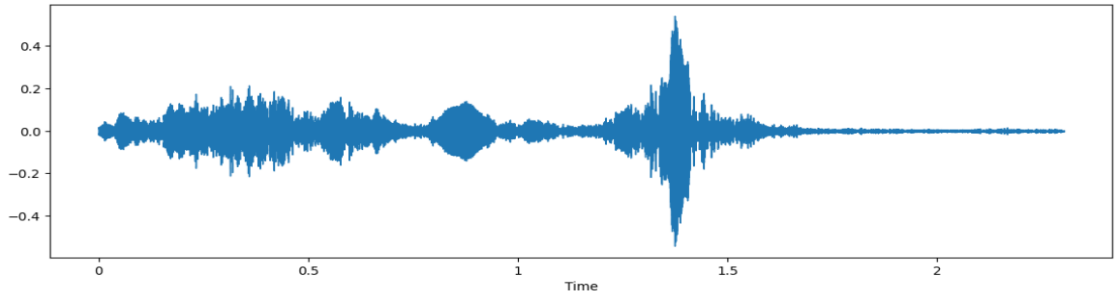
Figura 4.4: Differenza tra un audio senza modifiche ed un audio spostato nel tempo

Il cambio di tonalità è una tecnica molto utile nell'ambito della Data Augmentation, poiché consente di generare nuove varianti dell'audio simulando variazioni della voce a diversi livelli di tonalità. La funzione *pitch* utilizza il parametro **n_steps**, che determina lo spostamento del pitch in termini di semitoni: un valore positivo indica un innalzamento della tonalità, mentre un valore negativo corrisponde a un abbassamento.

Analizzando le Figure 4.5a e 4.5b, si nota che entrambe le forme d'onda coprono lo stesso intervallo temporale, ma la struttura interna delle oscillazioni nella seconda immagine presenta alcune alterazioni. In particolare, le oscillazioni in Figura 4.5b appaiono più compatte e concentrate. Inoltre, si osservano leggere variazioni nell'ampiezza della forma d'onda dell'audio modificato rispetto a quella originale, il che suggerisce che il cambio di tonalità potrebbe aver influenzato anche l'intensità del segnale in alcuni punti.



(a)



(b)

Figura 4.5: Differenza tra un audio senza modifiche ed un audio con cambio di tonalità

4.1.2 Feature Extraction

Dopo l'espansione del dataset con nuovi campioni, si è proceduto all'estrazione delle caratteristiche più significative. Questo processo è stato articolato in tre casi, ciascuno contraddistinto da una diversa combinazione di features estratte.

In ogni caso, le features selezionate vengono estratte tramite funzioni appositamente implementate per questo scopo. Successivamente, il set di caratteristiche estratte, insieme alle relative etichette, viene salvato utilizzando la funzione **to_pickle**, facilitando così la trasmissione ai modelli che verranno sviluppati successivamente. Di seguito è riportata la descrizione dei diversi casi di estrazione delle caratteristiche.

4.1.2.1 Caso 1

Il Caso 1, noto anche come **Caso base**, prevede l'estrazione di cinque caratteristiche fondamentali, selezionate per la loro capacità di rappresentare in modo efficace i segnali audio e per l'efficacia della loro combinazione nell'analisi del parlato e delle emozioni. Le caratteristiche estratte sono le seguenti:

- **Mel-Frequency Cepstral Coefficients (MFCC)**: I coefficienti cepstrali sulla scala Mel rappresentano l'energia spettrale dei segnali audio in una scala che imita il comportamento uditivo umano. Sono ampiamente utilizzati nel riconoscimento del parlato e delle emozioni, poiché catturano variazioni timbriche e tonali fondamentali per la classificazione delle emozioni vocali.

- **Chroma**: Queste feature rappresentano l'intensità dell'energia nelle 12 classi di pitch (una per ogni semitono) all'interno di un'ottava. Sono particolarmente utili per identificare la tonalità, essendo insensibili all'ottava ma sensibili alla struttura armonica del segnale.

- **Mel-Spectrogram**: È uno spettrogramma in cui le frequenze sono trasformate secondo la scala Mel, enfatizzando le frequenze basse. Questa trasformazione rispetta la percezione uditiva umana e consente di visualizzare la distribuzione dell'energia nel tempo e nelle frequenze in modo più efficace rispetto a una scala lineare.

- **Spectral Contrast**: Misura la differenza tra i picchi e le valli dello spettro, evidenziando la struttura armonica e timbrica del segnale. Questa caratteristica aiuta a distinguere suoni con dinamiche diverse e a cogliere dettagli sottili nella variazione del timbro.

- **Tonnetz Representation**: Questa caratteristica deriva da una mappa geometrica della percezione armonica, utile per catturare relazioni tonali e consonanze tra le frequenze. È particolarmente efficace per analizzare la musicalità del parlato e la sua espressività.

L'integrazione di queste feature permette di ottenere una rappresentazione più completa del segnale audio. Le *MFCC* forniscono informazioni timbriche, mentre *Chroma* e *Tonnetz Representation* evidenziano gli aspetti tonali e armonici. Il *Mel-Spectrogram* contribuisce alla caratterizzazione dell'energia nelle frequenze, e lo *Spectral Contrast* aiuta a distinguere segnali con dinamiche differenti.

4.1.2.2 Caso 2

Il Caso 2, noto anche come **Caso esteso**, rappresenta un'estensione del Caso Base, poiché alle cinque caratteristiche precedentemente estratte si aggiungono ulteriori cinque features, migliorando così l'analisi dei segnali audio. Queste nuove caratteristiche permettono di ottenere una rappresentazione più ricca del segnale, includendo informazioni sulla struttura energetica, la qualità vocale e le transizioni nel parlato. Le nuove caratteristiche aggiunte sono le seguenti:

- **Zero-Crossing Rate (ZCR)**: Indica la frequenza con cui il segnale audio cambia segno (da positivo a negativo e viceversa) in un breve intervallo temporale. È utile per identificare segmenti impulsivi e per l'analisi delle consonanti fricative nel parlato.
- **Harmonics-to-Noise Ratio (HNR)**: Rappresenta il rapporto tra l'energia armonica e l'energia del rumore in un segnale vocale. È utilizzato per valutare la qualità della voce, distinguendo voci pulite da quelle disturbate da rumore o affaticamento vocale.

- **Teager Energy Operator (TEO)**: È un operatore di energia non lineare che misura l'energia istantanea di un segnale. È altamente sensibile alle variazioni rapide, rendendolo utile per individuare transizioni e dettagli sottili nella modulazione vocale.

- **Short-Time Energy (STE)**: Misura l'intensità di un segnale in una piccola finestra temporale, aiutando ad individuare la presenza di voce, suoni forti o silenzi in segmenti temporali ridotti. È particolarmente utile nell'analisi della prosodia e dell'accento nel parlato.

- **Power Spectral Density (PSD)**: Descrive la distribuzione dell'energia del segnale nelle varie frequenze, evidenziando le componenti spettrali dominanti e permettendo di analizzare la struttura armonica globale.

L'aggiunta di queste cinque features alle caratteristiche già presenti nel Caso 1 consente una rappresentazione più dettagliata e multidimensionale dei segnali vocali. *STE* e *TEO* forniscono informazioni sull'energia e sulle transizioni nel parlato, migliorando la capacità di identificare segmenti ad alta intensità emotiva. *HNR* permette di valutare la qualità vocale, distinguendo tra parlato pulito e segnali affetti da tensione vocale. *ZCR* aiuta a distinguere tra suoni armonici e segmenti a contenuto impulsivo, risultando particolarmente utile nella segmentazione del parlato. *PSD* completa l'analisi fornendo informazioni dettagliate sulla distribuzione dell'energia nelle diverse bande di frequenza. Questa combinazione arricchita fornisce una rappresentazione più completa della voce e delle sue variazioni nel tempo, potenziando significativamente la capacità del modello di cogliere le sfumature emotive nei segnali audio.

4.1.2.3 Caso 3

Il Caso 3, noto anche come **Caso Specifico**, rappresenta un'estensione del Caso 1 con l'integrazione di due parametri fondamentali per l'analisi della qualità vocale: **Jitter** e **Shimmer**. Queste caratteristiche forniscono informazioni cruciali sulla stabilità della voce e sulla sua variazione nel tempo. Analisi delle nuove features introdotte:

- **Jitter**: misura la variazione ciclo per ciclo della frequenza fondamentale (F0) della voce. In altri termini, indica il livello di instabilità delle vibrazioni delle corde vocali. Valori elevati di jitter possono segnalare una voce instabile, mentre valori bassi sono associati a una maggiore stabilità e regolarità vocale.

- **Shimmer**: misura la variazione ciclo per ciclo dell'ampiezza della voce, indicando quanto l'intensità del suono varia da un ciclo all'altro. Un elevato shimmer suggerisce una maggiore instabilità nell'emissione vocale, mentre un valore basso è tipico di una voce più uniforme e controllata.

L'inclusione di *Jitter* e *Shimmer* nel set di caratteristiche di base arricchisce l'analisi del segnale vocale, permettendo al modello di cogliere con maggiore precisione le sfumature sottili della voce.

Nel riconoscimento delle emozioni nel parlato, la stabilità della voce è direttamente influenzata dallo stato emotivo: emozioni come paura e rabbia tendono ad aumentare i valori di jitter e shimmer, mentre una voce neutrale presenta una maggiore regolarità. L'integrazione di queste feature potrebbe quindi migliorare la capacità del modello di distinguere emozioni tra loro simili, come paura e neutralità o disgusto e paura, che potrebbero risultare spesso confuse.

Nel complesso, l'aggiunta queste due caratteristiche permette di ottenere una rappresentazione più dettagliata e accurata della voce, affinando non solo il riconoscimento delle emozioni, ma anche l'analisi della qualità vocale e dello stato psicofisico del parlante.

Il Caso 3 verrà esaminato in modo approfondito ed in via esclusiva nella Sezione 5.2 di questa Tesi, dove sarà presentata l'analisi dei risultati ottenuti al termine del primo ciclo di valutazione.

4.2 Visualizzazione

Dopo aver completato il pre-processamento dei dati, si è proceduto con l'analisi esplorativa del dataset, esaminandone la struttura e visualizzando le caratteristiche delle istanze in relazione alle diverse emozioni.

4.2.1 Grafico di Distribuzione delle Emozioni

La prima operazione effettuata è stata l'analisi della distribuzione dei file audio per ciascuna emozione considerata. Questa visualizzazione è stata eseguita sul dataset originale, senza includere le istanze generate tramite data augmentation. Tale scelta è giustificata dal fatto che, come descritto in Sezione 4.1.1, la data augmentation è stata applicata uniformemente a tutte le classi, mantenendo invariato il rapporto tra di esse.

Come mostrato in Figura 4.6, il dataset risulta complessivamente bilanciato: infatti, nel dataset originale sono presenti circa 1200 esempi per ciascuna classe (nello specifico, 1271), ad eccezione della classe *NEUTRALITY*, che conta poco più di 1000 campioni (esattamente 1087). Complessivamente, la quantità di dati di partenza può essere considerata soddisfacente.

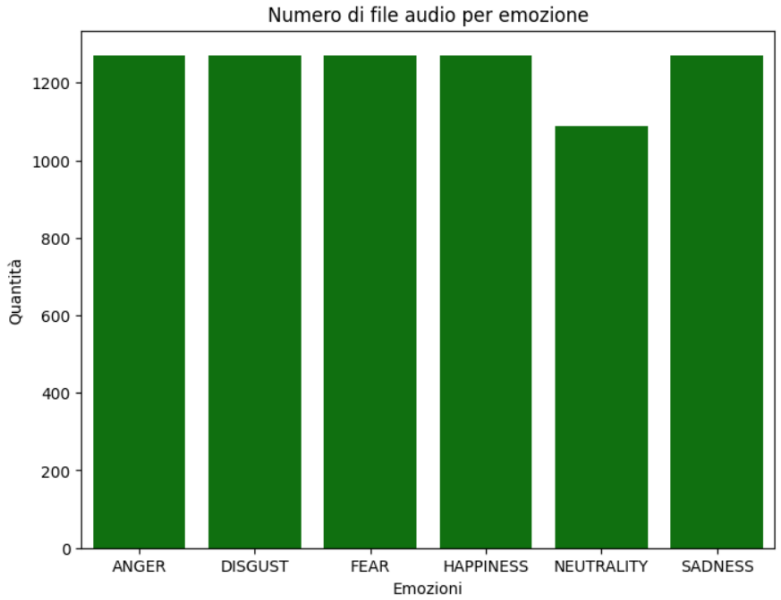


Figura 4.6: Distribuzione delle 6 emozioni studiate

4.2.2 Waveforms e Spettogrammi

In questa sezione, l'analisi è stata approfondita a livello di singola classe, attraverso la visualizzazione delle forme d'onda e degli spettrogrammi di ciascuna emozione.

Le **waveforms** rappresentano graficamente un segnale audio nel dominio del tempo, consentendo di analizzare l'andamento dell'ampiezza del suono in funzione del tempo. L'asse orizzontale indica la progressione temporale, mentre l'asse verticale esprime l'ampiezza del segnale, che corrisponde all'intensità sonora in un determinato istante. Un'ampiezza maggiore indica un volume più elevato, mentre un'ampiezza ridotta segnala un suono più tenue.

Questa rappresentazione visiva costituisce uno strumento fondamentale per l'analisi del suono: l'osservazione di una waveform consente di individuare pause e momenti di silenzio, valutare le variazioni di intensità e identificare schemi distintivi nelle diverse espressioni vocali.

