Ravdess Emotional Speech Audio - Multipli Approcci di Deep Learning per l'Analisi dei Segnali Audio

Ravenda Federico, 829449[1] **and Longobucco Andrea, 825016**[2]

[1] *Università degli Studi Milano-Bicocca, CdLM Scienze Statistiche ed Economiche, STAT*
[2] *Università degli Studi Milano-Bicocca, CdLM Scienze Statistiche ed Economiche, STAT*

Compiled January 17, 2023

Nella presente trattazione sono stati studiati multipli approcci per lo studio e l'analisi dei segnali audio. Il dataset in esame è una raccolta di 1440 file audio con estensione .wav dal nome *Ravdess Emotional Speech Audio*, disponibile sul sito https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio, pubblicato per la prima volta nel 2019 proprio sulla piattaforma *Kaggle*.

L'obiettivo della trattazione è quello di esplorare diverse tecniche di *Deep Learning* al fine di fornire un overview sulle applicazioni di queste nell'ambito dell'analisi dei segnali audio.
Verranno inoltre approfonditi approcci di *preprocessing* particolarmente performanti in ambito di audio recognition.

Ravdess Emotional Speech Audio - Multipli Approcci di Deep Learning per l'Analisi dei Segnali Audio

Ravenda Federico, 829449[1] **and Longobucco Andrea, 825016**[2]

[1] *Università degli Studi Milano-Bicocca, CdLM Scienze Statistiche ed Economiche, STAT*
[2] *Università degli Studi Milano-Bicocca, CdLM Scienze Statistiche ed Economiche, STAT*

Compiled January 17, 2023

Nella presente trattazione sono stati studiati multipli approcci per lo studio e l'analisi dei segnali audio. Il dataset in esame è una raccolta di 1440 file audio con estensione .wav dal nome *Ravdess Emotional Speech Audio*, disponibile sul sito https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio, pubblicato per la prima volta nel 2019 proprio sulla piattaforma *Kaggle*.

L'obiettivo della trattazione è quello di esplorare diverse tecniche di *Deep Learning* al fine di fornire un overview sulle applicazioni di queste nell'ambito dell'analisi dei segnali audio.
Verranno inoltre approfonditi approcci di *preprocessing* particolarmente performanti in ambito di audio recognition.

CONTENTS

1. INTRODUZIONE

Cos'è il suono?

"Il suono è un cambiamento di pressione delle molecole d'aria creato da un oggetto vibrante"

Il cambiamento di pressione di un oggetto vibrante crea *un'onda*. Le *onde sonore* possono essere descritte dalla *frequenza del suono* e dalla dimensione delle onde sonore stesse. La *frequenza del suono* non è altro che il numero che definisce quante volte al secondo oscilla un'onda sonora.

In sostanza, un *segnale sonoro* è prodotto dalle variazioni della pressione dell'aria. Si può misurare l'intensità delle variazioni di pressione e tracciare tali misurazioni nel tempo.

Fino a pochi anni fa, prima dell'avvento delle più recenti architetture di ***Deep Learning*** (*CNN, RNN*), le applicazioni di Machine Learning in *Computer Vision* si affidavano a tradizionali tecniche di elaborazione dei dati per eseguire *Features Engineering*; ad esempio, per quanto riguarda l'elaborazione delle immagini, era molto comune la creazione di *Features Hand-Crafted* per rilevare gli angoli, i bordi e le facce dei soggetti in foto o immagini [1] [2].

Allo stesso modo, le applicazioni di apprendimento automatico per i file audio audio, dipendevano dalle tradizionali tecniche di elaborazione *"pure"* del segnale digitale per estrarre le relative features di interesse; ad esempio, per comprendere il linguaggio umano, i segnali audio venivano soprattutto analizzati utilizzando concetti fonetici in senso stretto, attraverso l'estrazione di elementi specifici e specializzati (come appunto i fonemi).
Tutto ciò richiedeva molte competenze avanzate del dominio in cui si operava per risolvere questo tipo di problemi.
Tuttavia negli ultimi anni, in seguito al maggiore progresso tecnologico e all'avvento di un utilizzo sempre più comune delle **GPU**, il *Deep Learning* ha trovato molta più fortuna, riscontrando un enorme successo anche nella gestione dei file audio; nella fattispecie, vi è stato un graduale abbandono delle tradizionali tecniche di elaborazione audio che, col passare del tempo, sono risultate sempre meno necessarie.
Le ottime performance ottenuto nell'ambito della Computer Vision hanno portato gli esperti ad utilizzare le architetture CNN anche per i segnali audio (segnali monodimensionali). Si è quindi studiato un metodo per convertire i segnali 1D in segnali 2D.