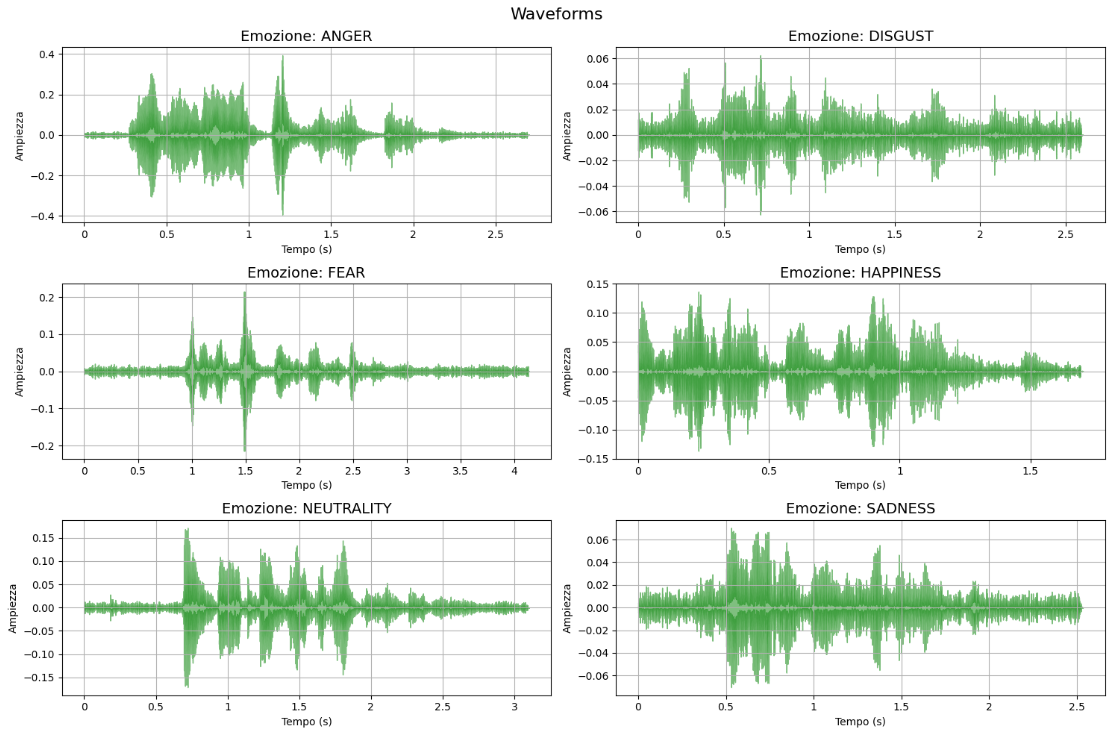


Figura 4.7: Forme d'onda delle 6 emozioni analizzate

Lo **spettrogramma** è una rappresentazione visiva di un segnale audio che offre una visione tridimensionale del contenuto acustico, combinando tempo, frequenza e ampiezza in un'unica visualizzazione. A differenza della waveform, che si limita a mostrare l'andamento dell'ampiezza nel tempo, lo spettrogramma aggiunge una dimensione di frequenza, evidenziando come l'energia del segnale si distribuisca lungo lo spettro delle frequenze. L'asse orizzontale rappresenta il tempo, mentre quello verticale mostra le frequenze, dalle più basse alle più alte. Inoltre, l'intensità o il colore dei punti nello spettrogramma indica l'ampiezza del segnale, fornendo una **mappa di calore** (o **heatmap**) che evidenzia i momenti di maggiore o minore intensità acustica.

Questo strumento permette di esaminare le caratteristiche spettrali che determinano l'intensità e la qualità dei suoni: nel contesto specifico della classificazione delle emozioni,

diversi stati emotivi si manifestano attraverso distinte distribuzioni spettrali. Grazie alla sua capacità di illustrare l'evoluzione delle frequenze nel tempo, lo spettrogramma rappresenta uno strumento particolarmente utile per un'analisi approfondita delle sfumature sonore che costituiscono il parlato.

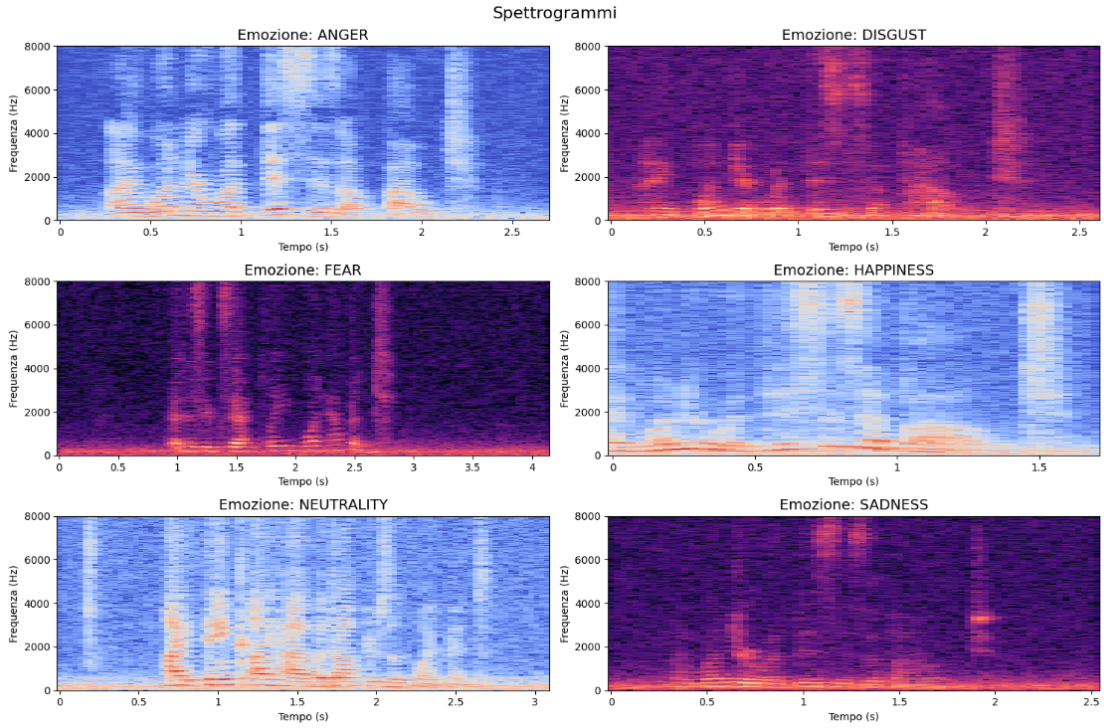


Figura 4.8: Spettogrammi delle 6 emozioni considerate

4.3 Preparazione dei dati

La fase di preparazione dei dati include pochi passaggi, ma ognuno di essi è fondamentale per garantire un addestramento ottimale del modello. In particolare, questa fase si articola in:

- Formattazione dei Dati;
- Randomizzazione e Splitting;

- One-Hot Encoding.

4.3.1 Formattazione dei Dati

Nel contesto del progetto, la fase di formattazione dei dati rappresenta il primo e fondamentale passaggio nella preparazione dei dati stessi. In questa fase, vengono eseguite due operazioni principali: la concatenazione delle feature e la separazione tra i dati delle feature e le etichette. La funzione **safe_concatenate** è responsabile della concatenazione sicura dei valori presenti nelle colonne delle diverse feature estratte, combinandoli in un'unica struttura array, mentre garantisce robustezza contro vari casi ed eventuali eccezioni. In particolare, la funzione esamina ogni singola riga e concatena i valori delle colonne specificate. Durante questo processo, si verifica che ogni valore non sia vuoto. Nel caso in cui i valori nelle colonne siano array monodimensionali, questi vengono trattati come tali e concatenati in un array monodimensionale. Infine, un controllo viene effettuato per assicurarsi che tutti gli array risultanti dalle righe abbiano la stessa dimensione. Completata questa procedura, i dati vengono separati in due set distinti: le **features**, che vengono concatenate e strutturate in un array NumPy 2D, salvate nella variabile **X**, e le **etichette**, che vengono estratte e convertite in un array NumPy 1D, memorizzate invece nella variabile **y**.

4.3.2 Randomizzazione e Splitting

Il passo successivo nella preparazione dei dati è lo **splitting**, un'operazione fondamentale per garantire che il modello possa imparare in modo efficace e generalizzare sui nuovi dati senza memorizzare pattern specifici.

In primo luogo, i dati nelle variabili X e y vengono mescolati casualmente per evitare qualsiasi ordine che possa influenzare il processo di training. Successivamente, viene applicata la tecnica di **Stratified K-Fold cross-validation** per dividere il dataset in

5 sottoinsiemi (**folds**), mantenendo la proporzione originale delle classi in ciascun fold. Questo approccio garantisce che ogni fold contenga una distribuzione delle etichette che rispecchi quella dell'intero dataset. Per ciascun fold, i dati vengono separati in due gruppi: **X_train**, il set di addestramento, e **X_temp**, che contiene i dati da suddividere ulteriormente in un set di test e uno di validazione. Successivamente, i dati in *X_ temp* vengono divisi in due sottoinsiemi: **X_val** per il set di validazione, che costituisce il 30% dei dati, e **X_test** per il set di test. Analogamente, le etichette in y vengono suddivise in **y_train**, **y_val** e **y_test**. La suddivisione avviene secondo la seguente logica:

- **Training Set**: è il subset su cui il modello viene addestrato. Il modello impara i pattern dai dati
- **Validation Set**: viene utilizzato per monitorare le prestazioni del modello durante l'addestramento, permettendo di ottimizzare gli iperparametri e prevenire il rischio di overfitting.
- **Test Set**: è il subset finale utilizzato per valutare le prestazioni del modello una volta completato l'addestramento. Non viene mai utilizzato durante la fase di training, assicurando che la valutazione del modello simuli il comportamento su dati nuovi e sconosciuti.

Successivamente, viene verificato il bilanciamento delle classi nei set di dati per assicurarsi che la distribuzione delle etichette sia coerente dopo la suddivisione.
Infine, i dati vengono trasformati in modo che siano compatibili con una rete neurale convoluzionale 1D. Questo passaggio aggiunge una dimensione in modo che la forma finale dei dati diventi **(n_samples, n_features, 1)**, dove *n_features* rappresenta il numero di caratteristiche estratte per ciascun esempio.

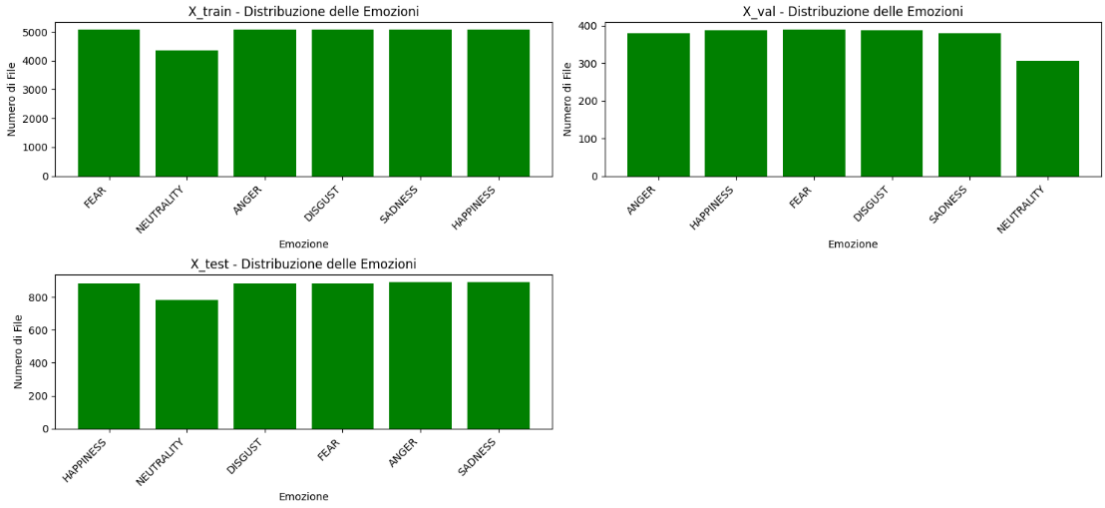


Figura 4.9: Esempio di randomizzazione della distribuzione dei file audio

4.3.3 One-Hot Encoding

La **One-Hot Encoding** è una tecnica di codifica utilizzata per trasformare variabili categoriali in un formato numerico comprensibile per le reti neurali. Questo metodo converte ogni categoria in un vettore binario in cui solo un elemento assume il valore 1, mentre tutti gli altri sono impostati a 0. In questo modo, ogni categoria viene rappresentata in modo univoco senza introdurre relazioni ordinali tra i valori. La codifica One-Hot è particolarmente utile quando i dati contengono valori categoriali, poiché evita che il modello interpreti erroneamente una relazione numerica tra le categorie. D'altra parte, questa tecnica può portare ad un aumento significativo del numero di colonne nel dataset, specialmente quando il numero di categorie è elevato.

Nel caso specifico, le etichette sono memorizzate nella variabile y, che, dopo la suddivisione tramite Stratified K-Fold, è stata separata in y_train, y_val e y_test. Di conseguenza, è stato necessario applicare la codifica One-Hot separatamente su ciascuna di queste tre variabili, garantendo così che tutte le etichette fossero trasformate correttamente nel

formato binario richiesto dal modello.

4.4 Creazione ed Addestramento del Modello

La fase finale del progetto prevede la creazione del modello e il processo di addestramento. Di seguito viene descritta in dettaglio la costruzione della rete neurale convoluzionale 1D, illustrandone l'architettura e le scelte progettuali. Inoltre, si analizza la fase di training, evidenziando le metriche impiegate per la valutazione delle performance del modello.

4.4.1 Modello

Il modello scelto per la rete neurale convoluzionale 1D è di tipo **Sequential**, una struttura lineare che consente di impilare gli strati della rete uno dopo l'altro in modo sequenziale. Questa scelta è motivata dalla semplicità di implementazione e dalla possibilità di costruire il modello in modo incrementale utilizzando il metodo *add*. La sua natura lineare crea un flusso di dati unidirezionale, senza connessioni multiple tra strati, ricorsioni o salti tra livelli.

Nonostante la sua semplicità, la rete è progettata per essere sufficientemente efficiente nel portare a termine il compito di classificazione. Il primo strato è una convoluzione 1D (**Conv1D**) con 64 filtri e funzione di attivazione $ReLU$, che introduce non linearità nel modello e garantisce un basso costo computazionale. I dati in ingresso non vengono definiti a priori, ma vengono adattati in base alla variabile X_train opportunamente ridimensionata. A questo strato convoluzionale segue un livello di **MaxPooling**, che riduce la dimensionalità dei dati, e un livello di **Dropout** con un tasso di 0.2, utile per mitigare il rischio di overfitting disattivando casualmente il 20% dei neuroni.

Successivamente, il modello include un secondo strato convoluzionale con 128 filtri e funzione di attivazione *ReLU*, seguito nuovamente da uno strato di *MaxPooling* e un altro livello di *Dropout*. Dopo questa fase convoluzionale, l'output viene trasformato in un vettore monodimensionale attraverso uno strato **Flatten**. Infine, la rete termina con due strati completamente connessi: il primo, con 64 neuroni e attivazione *ReLU*, e il secondo, che rappresenta il livello di output, dotato di funzione di attivazione **softmax**. Quest'ultima è ideale per la classificazione multiclasse, poiché trasforma l'output in una distribuzione di probabilità con somma unitaria. Il numero di classi nell'ultimo livello **Dense** non è fissato in fase di costruzione, ma è determinato dalla variabile *y_train*, codificata precedentemente in formato One-Hot. La struttura completa del modello è riportata in Figura 4.10.
La compilazione del modello avviene tramite il metodo *compile* del modello Sequential. L'ottimizzatore scelto per questa fase è **Adam** (**Adaptive Moment Estimation**), un metodo che combina le tecniche di *Momentum* e *RMSprop*. La prima accelera la convergenza accumulando i gradienti passati, mentre la seconda adatta il tasso di apprendimento per ogni parametro, riducendo così le oscillazioni. *Adam* si distingue per la sua capacità di regolare dinamicamente il tasso di apprendimento per ogni peso, ottimizzando quindi l'aggiornamento dei parametri durante l'addestramento.

Per quanto riguarda la funzione di perdita, è stata selezionata la **categorical_crossentropy**, che è ampiamente utilizzata nei problemi di classificazione multiclasse. Questa funzione misura la differenza tra le probabilità previste dal modello e le etichette reali dei dati di addestramento, calcolando quanto le previsioni del modello si discostano dalla distribuzione di probabilità corretta.

Il modello viene infine valutato utilizzando la metrica **accuracy**, che rappresenta la percentuale di previsioni corrette rispetto al numero totale di previsioni effettuate dal

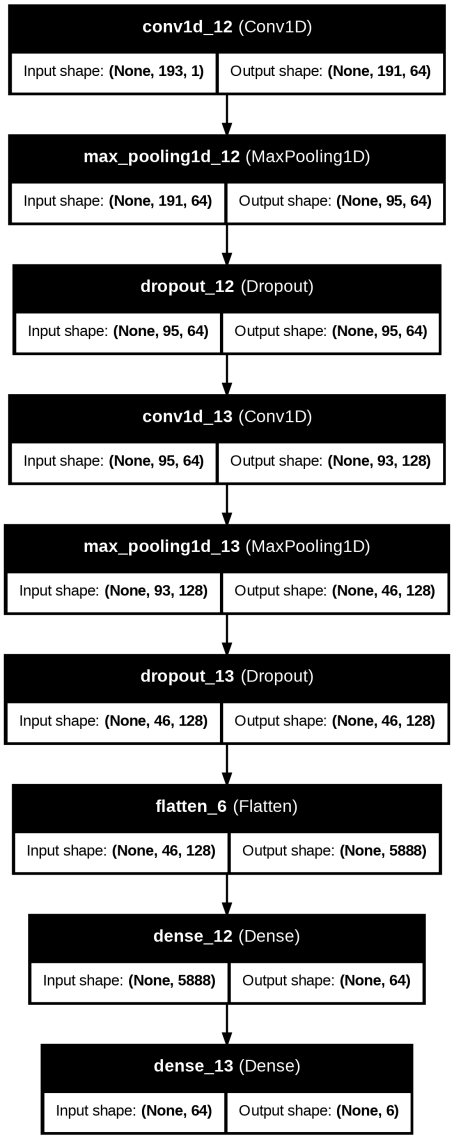


Figura 4.10: Struttura della Rete Neurale Covoluzionale 1D

modello.

Come ultimo passo, è stata implementata la visualizzazione del sommario del modello tramite il metodo *summary*. Questa operazione consente di stampare la struttura complessiva del modello in formato tabellare, come mostrato in Figura 4.11. Il sommario fornisce informazioni dettagliate sulle forme degli output e sul numero di parametri addestrabili

per ciascun strato della rete.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 191, 64)	256
max_pooling1d (MaxPooling1D)	(None, 95, 64)	0
dropout (Dropout)	(None, 95, 64)	0
conv1d_1 (Conv1D)	(None, 93, 128)	24,704
max_pooling1d_1 (MaxPooling1D)	(None, 46, 128)	0
dropout_1 (Dropout)	(None, 46, 128)	0
flatten (Flatten)	(None, 5888)	0
dense (Dense)	(None, 64)	376,896
dense_1 (Dense)	(None, 6)	390

Total params: 402,246 (1.53 MB)
Trainable params: 402,246 (1.53 MB)
Non-trainable params: 0 (0.00 B)

Figura 4.11: Sommario del modello implementato

4.4.2 Training

L'addestramento del modello viene avviato tramite il metodo **fit** offerto dal modello *Sequential*. Per iniziare il training, è necessario fornire al metodo alcuni parametri specifici. Innanzitutto, bisogna indicare le variabili di input per le feature dei dati (X_train) e per le etichette (y_train). Successivamente, è fondamentale definire i dati di validazione, che vengono utilizzati per valutare le prestazioni del modello ad ogni epoca durante l'addestramento. In questo caso, i dati di validazione sono rappresentati dalle variabili X_val e y_val, precedentemente definite per tale scopo. Un altro parametro da specificare è il numero di **epoche**, che rappresentano i cicli completi di addestramento. In questo caso, è stato scelto di eseguire 100 epoche. Infine, il parametro **batch_size** indica il numero di istanze di dati che vengono elaborate prima che i pesi vengano aggiornati. In questo caso, si è deciso di eseguire un aggiornamento dei pesi dopo aver elaborato 32 esempi.

Il processo di addestramento si sviluppa in tre fasi principali, che vengono ripetute per ogni batch di dati durante ogni epoca:

1. **Forward Propagation**: in questa fase, per ogni batch, il modello calcola le previsioni a partire dagli input, passando attraverso tutti gli strati della rete.

2. **Calcolo della Perdita**: qui, la funzione di perdita *categorical_ crossentropy* misura la differenza tra le previsioni del modello e le etichette reali, quantificando l'errore.

3. **Backpropagation**: in questa fase, il modello calcola il gradiente della funzione di perdita rispetto ai pesi ed aggiorna i pesi stessi utilizzando l'ottimizzatore *Adam*.

Durante l'addestramento, è possibile monitorare l'andamento della perdita e dell'accuratezza per ogni epoca, sia sui dati del training set che su quelli del validation set, per osservare i progressi del modello e l'efficacia dell'addestramento.

Infine, per effettuare una valutazione completa del modello, è necessario utilizzare uno strumento chiamato **matrice di confusione**. La matrice di confusione è un elemento fondamentale per analizzare nel dettaglio le performance di un modello di classificazione, in quanto consente di identificare facilmente i punti di forza e di debolezza, nonché di comprendere i tipi di errori più frequentemente commessi. La matrice di confusione può essere rappresentata, come mostrato in Figura 4.12, e si concentra sulle quattro categorie principali che derivano dal confronto tra i valori reali e quelli predetti: **True Positive (TP)**, **False Positive (FP)**, **False Negative (FN)** e **True Negative (TN)**.

- **True Positive (TP)**: che rappresenta il numero di esempi correttamente classificati come positivi;

- **False Positive (FP)**: che indica il numero di esempi erroneamente classificati come positivi, quando in realtà hanno etichetta negativa;

- **False Negative (FN)**: ovvero il numero di esempi erroneamente classificati come negativi, quando in realtà presentano un'etichetta positiva;
- **True Negative (TN)**: che denota il numero di esempi correttamente classificati come negativi.

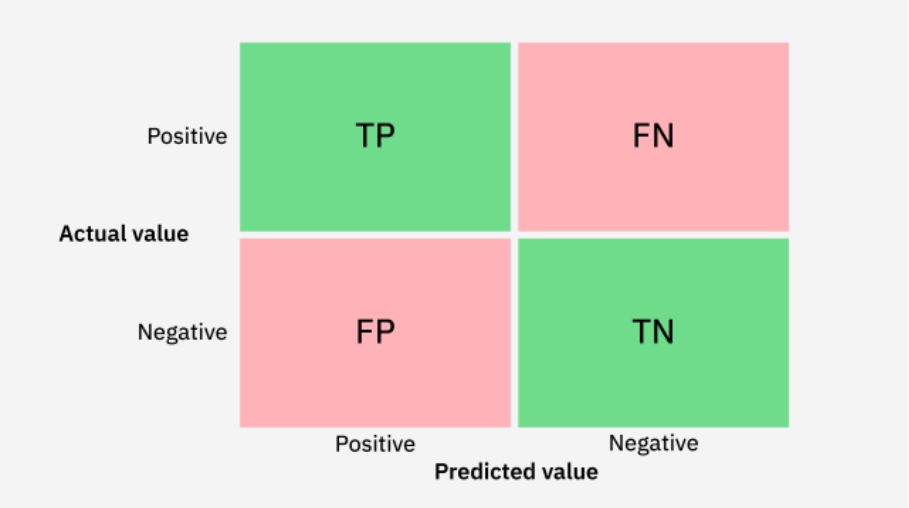


Figura 4.12: Esempio di matrice di confusione

Capitolo 5

Sperimentazione

Il progetto è stato articolato in quattro diverse sperimentazioni. In questo Capitolo finale verranno descritte nel dettaglio le analisi e le valutazioni condotte per ciascuna di esse. Sebbene il metodo di analisi dei risultati rimanga invariato, i risultati ottenuti sono differenti. Nelle sezioni seguenti, tali valutazioni verranno esaminate in modo approfondito.

5.1 6 Emozioni

In questa prima sezione viene presentata l'analisi delle sperimentazioni condotte sulla versione originale del progetto, che include tutte le sei emozioni. La valutazione si basa sui casi 1 e 2, descritti nelle Sezioni 4.1.2.1 e 4.1.2.2.

5.1.1 Caso 1

Il primo passo fondamentale per una corretta valutazione consiste nella creazione dei grafici delle curve di accuratezza e perdita. Grazie al metodo **plot** fornito dalla libreria *Matplotlib*, è stato possibile generare due grafici rappresentativi di questi valori, come mostrato in Figura 5.1.

Ogni grafico è stato tracciato utilizzando i dati memorizzati nella variabile **history**, che conserva tutti i checkpoint dell'addestramento del modello relativo al caso. In entrambi i grafici, l'asse delle ascisse rappresenta le epoche, consentendo così una valutazione completa dell'intero processo di addestramento. Inoltre, in entrambe le rappresentazioni, sono riportate le curve relative sia al training set che al validation set.
Andando a studiare i due grafici, si osserva che l'accuratezza del modello sul training set e sul validation set aumenta progressivamente con il numero di epoche, indicando un miglioramento delle capacità di apprendimento. Tuttavia, dopo circa 20-30 epoche, l'incremento dell'accuratezza di validazione diviene meno regolare, evidenziando possibili oscillazioni dovute a variazioni nei batch di validazione o a una lieve instabilità del modello. Inoltre, l'accuratezza sul training set risulta costantemente superiore rispetto a quella sul validation set, suggerendo una possibile tendenza all'overfitting.

Per quanto riguarda la funzione di perdita, invece, si osserva una progressiva diminuzione sia della training loss che della validation loss, un comportamento chiaramente positivo per l'apprendimento. Tuttavia, mentre la validation loss si stabilizza dopo circa 40-50 epoche, la training loss continua a diminuire: tale andamento suggerisce l'insorgenza di overfitting, in cui il modello si adatta in modo eccessivo ai dati di addestramento senza ottenere miglioramenti significativi sulle istanze di validazione. La crescente discrepanza tra le due curve nelle ultime epoche rappresenta un ulteriore indicatore di questo fenomeno.

Il passaggio successivo è stato l'analisi dei risultati delle principali metriche di valutazione tramite un **report di classificazione**, che fornisce un riassunto delle performance di un modello di classificazione. Questo report viene generato utilizzando la funzione **classification_report** ed include diverse metriche fondamentali, tra cui **Precision**, **Recall** e **F1-Score**.

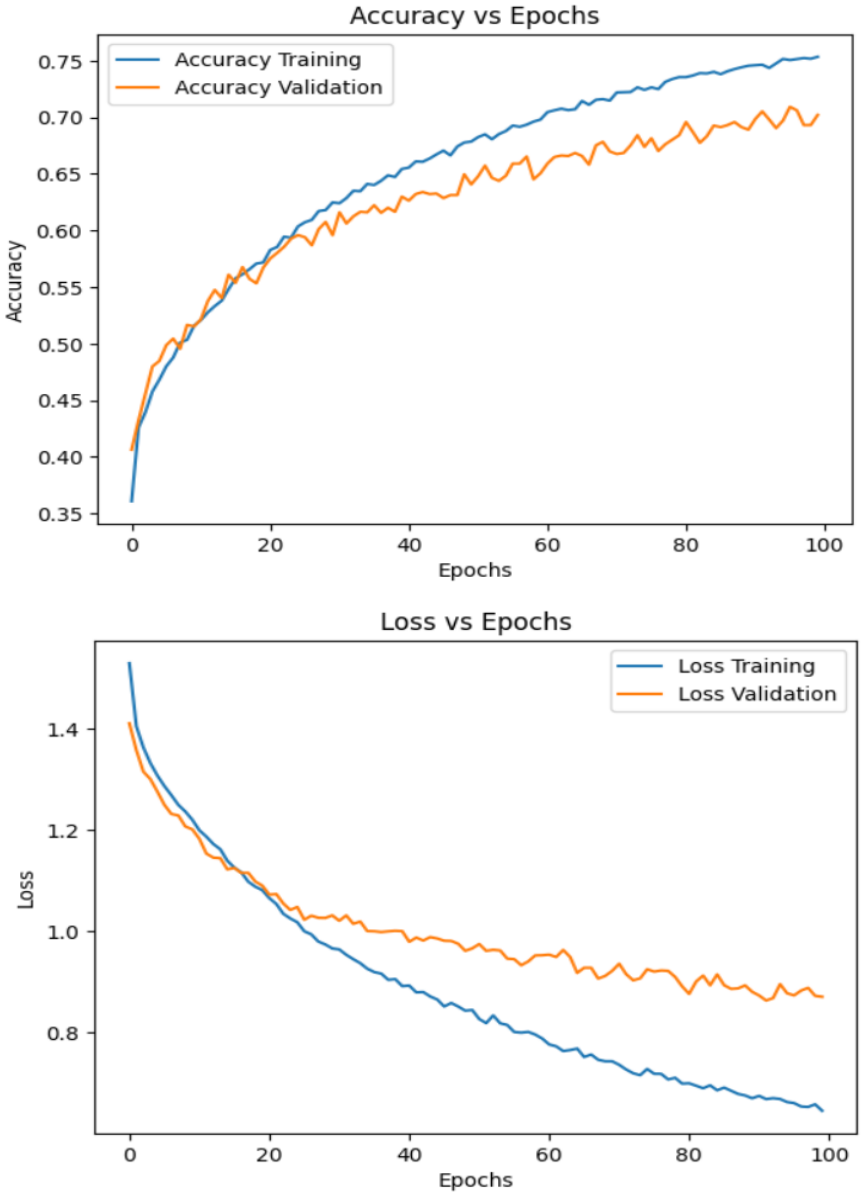


Figura 5.1: Grafici di Accuracy e Loss relativi al Caso 1 dello studio svolto su 6 emozioni

- **Precision**: misura la percentuale di veri positivi rispetto al totale dei casi classificati come positivi, secondo la formula:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: misura la percentuale di veri positivi rispetto al totale dei casi realmente positivi, secondo la seguente formula:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score**: è la media armonica tra Precision e Recall, calcolata come:

$$\text{F1-Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}$$

Oltre ai valori per ciascuna classe, il report include anche l'accuratezza complessiva del modello, la **Macro avg**, ovvero la media semplice delle metriche di tutte le classi, e la **Weighted avg**, che rappresenta la media pesata in base al numero di campioni per ciascuna classe. Il report completo è mostrato nella Figura 5.2.

Rapporto di classificazione:

	precision	recall	f1-score	support
ANGER	0.82	0.82	0.82	891
DISGUST	0.68	0.57	0.62	883
FEAR	0.72	0.64	0.68	881
HAPPINESS	0.67	0.71	0.69	883
NEUTRALITY	0.64	0.68	0.66	781
SADNESS	0.64	0.74	0.69	891
accuracy			0.69	5210
macro avg	0.70	0.69	0.69	5210
weighted avg	0.70	0.69	0.69	5210

Figura 5.2: Report di classificazione relativo al Caso 1 dello studio svolto su 6 emozioni

Nel caso dello studio condotto, le medie macro e weighted di precisione, recall e F1-score sono circa 0.70, ciò indica prestazioni abbastanza bilanciate tra le classi. Tuttavia, analizzando i risultati per singola classe, emergono alcune osservazioni importanti. L'emozione *ANGER* mostra i valori più alti sia per la precisione che per il recall, pari a 0.82, suggerendo che il modello riconosce molto bene questa emozione. Al contrario, l'emozione di

DISGUST presenta il recall più basso, con un valore di 0.57, indicando che molti campioni di questa emozione vengono erroneamente classificati come appartenenti ad altre classi. Per l'emozione di *FEAR*, si osserva una precisione discreta (0.72), ma un recall più basso (0.64), il che implica che alcuni esempi di paura vengono confusi con altre emozioni. Nel caso di *HAPPINESS* e *SADNESS*, il modello ha un recall più alto della precisione (0.71 e 0.67 per la felicità e 0.74 e 0.64 per la tristezza), il che suggerisce che tende a classificare più esempi come appartenenti a queste classi di quanti dovrebbero appartenerne, sebbene alcuni possano risultare falsi positivi. Per l'emozione di *NEUTRALITY*, precisione e recall sono simili (0.64 e 0.68), il che implica che il modello spesso la confonde con altre emozioni. Dai risultati evidenziati in Figura 5.2, si nota, quindi, che la classe più problematica è *DISGUST*, spesso confusa con altre emozioni. Inoltre, il modello tende a scambiare *FEAR* e *NEUTRALITY*, suggerendo una difficoltà nel distinguerle con precisione.

Per analizzare più a fondo il problema della misclassificazione delle emozioni, è stata generata una matrice di confusione dettagliata. Come si può osservare in Figura 5.3, in questa rappresentazione le classi predette sono riportate sull'asse delle ascisse, mentre le etichette reali si trovano sull'asse delle ordinate. Accanto alla matrice è presente una scala di colori che rappresenta il numero di istanze classificate: tonalità più scure o intense indicano una maggiore frequenza di esempi per una determinata combinazione di etichetta reale e predetta. Se il colore più scuro si concentra lungo la diagonale principale, significa che un elevato numero di istanze è stato classificato correttamente. Al contrario, se l'intensità del colore aumenta in celle al di fuori della diagonale, ciò evidenzia gli errori più frequenti nella classificazione.

In aggiunta alla matrice di classificazione, sono stati implementati un dizionario ed una funzione finalizzati al calcolo della percentuale di misclassificazione per ciascuna classe. I ri-

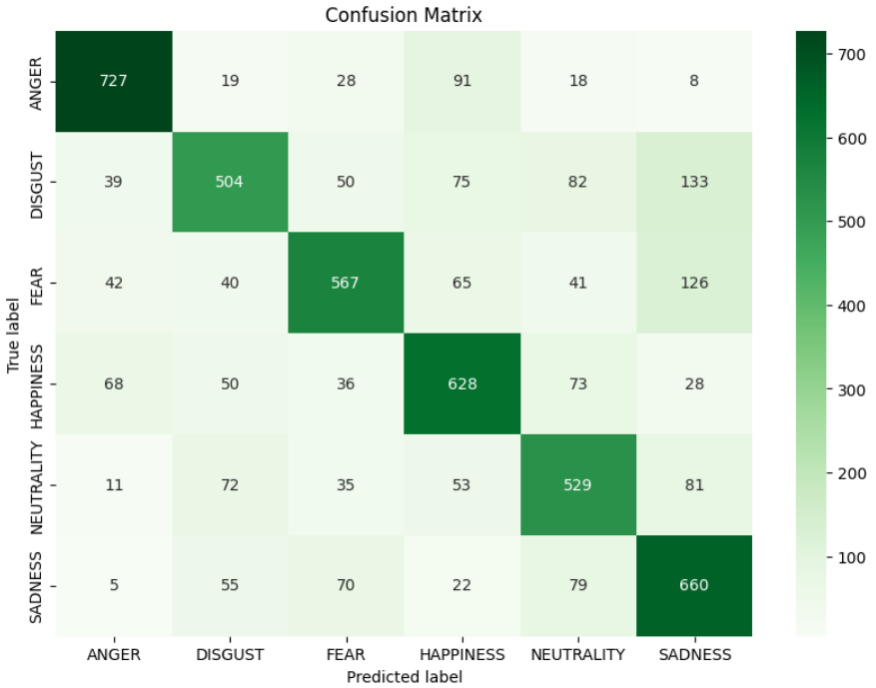


Figura 5.3: Matrice di confusione relativa al Caso 1 dello studio svolto su 6 emozioni

sultati ottenuti, illustrati in Figura 5.4, evidenziano che le due emozioni più frequentemente classificate in modo errato sono *NEUTRALITY* e *FEAR*, con un tasso di misclassificazione rispettivamente del 42.92% e 35.64%. La terza classe maggiormente soggetta ad errore è *DISGUST*, con un tasso del 32.27%. Tali risultati confermano quanto precedentemente emerso dall'analisi del rapporto di classificazione.

5.1.2 Caso 2

Analogamente al Caso 1, l'analisi dei risultati ottenuti nel Caso 2, in cui sono state estratte 10 caratteristiche, è articolata in tre fasi: l'esame delle curve di accuracy e loss, l'analisi del report di classificazione e, infine, la valutazione dei risultati forniti dalla confusion matrix

Percentuale di misclassificazione per emozione:

Emozione	Percentuale (%)
Neutrality	42.92
Fear	35.64
Disgust	32.27
Happiness	28.88
Sadness	25.93
Anger	18.41

Figura 5.4: Dettagli sulla misclassificazione relativi al Caso 1 dello studio svolto su 6 emozioni

e dalla percentuale di misclassificazione.

In Figura 5.5 è possibile osservare l'andamento delle curve di accuracy e loss per il training set e il validation set. Per quanto riguarda l'accuratezza, la curva relativa al training set cresce costantemente fino a stabilizzarsi intorno a 0.75, mentre quella del validation set segue un andamento simile ma si assesta intorno a 0.7, presentando alcune oscillazioni. La differenza tra le due curve diventa più evidente verso le ultime epoche di addestramento, con un divario che inizia a manifestarsi intorno all'epoca 20, suggerendo un possibile inizio di overfitting. Il secondo grafico mostra invece l'andamento della loss: la curva del training set diminuisce in modo regolare e progressivo, segnalando che il modello sta apprendendo efficacemente dai dati. La curva della loss per il validation set, invece, cala nelle prime fasi dell'addestramento, per poi stabilizzarsi con leggere oscillazioni. Questa divergenza tra le due curve potrebbe indicare che il modello sta iniziando a sovradattarsi ai dati di training.

Per approfondire l'analisi delle performance del modello, è stata esaminata la classificazione delle singole emozioni attraverso il report di classificazione. Un primo aspetto da evidenziare è la leggera diminuzione dell'accuratezza complessiva rispetto al Caso 1, con una riduzione di circa l'1%.

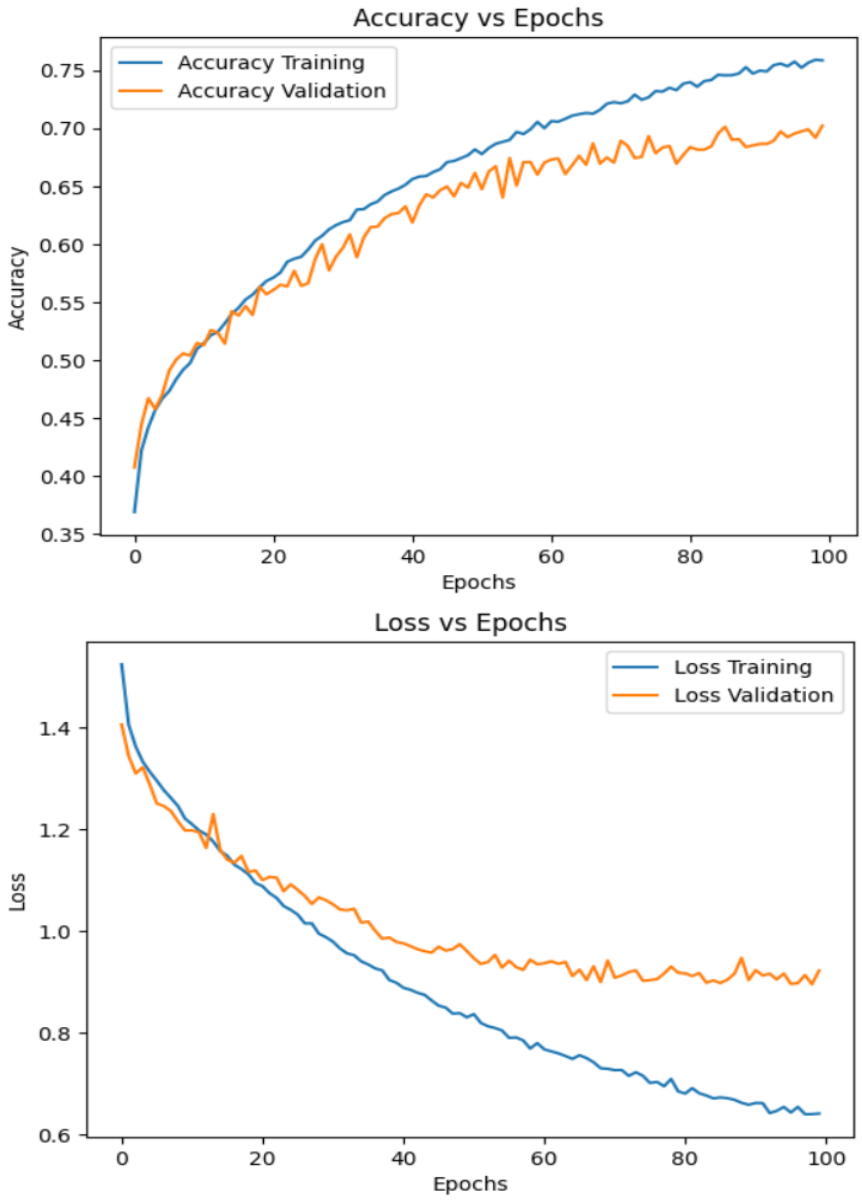


Figura 5.5: Grafici di Accuracy e Loss relativi al Caso 2 dello studio svolto su 6 emozioni

Dall'analisi dettagliata delle classi emerge che l'emozione meglio riconosciuta è *ANGER*, con valori di precision e recall pari a 0.83. Anche *HAPPINESS* viene classificata in modo soddisfacente, con entrambe le metriche che raggiungono 0.7. Per le altre emozioni, si osservano variazioni nei valori di precision, recall e F1-score, segnalando possibili

confusioni con altre classi durante la classificazione. Ancora una volta, l'emozione meno riconosciuta risulta essere *DISGUST*, che presenta valori inferiori rispetto alle altre, con una precisione di 0.59 e un recall di 0.65. Il report completo, con i valori dettagliati delle metriche per ciascuna classe, è mostrato in Figura 5.6.

Rapporto di classificazione:

	precision	recall	f1-score	support
ANGER	0.83	0.83	0.83	891
DISGUST	0.59	0.65	0.61	883
FEAR	0.68	0.63	0.66	881
HAPPINESS	0.70	0.70	0.70	883
NEUTRALITY	0.63	0.67	0.65	781
SADNESS	0.66	0.61	0.64	891
accuracy			0.68	5210
macro avg	0.68	0.68	0.68	5210
weighted avg	0.68	0.68	0.68	5210

Figura 5.6: Report di classificazione relativo al Caso 2 dello studio svolto su 6 emozioni

La valutazione è completata dall'analisi della matrice di confusione e del report di misclassificazione. Come emerso nelle fasi precedenti, *ANGER* e *HAPPINESS* non presentano problemi di classificazione, infatti, come mostrato in Figura 5.7, sono le emozioni con le percentuali di misclassificazione più basse. Diversamente dal Caso 1, *SADNESS* risulta essere l'emozione con la percentuale più alta di classificazioni errate, pari al 38.95%, indicando che è la classe che più frequentemente si confonde con le altre emozioni. A seguire, *FEAR* ha un tasso di misclassificazione del 36.66%, mentre *DISGUST* e *NEUTRALITY* presentano rispettivamente valori del 35.45% e 32.52%.

In Figura 5.8 è possibile osservare la matrice di confusione, che fornisce un'analisi più dettagliata della quantità di campioni erroneamente classificati come appartenenti ad altre classi.

```
Percentuale di misclassificazione per emozione:
--------------------------------------------------
Emozione        Percentuale (%)
--------------------------------------------------
Sadness         38.95
Fear            36.66
Disgust         35.45
Neutrality      32.52
Happiness       30.24
Anger           17.06
--------------------------------------------------
```

Figura 5.7: Dettagli sulla misclassificazione relativi al Caso 2 dello studio svolto su 6 emozioni

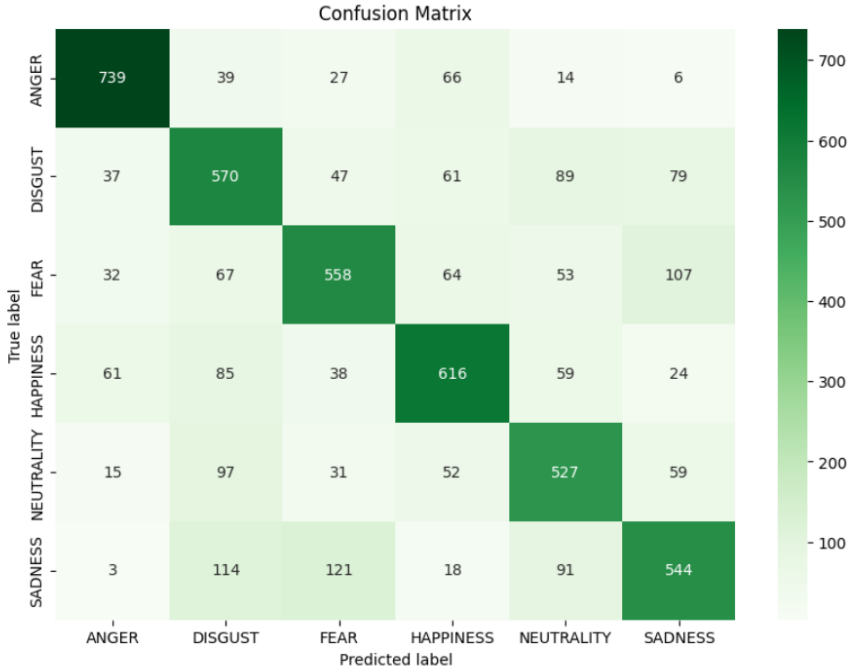


Figura 5.8: Matrice di confusione relativa al Caso 2 dello studio svolto su 6 emozioni

5.1.3 Osservazioni finali

Dalle valutazioni effettuate nei due casi, emerge che il modello mostra buone prestazioni per alcune emozioni, come *ANGER* e *HAPPINESS*, ma incontra difficoltà nel riconoscere emozioni più sottili o simili tra loro. In generale, il modello potrebbe beneficiare di mi-

glioramenti per affrontare meglio le problematiche legate alle emozioni più ambigue. Nelle sezioni successive vengono proposte e discusse in dettaglio alcune soluzioni per affrontare questi aspetti. Per concludere, in Figura 5.9 è riportata una panoramica complessiva delle metriche analizzate nei casi 1 e 2.

	Metriche	Modello con 5 caratteristiche	Modello con 10 caratteristiche
0	Accuracy	69.39%	68.21%
1	Precision	69.62%	68.41%
2	Recall	69.39%	68.21%
3	F1-score	69.30%	68.25%

Figura 5.9: Panoramica complessiva dello studio svolto su 6 emozioni

5.2 6 Emozioni-Caso Specifico

In questa sezione viene analizzata in dettaglio la prima soluzione proposta per affrontare gli aspetti problematici precedentemente evidenziati, che include lo studio del Caso 3, descritto approfonditamente nella Sezione 4.1.2.3. Come già precedentemente svolto, l'analisi è stata condotta esaminando le curve di accuracy e loss, valutando la classificazione complessiva di ciascuna emozione e, infine, analizzando la matrice di confusione insieme al rapporto di misclassificazione.

I due grafici riportati in Figura 5.10 illustrano l'andamento dell'accuracy e della loss in funzione delle epoche durante l'addestramento del modello. Nel primo grafico, la curva relativa al training set mostra un incremento progressivo e costante, raggiungendo circa il 75% al termine delle 100 epoche. La curva del validation set segue un andamento simi-

le, ma con oscillazioni più marcate rispetto alla prima. In particolare, intorno all'epoca 60, si stabilizza su un valore compreso tra il 65% e il 67%. Questo comportamento è un chiaro indicatore dell'inizio dell'overfitting, dove il modello apprende troppo bene i dati di training ma fatica a generalizzare sui dati di validazione. Il secondo grafico rappresenta l'andamento della loss. Entrambe le curve mostrano una diminuzione costante, tuttavia, la curva del validation set si stabilizza intorno a 1.0 tra l'epoca 50 e 60 e presenta leggere oscillazioni rispetto a quella del training set. Queste variazioni suggeriscono una lieve instabilità del modello, probabilmente dovuta a un'eccessiva complessità.

L'accuratezza ottenuta in questa sperimentazione è di circa il 67%. Le prestazioni del modello si sono ridotte rispetto agli studi sui casi 1 e 2, come evidenziato dal report di classificazione delle singole emozioni in Figura 5.11, che mostra anche una generale diminuzione della precisione nel classificare le diverse classi, confermando quanto appena osservato. L'emozione più precisamente classificata rimane *ANGER*, con valori di precisione e recall intorno all'81%. Le emozioni *DISGUST* e *FEAR*, invece, sono quelle con le peggiori performance: per *DISGUST* si riscontrano valori di precisione e recall pari al 56% e 66%, mentre per *FEAR* i valori sono 67% e 60%, rispettivamente per precisione e recall. Questo indica che, sebbene il modello tenda a classificare correttamente queste emozioni, spesso include anche campioni appartenenti ad altre classi. Per quanto riguarda le emozioni *NEUTRALITY* e *SADNESS*, il modello sembra aver migliorato la capacità di riconoscimento, come evidenziato dal bilanciamento tra precisione e recall in entrambi i casi. Infine, nonostante la discrepanza tra precisione e recall nella classe *HAPPINESS*, la classificazione complessiva risulta comunque accettabile.

Le informazioni riportate nelle Figure 5.12 e 5.13 evidenziano che il modello presentato in questa prima soluzione proposta sta operando sui dati in modo meno efficace rispetto

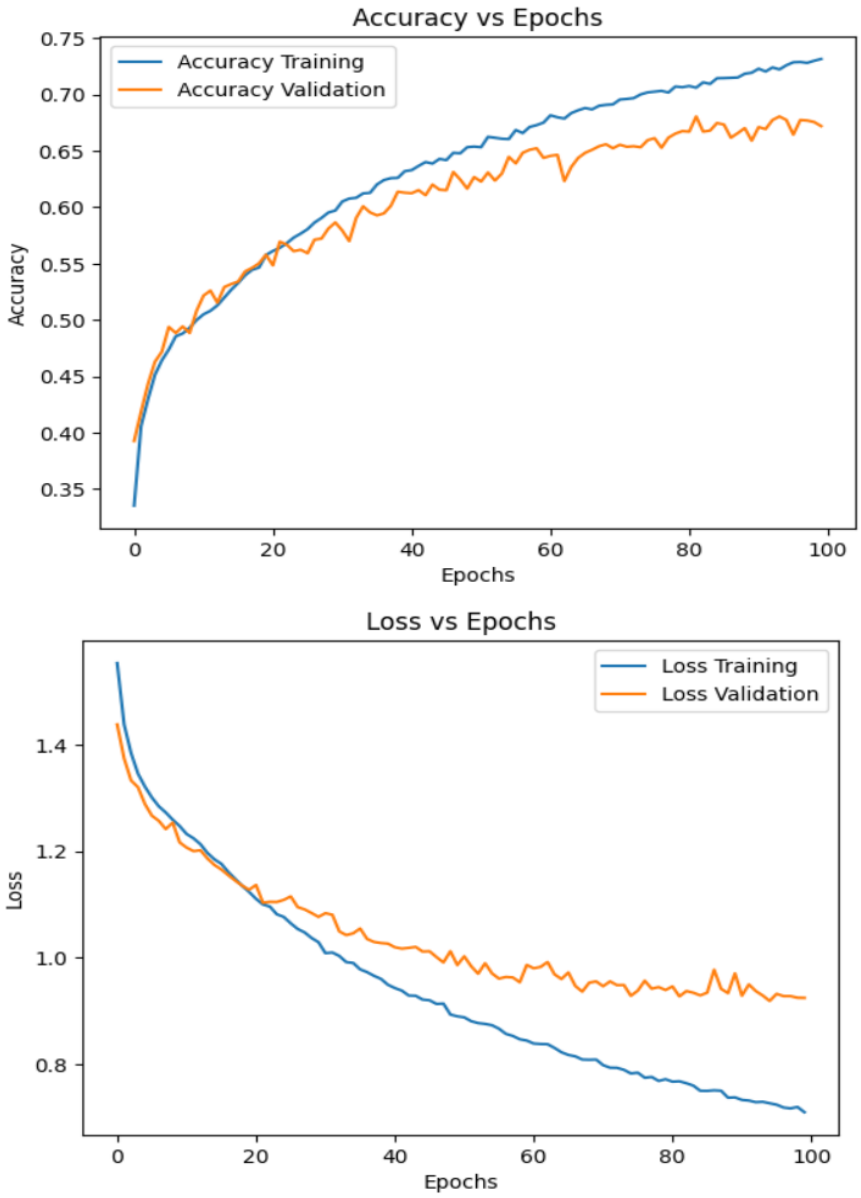


Figura 5.10: Grafici di Accuracy e Loss relativi al Caso 3 dello studio svolto su 6 emozioni

alla sperimentazione iniziale. Questo è confermato dal report di misclassificazione e dalla matrice di confusione, che mostrano una performance inferiore. In particolare, l'emozione *FEAR* risulta quella con la percentuale più alta di misclassificazione, pari al 40,41%. Confrontando questi risultati con quelli ottenuti nelle Sezioni 5.1.1 e 5.1.2, si nota un aumento

Rapporto di classificazione:

	precision	recall	f1-score	support
ANGER	0.81	0.82	0.81	891
DISGUST	0.56	0.66	0.60	883
FEAR	0.67	0.60	0.63	881
HAPPINESS	0.73	0.64	0.68	883
NEUTRALITY	0.62	0.62	0.62	781
SADNESS	0.64	0.66	0.65	891
accuracy			0.67	5210
macro avg	0.67	0.67	0.67	5210
weighted avg	0.67	0.67	0.67	5210

Figura 5.11: Report di classificazione relativo al Caso 3 dello studio svolto su 6 emozioni

di circa il 5% nell'errore di classificazione per questa classe, rispetto ai valori registrati precedentemente, che erano rispettivamente pari al 35,64% e al 36,66%. Analizzando tutti i singoli casi, il modello non sembra aver apportato miglioramenti rispetto alla condizione di partenza, risultando attualmente il modello con le performance più basse.

Percentuale di misclassificazione per emozione:

Emozione	Percentuale (%)
Fear	40.41
Neutrality	37.52
Happiness	35.56
Disgust	34.09
Sadness	33.56
Anger	18.41

Figura 5.12: Dettagli sulla misclassificazione relativi al Caso 3 dello studio svolto su 6 emozioni

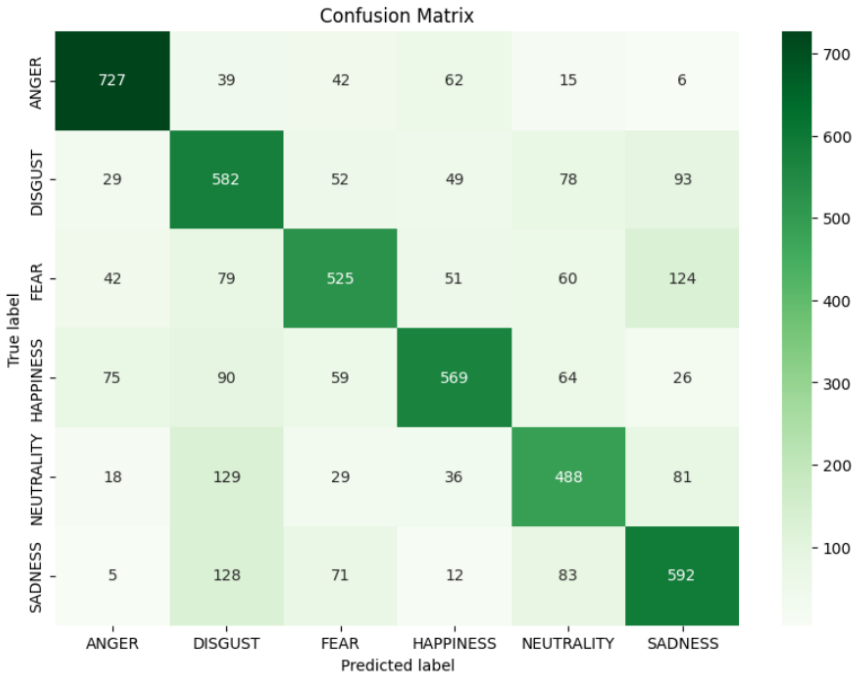


Figura 5.13: Matrice di confusione relativa al Caso 3 dello studio svolto su 6 emozioni

5.3 4 Emozioni

Le soluzioni proposte in questa sezione prevedono l'eliminazione delle due classi con la maggiore percentuale di misclassificazione, evidenziate nelle Sezioni 5.1.1 e 5.1.2.

5.3.1 Caso 1

Prima di procedere con la preparazione dei dati per l'addestramento del modello, è stato effettuato un semplice passaggio per rimuovere le colonne relative alle etichette *FEAR* e *NEUTRALITY*. Questo è stato fatto filtrando il DataFrame: la funzione **isin()** verifica se i valori della colonna *label* appartengono all'elenco *['FEAR', 'NEUTRALITY']*, mentre il simbolo $\sim$ viene utilizzato per negare il risultato, selezionando così solo le righe che non corrispondono a queste etichette.

```
# Rimozione delle classi misclassificate

base_df = base_df[~base_df['label'].isin(['FEAR', 'NEUTRALITY'])]
```

Figura 5.14: Rimozione delle emozione *FEAR* e *NEUTRALITY* per l'analisi del Caso 1 dello studio svolto su 4 emozioni

Dopo aver rimosso le etichette ambigue, si procede con l'analisi delle curve di accuracy e loss. La curva dell'accuratezza sul set di training mostra una crescita costante, senza particolari anomalie, fino a raggiungere circa il 90% al termine dell'addestramento. La curva relativa al set di validazione, invece, presenta leggere oscillazioni e si stabilizza attorno all'80%. Come osservato anche nelle sperimentazioni precedenti, il modello si adatta bene ai dati di training, ma mostra una capacità di generalizzazione inferiore sui dati di validazione. Per quanto riguarda la loss, nel training set si osserva una diminuzione continua, segno che il modello sta apprendendo in modo efficace. Anche la loss sul set di validazione segue inizialmente un andamento simile, ma mostra delle oscillazioni e smette di diminuire intorno al valore di 0.6. Questo comportamento suggerisce l'inizio di un fenomeno di overfitting. Le curve di accuracy e loss sono riportate nella Figura 5.15.

Il valore complessivo di accuratezza dell'82% indica che il modello ha fornito previsioni di buona qualità. La classe *ANGER* si conferma la più precisamente classificata, con una precisione che indica che il 90% delle previsioni per questa classe sono state corrette. Il recall dell'87% suggerisce che la classe è stata riconosciuta correttamente nell'87% dei casi, mentre l'F1-score dell'88% conferma una solida performance nella previsione. Al contrario, la classe peggio classificata è *DISGUST*, con un F1-score del 77%, che, pur riflettendo una prestazione soddisfacente, non è completamente eccellente. Per quanto riguarda le classi *HAPPINESS* e *SADNESS*, entrambe sono state previste con ottimi risultati: *HAPPINESS*

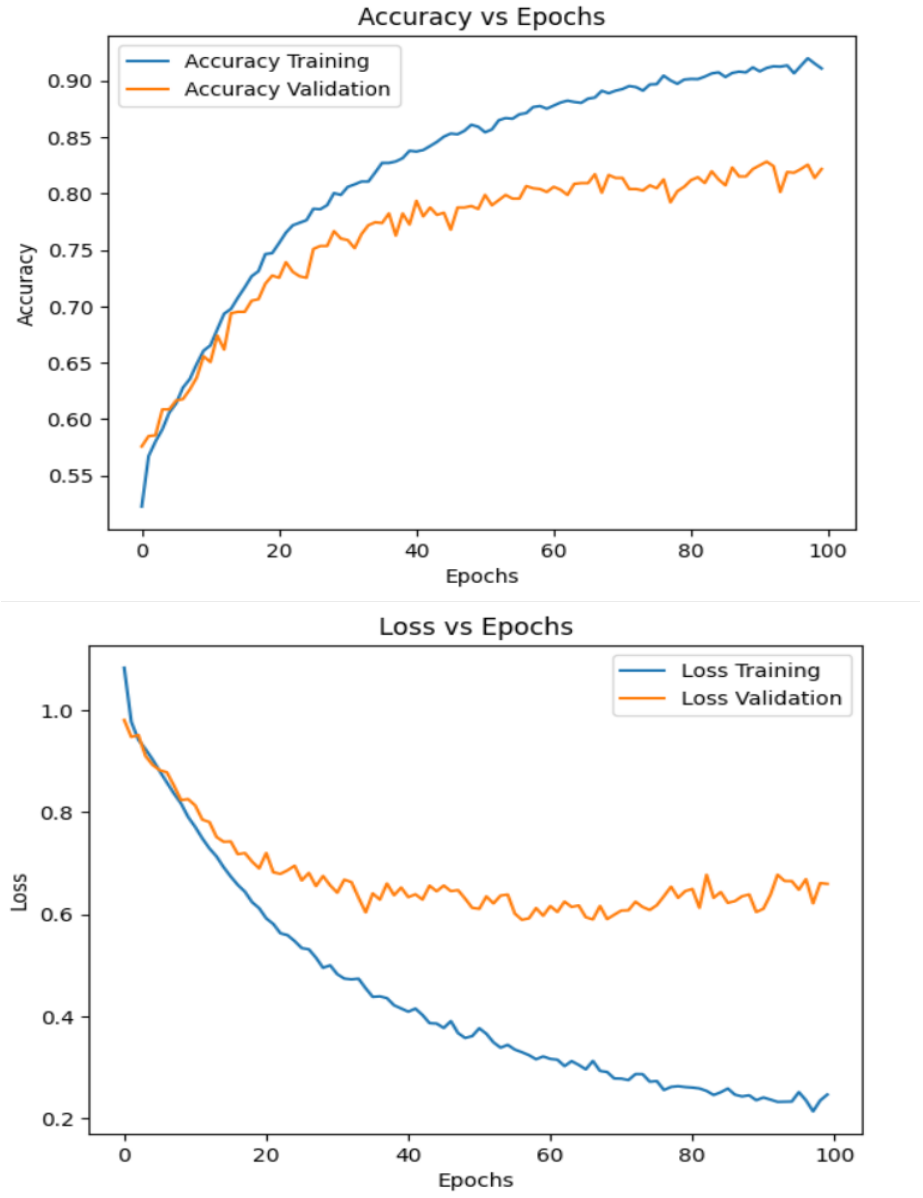


Figura 5.15: Grafici di Accuracy e Loss relativi al Caso 1 dello studio svolto su 4 emozioni

presenta una precisione di 0.81 e un recall di 0.77, mentre *SADNESS* ha una precisione di 0.83 e un recall di 0.84. Sebbene vi siano lievi misclassificazioni in alcuni campioni, entrambe le classi sono state previste con successo. In generale, il modello mostra una performance complessivamente positiva.

Rapporto di classificazione:

	precision	recall	f1-score	support
ANGER	0.90	0.87	0.88	908
DISGUST	0.75	0.79	0.77	882
HAPPINESS	0.81	0.77	0.79	898
SADNESS	0.83	0.84	0.83	871
accuracy			0.82	3559
macro avg	0.82	0.82	0.82	3559
weighted avg	0.82	0.82	0.82	3559

Figura 5.16: Report di classificazione relativo al Caso 1 dello studio svolto su 4 emozioni

Come supporto a quanto appena affermato, nelle Figure 5.17 e 5.18 si osserva come il tasso di misclassificazione per tutte le classi sia sceso al di sotto del 30%. Questi risultati suggeriscono che, mediante la rimozione delle classi più problematiche, il modello è in grado di distinguere in modo soddisfacente le emozioni presenti nel dataset.

Percentuale di misclassificazione per emozione:

Emozione	Percentuale (%)
Fear	22.83
Disgust	20.75
Happiness	15.96
Anger	13.00

Figura 5.17: Dettagli sulla misclassificazione relativi al Caso 1 dello studio svolto su 4 emozioni

5.3.2 Caso 2

Come nel Caso 1, anche per il caso con l'estrazione di 10 caratteristiche è stato necessario rimuovere le classi che presentavano problematiche. In questo caso, le emozioni escluse sono state *FEAR* e *SADNESS*, mediante un filtraggio effettuato in modo analogo a quanto

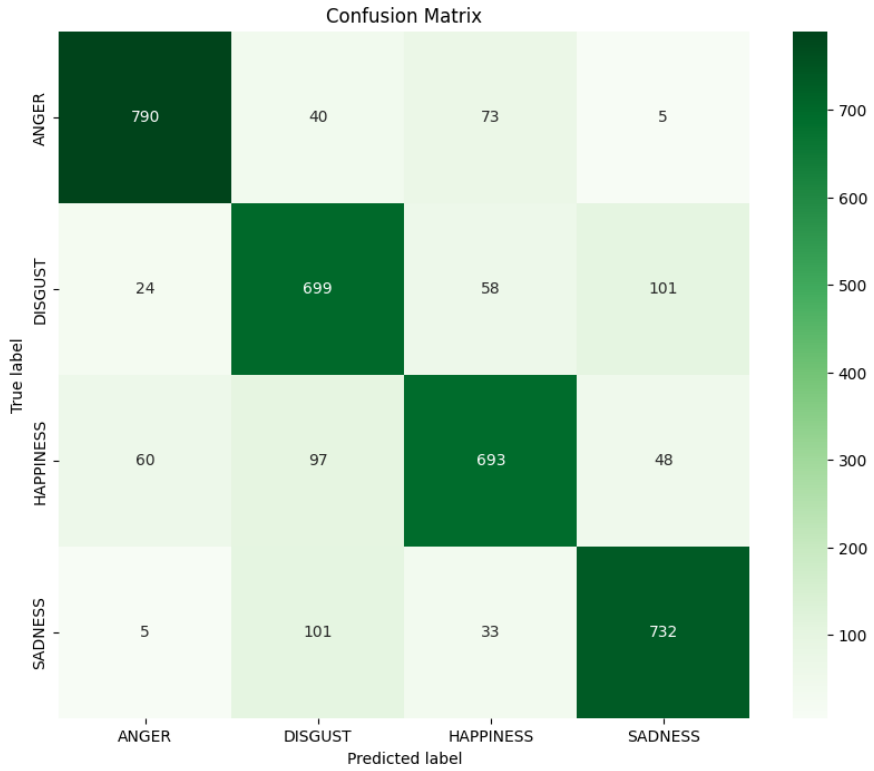


Figura 5.18: Matrice di confusione relativa al Caso 1 dello studio svolto su 4 emozioni

fatto precedentemente.

```
# Rimozione delle classi misclassificate

total_df = total_df[~total_df['label'].isin(['FEAR', 'SADNESS'])]
```

Figura 5.19: Rimozione delle emozione *FEAR* e *SADNESS* per l'analisi del Caso 2 dello studio svolto su 4 emozioni

I grafici in Figura 5.20 mostrano l'andamento dell'accuratezza e della loss del modello durante l'addestramento. La curva di accuratezza del training segue una crescita costante, avvicinandosi a un valore di 0.9, mentre quella della validazione presenta oscillazioni e si stabilizza su valori inferiori. Per quanto riguarda la loss, nel training set si osserva una

diminuzione continua, mentre nel set di validazione si notano oscillazioni e un'inversione di tendenza con un leggero aumento. Le discrepanze tra le curve di training e validazione indicano chiaramente l'insorgere di overfitting.

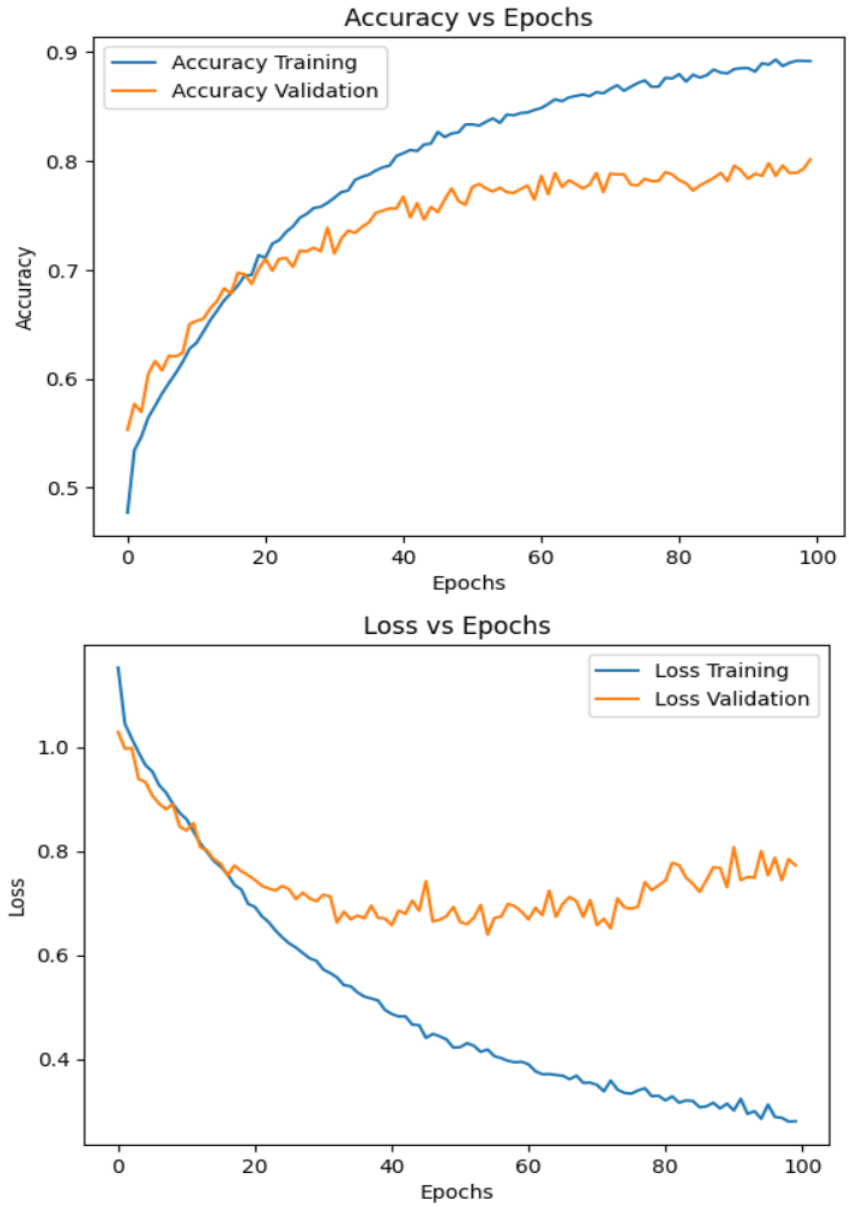


Figura 5.20: Grafici di Accuracy e Loss relativi al Caso 2 dello studio svolto su 4 emozioni

Nel complesso, con un'accuratezza del 78%, il modello mostra buone performance. In particolare, l'emozione *ANGER* risulta essere la meglio riconosciuta, con una precisione del 90% e una recall dell'80%. Sebbene la recall sia inferiore rispetto al Caso 1, il valore dell'F1-score offre comunque un buon bilanciamento tra le due metriche. Le classi *HAPPINESS* e *NEUTRALITY*, con F1-score di 0.76 e 0.77 rispettivamente, possono essere considerate ben classificate, sebbene i valori di precisione e recall suggeriscano la presenza di alcune istanze per cui la classificazione non è stata corretta. Infine, la classe *DISGUST* si conferma come la più difficilmente classificata. Nonostante i valori di precisione e recall, rispettivamente del 72% e del 77%, siano comunque soddisfacenti, il numero di campioni mal classificati è il più alto per questa classe. Un quadro completo delle metriche di classificazione è illustrato nella Figura 5.21.

Rapporto di classificazione:

	precision	recall	f1-score	support
ANGER	0.90	0.80	0.85	904
DISGUST	0.72	0.77	0.74	877
HAPPINESS	0.77	0.75	0.76	880
NEUTRALITY	0.75	0.80	0.77	769
accuracy			0.78	3430
macro avg	0.78	0.78	0.78	3430
weighted avg	0.79	0.78	0.78	3430

Figura 5.21: Report di classificazione relativo al Caso 2 dello studio svolto su 4 emozioni

Per completare l'analisi ed a supporto delle osservazioni precedenti, la matrice di confusione (Figura 5.22) e il report sulle misclassificazioni (Figura 5.23) confermano che il modello presenta buone performance. Analogamente al Caso 1, la percentuale complessiva di classificazioni errate è significativamente inferiore al 30%. L'emozione che risulta essere maggiormente misclassificata è *NEUTRALITY*, con il 24,89%, seguita da *DISGUST*

(22,58%), *ANGER* (20,24%) e *HAPPINESS* (19,77%).

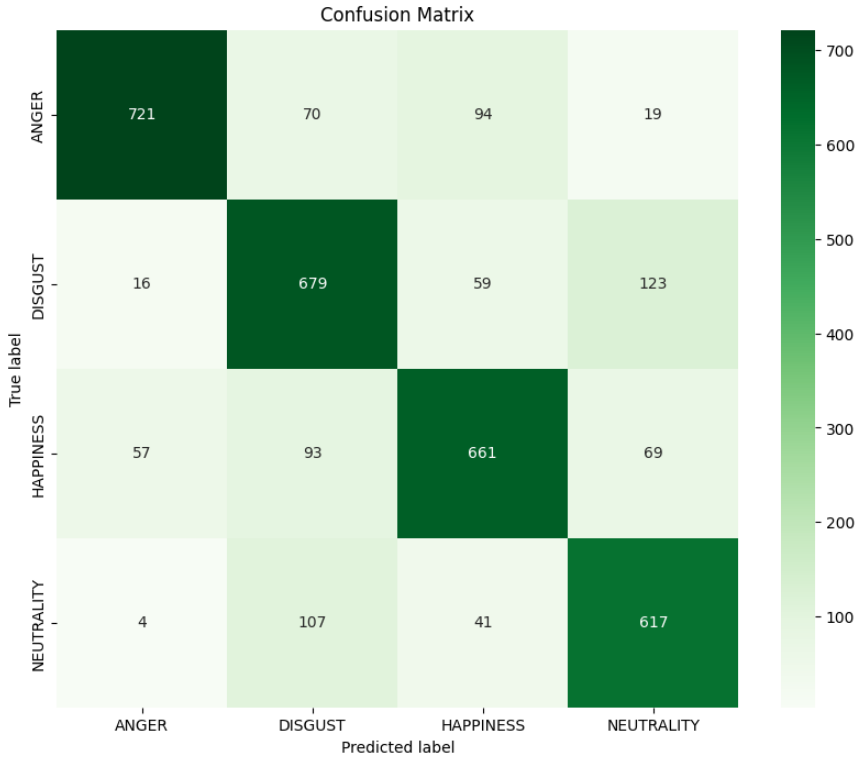


Figura 5.22: Matrice di confusione relativa al Caso 2 dello studio svolto su 4 emozioni

```
Percentuale di misclassificazione per emozione:
--------------------------------------------------
Emozione        Percentuale (%)
--------------------------------------------------
Neutrality      24.89
Disgust         22.58
Anger           20.24
Happiness       19.77
--------------------------------------------------
```

Figura 5.23: Dettagli sulla misclassificazione relativi al Caso 2 dello studio svolto su 4 emozioni

5.3.3 Osservazioni finali

Dalle valutazioni effettuate nei due casi, emerge che il modello ha mostrato buone prestazioni rispetto alla sperimentazione originale. La rimozione delle classi più problematiche ha migliorato la capacità del modello di distinguere correttamente tra le classi rimanenti, portando così a risultati ottimali. Per concludere, in Figura 5.24 è riportata una panoramica complessiva delle metriche analizzate nei casi 1 e 2.

	Metriche	Modello con 5 caratteristiche	Modello con 10 caratteristiche
0	Accuracy	81.88%	78.08%
1	Precision	82.04%	78.65%
2	Recall	81.88%	78.08%
3	F1-score	81.92%	78.22%

Figura 5.24: Panoramica complessiva dello studio svolto su 4 emozioni

5.4 4 Emozioni-Ottimizzazione Bayesiana

Questa sezione presenta l'ultima soluzione sviluppata per gestire le dinamiche emerse dall'analisi iniziale. La proposta prevede l'utilizzo del caso di studio basato su 4 emozioni, affiancato dall'ottimizzazione bayesiana degli iperparametri. A differenza delle analisi precedenti, questa sperimentazione non è altrettanto esaustiva, poiché la valutazione si limita a confrontare i valori di accuratezza ottenuti nelle diverse prove.

5.4.1 Iperparametri e Configurazione del Tuner

Prima di avviare la fase di valutazione, è stato necessario apportare alcune modifiche all'implementazione del modello. Come mostrato in Figura 5.25, il modello è stato riscritto sotto forma di funzione per poter essere utilizzato dal tuner. Pur mantenendo invariata la sua struttura rispetto alle versioni precedenti, in questa implementazione vengono messi in evidenza gli iperparametri che saranno oggetto di ottimizzazione: **filters_1**, **dropout_1**, **filters_2**, **dropout_2** e **dense_units**. Questi iperparametri influenzano direttamente le prestazioni del modello in termini di capacità di apprendimento, generalizzazione e robustezza. *filters_1* e *filters_2* determinano il numero di filtri nei livelli convoluzionali: ottimizzare questo parametro è essenziale, poiché un numero troppo basso potrebbe causare la perdita di informazioni rilevanti, mentre un numero eccessivo potrebbe aumentare il rischio di overfitting ed allungare i tempi di addestramento. Trovare un equilibrio tra complessità della rete e capacità di generalizzazione è quindi cruciale. Gli iperparametri *dropout_1* e *dropout_2* regolano la percentuale di neuroni disattivati casualmente per ridurre la dipendenza da connessioni specifiche. Un valore troppo basso aumenta il rischio di overfitting, mentre uno eccessivamente alto potrebbe portare ad underfitting, riducendo la capacità del modello di apprendere dai dati. Infine, *dense_units* controlla la capacità del livello completamente connesso di combinare le caratteristiche apprese dai layer convoluzionali. Un valore troppo basso potrebbe rendere il modello poco espressivo, mentre uno troppo alto potrebbe favorire l'overfitting. Anche in questo caso, è fondamentale individuare il valore ottimale per massimizzare le capacità predittive del modello.

Definiti gli iperparametri da ottimizzare, si procede con la configurazione del tuner. Viene invocata la funzione **BayesianOptimization**, alla quale vengono passati tre parametri principali: la funzione che definisce il modello, l'obiettivo da massimizzare, **val_accuracy**

```python
# Creazione del modello

def build_model(hp):
    model = Sequential()
    model.add(Conv1D(
        filters=hp.Int('filters_1', min_value=32, max_value=128, step=32),
        kernel_size=3,
        activation='relu',
        input_shape=(X_train_res_5.shape[1], X_train_res_5.shape[2])
    ))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(hp.Float('dropout_1', min_value=0.1, max_value=0.5, step=0.1)))
    model.add(Conv1D(
        filters=hp.Int('filters_2', min_value=64, max_value=256, step=64),
        kernel_size=3,
        activation='relu'
    ))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(hp.Float('dropout_2', min_value=0.1, max_value=0.5, step=0.1)))
    model.add(Flatten())
    model.add(Dense(hp.Int('dense_units', min_value=32, max_value=128, step=32), activation='relu'))
    model.add(Dense(y_train_enc_5.shape[1], activation='softmax'))

    model.compile(
        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Figura 5.25: Esempio di implementazione del modello adattato per la configurazione del tuner

in questo caso, e il numero di tentativi da eseguire con diverse combinazioni di iperparametri, fissato a **5**. Operativamente, il tuner inizia scegliendo casualmente una configurazione degli iperparametri ed addestra il modello su di essa. Al termine dell'addestramento, analizza i risultati e, basandosi sulle informazioni ottenute, la Bayesian Optimization seleziona la configurazione successiva in modo guidato, cercando di migliorare le prestazioni. Questo processo si ripete per tutti e cinque i tentativi. Al termine dell'ottimizzazione, il tuner restituisce la combinazione di iperparametri che ha ottenuto le migliori performance sul validation set.

5.4.2 Caso 1

Completata la configurazione del tuner, si è proceduto con l'avvio dell'addestramento utilizzando la prima configurazione di iperparametri. In Figura 5.27 sono riportati i risultati

```
# Configurazione del tuner

tuner5 = BayesianOptimization(
    build_model,
    objective='val_accuracy',
    max_trials=5,
    directory='model5_tuning',
    project_name='emotion_recognition5'
)
```

Figura 5.26: Esempio di configurazione del tuner

di ciascun tentativo effettuato dal tuner. Nella tabella mostrata, la colonna **VALUE** indica i valori degli iperparametri adottati in ciascun trial, la colonna **BEST VALUE SO FAR** mostra i valori che, fino a quel momento, hanno prodotto le migliori performance, mentre la colonna **HYPERPARAMETERS** riporta gli iperparametri oggetto di ottimizzazione. Osservando attentamente le figure, si può notare che la configurazione ottimale è rimasta invariata durante tutto il processo di tuning, corrispondendo a quella del primo tentativo. Questo è ulteriormente confermato dalle informazioni contenute in Figura 5.28, dove viene mostrata la combinazione finale di iperparametri selezionata dall'ottimizzatore.

5.4.3 Caso 2

Nel Caso 2, il procedimento adottato rimane invariato. In Figura 5.29 sono presentati i risultati dei 5 tentativi, mentre in Figura 5.30 è visibile la configurazione finale determinata dall'ottimizzatore. Anche in questa circostanza, la configurazione impostata prima del tentativo iniziale si è rivelata quella ottimale per il modello.

5.4.4 Osservazioni finali

Dalle valutazioni effettuate sui due casi, si osserva un miglioramento significativo rispetto allo studio condotto su 4 emozioni senza l'uso dell'ottimizzatore. I risultati ottenuti per la validation accuracy sono circa 85% per il Caso 1 e 83% per il Caso 2. Nonostante il

```
Trial 1 Complete [00h 31m 52s]
val_accuracy: 0.8510203957557678

Best val_accuracy So Far: 0.8510203957557678
Total elapsed time: 00h 31m 52s

Search: Running Trial #2

Value             |Best Value So Far |Hyperparameter
128               |32                |filters_1
0.3               |0.1               |dropout_1
192               |128               |filters_2
0.5               |0.3               |dropout_2
96                |96                |dense_units
```

(a) Trial 1-2

```
Trial 2 Complete [01h 54m 47s]
val_accuracy: 0.8455782532691956

Best val_accuracy So Far: 0.8510203957557678
Total elapsed time: 02h 26m 39s

Search: Running Trial #3

Value             |Best Value So Far |Hyperparameter
128               |32                |filters_1
0.1               |0.1               |dropout_1
64                |128               |filters_2
0.3               |0.3               |dropout_2
64                |96                |dense_units
```

(b) Trial 2-3

```
Trial 3 Complete [00h 59m 16s]
val_accuracy: 0.8489795923233032

Best val_accuracy So Far: 0.8510203957557678
Total elapsed time: 03h 25m 55s

Search: Running Trial #4

Value             |Best Value So Far |Hyperparameter
32                |32                |filters_1
0.3               |0.1               |dropout_1
256               |128               |filters_2
0.4               |0.3               |dropout_2
96                |96                |dense_units
```

(c) Trial 3-4

```
Trial 4 Complete [01h 02m 38s]
val_accuracy: 0.8170068264007568

Best val_accuracy So Far: 0.8510203957557678
Total elapsed time: 04h 28m 33s

Search: Running Trial #5

Value             |Best Value So Far |Hyperparameter
96                |32                |filters_1
0.3               |0.1               |dropout_1
128               |128               |filters_2
0.4               |0.3               |dropout_2
32                |96                |dense_units
```

(d) Trial 4-5

```
Trial 5 Complete [01h 00m 55s]
val_accuracy: 0.781632661819458

Best val_accuracy So Far: 0.8510203957557678
Total elapsed time: 05h 29m 28s
```

(e) Trial 5

Figura 5.27: Tentativi dell'ottimizzazione bayesiana relativi al Caso 1 dello studio svolto su 4 emozioni

miglioramento nella performance, il carico computazionale risulta proporzionale al numero di tentativi effettuati dal tuner: in termini di tempo, infatti, l'ottimizzatore ha impiegato 5 ore e 29 minuti per il tuning degli iperparametri nel Caso 1, mentre ha richiesto 7 ore e 33 minuti per il caso con 10 caratteristiche estratte.

```
# Analisi dei migliori iperparametri

best_hps5 = tuner5.get_best_hyperparameters(num_trials=1)[0]
print(best_hps5.values)
```

```
{'filters_1': 32, 'dropout_1': 0.1, 'filters_2': 128, 'dropout_2': 0.30000000000000004, 'dense_units': 96}
```

Figura 5.28: Migliore configurazione di iperparametri relativa al Caso 1 dello studio svolto su 4 emozioni

5.5 Considerazioni Conclusive

Come riportato nel Capitolo 1, in letteratura sono presenti diversi studi che utilizzano lo stesso dataset impiegato in questa Tesi. La sperimentazione che ha ottenuto i risultati migliori in questo lavoro ha raggiunto un'accuratezza dell'85%, grazie ad un'analisi condotta su quattro emozioni ed ottimizzata mediante l'Ottimizzazione Bayesiana.

Questo risultato supera quelli riportati in letteratura, dove l'approccio descritto in [7] raggiunge un'accuratezza dell'84%, mentre il metodo proposto in [8] si ferma al 61%.

```
Trial 1 Complete [01h 59m 32s]
val_accuracy: 0.8299319744110107

Best val_accuracy So Far: 0.8299319744110107
Total elapsed time: 01h 59m 32s

Search: Running Trial #2

Value             |Best Value So Far |Hyperparameter
128               |96                |filters_1
0.3               |0.1               |dropout_1
256               |256               |filters_2
0.4               |0.5               |dropout_2
96                |96                |dense_units
```

(a) Trial 1-2

```
Trial 2 Complete [02h 06m 23s]
val_accuracy: 0.7925170063972473

Best val_accuracy So Far: 0.8299319744110107
Total elapsed time: 04h 05m 55s

Search: Running Trial #3

Value             |Best Value So Far |Hyperparameter
96                |96                |filters_1
0.5               |0.1               |dropout_1
64                |256               |filters_2
0.1               |0.5               |dropout_2
128               |96                |dense_units
```

(b) Trial 2-3

```
Trial 3 Complete [00h 48m 21s]
val_accuracy: 0.8074830174446106

Best val_accuracy So Far: 0.8299319744110107
Total elapsed time: 04h 54m 16s

Search: Running Trial #4

Value             |Best Value So Far |Hyperparameter
96                |96                |filters_1
0.3               |0.1               |dropout_1
256               |256               |filters_2
0.1               |0.5               |dropout_2
32                |96                |dense_units
```

(c) Trial 3-4

```
Trial 4 Complete [01h 35m 00s]
val_accuracy: 0.7489795684814453

Best val_accuracy So Far: 0.8299319744110107
Total elapsed time: 06h 29m 16s

Search: Running Trial #5

Value             |Best Value So Far |Hyperparameter
32                |96                |filters_1
0.2               |0.1               |dropout_1
256               |256               |filters_2
0.2               |0.5               |dropout_2
128               |96                |dense_units
```

(d) Trial 4-5

```
Trial 5 Complete [01h 04m 27s]
val_accuracy: 0.8040816187858582

Best val_accuracy So Far: 0.8299319744110107
Total elapsed time: 07h 33m 43s
```

(e) Trial 5

Figura 5.29: Tentativi dell'ottimizzazione bayesiana relativi al Caso 2 dello studio svolto su 4 emozioni

```
{'filters_1': 96, 'dropout_1': 0.1, 'filters_2': 256, 'dropout_2': 0.5, 'dense_units': 96}
```

Figura 5.30: Migliore configurazione di iperparametri relativa al Caso 2 dello studio svolto su 4 emozioni

Conclusioni e sviluppi futuri

Il lavoro svolto ha permesso di studiare e riconoscere emozioni quali felicità, tristezza, rabbia, paura, disgusto e neutralità a partire da file audio in formato .wav. A tal fine, sono state estratte le caratteristiche fondamentali per il riconoscimento emotivo, addestrando una rete neurale convoluzionale 1D sul dataset CREMA-D.

Per migliorare le prestazioni del modello, sono state impiegate tecniche come la Data Augmentation, che ha ampliato il dataset, e l'Ottimizzazione Bayesiana, utilizzata per affrontare il problema delle classi ambigue. Il modello, infatti, ha evidenziato difficoltà nel distinguere emozioni come paura, disgusto e neutralità. Per mitigare questa problematica, sono state sperimentate diverse strategie, portando a risultati complessivamente soddisfacenti.

L'analisi delle varie sperimentazioni ha evidenziato come l'approccio basato sulla Bayesian Optimization sia il più efficace tra quelli testati, offrendo prestazioni ottimali nell'ambito del riconoscimento delle emozioni da segnali vocali. D'altra parte, i test condotti su un modello specializzato nell'analisi di 6 emozioni con l'estrazione di caratteristiche specifiche (Sezione 5.2) non hanno mostrato un miglioramento delle prestazioni rispetto alla sperimentazione iniziale.

In futuro, questo progetto potrà essere ampliato e perfezionato attraverso diverse strategie. Un aspetto cruciale riguarda la riduzione dell'overfitting, che potrebbe essere affron-

tata sia mediante un'ottimizzazione della struttura del modello, rendendolo più robusto, sia intervenendo sulla sua complessità. Un'ulteriore direzione di miglioramento riguarda l'estrazione delle caratteristiche dai file audio. In particolare, sarebbe opportuno sviluppare tecniche di feature engineering mirate ad individuare le caratteristiche più adatte per il riconoscimento di ciascuna emozione, così da ridurre le ambiguità emerse in questo studio. Inoltre, l'utilizzo di un numero maggiore di dataset differenti potrebbe rendere il modello più generalizzabile, evitando che si adatti eccessivamente ad un singolo tipo di dati.

Possibili estensioni del lavoro di ricerca includono l'applicazione del modello a domini più ampi, come lo studio di dataset multimodali che integrano sia dati audio che video. Un'ulteriore evoluzione potrebbe consistere nell'integrazione di parametri biometrici, quali frequenza cardiaca e respirazione, per una valutazione più completa dello stato emotivo.

Le potenziali applicazioni di questo sistema sono numerose. Nel settore del customer service, l'analisi automatica delle emozioni potrebbe migliorare e personalizzare l'esperienza del cliente, identificando tempestivamente situazioni di insoddisfazione. Un altro ambito di utilizzo riguarda i chatbot, che potrebbero adattare le risposte in base alle emozioni percepite, migliorando l'interazione con gli utenti. Infine, il modello potrebbe trovare impiego nei servizi pubblici di sicurezza e sorveglianza, ad esempio rilevando segnali di paura nelle chiamate di emergenza per allertare le autorità o monitorando suoni e voci in ambienti pubblici per prevenire situazioni di pericolo.

Bibliografia

[1] W. Q. Zheng, J. S. Yu, and Y. X. Zou. An experimental study of speech emotion recognition based on deep convolutional neural networks. *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 827–831, 2015.

[2] W. Lim, D. Jang, and T. Lee. Speech emotion recognition using convolutional and recurrent neural networks. *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–4, 2016.

[3] A. Satt, S. Rozenberg, and R. Hoory. Efficient emotion recognition from speech using deep learning on spectrograms. *Interspeech 2017*, pages 1089–1093, 2017.

[4] Z. Jianfeng, M. Xia, and C. Lijiang. Speech emotion recognition using deep 1d 2d cnn lstm networks. *Biomedical Signal Processing and Control*, 47:312–323, 2019.

[5] D. Issa, M. F. Demirci, and A. Yazici. Speech emotion recognition with deep convolutional neural networks. *Biomedical Signal Processing and Control*, 59, 2020.

[6] A. Hadhami and B. A. Yassine. Speech emotion recognition with deep learning. *Procedia Computer Science*, 176:251–260, 2020.

[7] M. Gokilavani, H. Katakam, SK. A. Basheer, and PVVS Srinivas. Ravdness, crema-d, tess based algorithm for emotion recognition using speech. *2022 4th International*

Conference on Smart Systems and Inventive Technology (ICSSIT), pages 1625–1631, 2022.

[8] P. Kumar, R. Khera, A. Grover, and K. Sharma. Exploring speech emotion recognition with mlp classifier: A comprehensive study on the crema-d dataset. *2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST)*, pages 542–547, 2024.