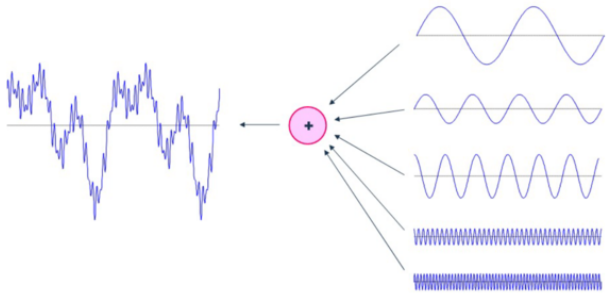
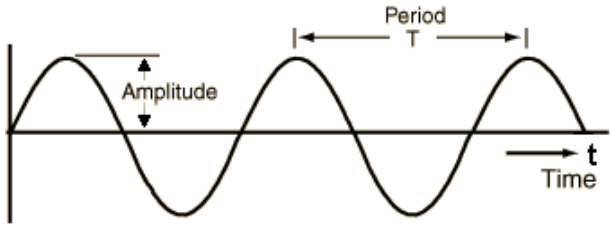


Fig. 1. La figura mostra come una funzione segnale viene decomposta in funzioni segnali elementari

A. I segnali audio: un po' di teoria.

Una caratteristica importante dei segnali audio è quella di ripetersi spesso a intervalli regolari in modo tale che ogni onda abbia la stessa **forma** (mantenendo così un qualche tipo di *periodicità nel segnale*). L'altezza dell'onda mostra l'intensità del suono ed è nota come **ampiezza**. Il tempo impiegato dal segnale per completare un'onda intera è il **periodo**. Il numero di onde prodotte dal segnale in un secondo è chiamato **frequenza**. La frequenza inoltre è il reciproco del periodo. L'unità di misura della frequenza è l'**Hertz (Hz)**.

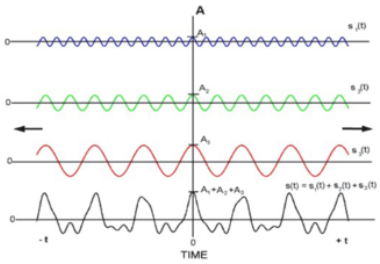


Il **campionamento audio** è il procedimento di conversione in forma *digitale* di un segnale **audio analogico**, per la creazione dei campioni audio da utilizzare in fase di analisi. Attraverso il campionamento audio si può trasformare il segnale in un insieme di veri e propri numeri (dati da esaminare) in modo tale che possa essere letto e analizzato dai modelli di Machine Learning.

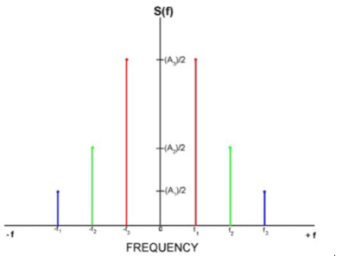
B. Lo spettro e gli spettrogrammi.

I segnali di frequenze diverse possono essere sommati per creare segnali compositi, che rappresentano qualsiasi suono che si verifica nel mondo reale; ciò significa che qualsiasi segnale è costituito da molte frequenze distinte e può essere espresso come la somma delle stesse, come si osserva dalla Figura 3.
Nella fattispecie, ogni funzione periodica può essere espressa come somma di funzioni seno e coseno (*combinazione di funzioni armoniche*).

Lo **spettro sonoro** è l'insieme di **frequenze** che vengono combinate insieme per produrre un segnale; questo tiene traccia di tutte le frequenze presenti nel segnale insieme all'intensità (o



(a) Rappresentazione nel dominio del tempo



(b) Rappresentazione nel dominio delle frequenze

Fig. 2. Le due immagini mostrano i due diversi tipi di rappresentazione del segnale audio

ampiezza) di ciascuna frequenza. La frequenza **più bassa** in un segnale è chiamata *frequenza fondamentale*.
Le frequenze che sono multipli di numeri interi della frequenza fondamentale sono note come *armoniche*.
Esistono due tipi diversi di rappresentazioni (*domini*) per i segnali audio.
Le forme d'onda di cui si è parlato in precedenza, rappresentano i segnali audio nei quali viene esplicitata *l'ampiezza* (o l'intensità) *rispetto al tempo* (in particolare, l'asse del dominio mostra l'intervallo di valori temporali del segnale).

Lo **spettro** offre invece una modalità alternativa di rappresentare lo stesso segnale in un dominio differente; questo infatti mostra *l'ampiezza* (o intensità) *rispetto alla frequenza* (in particolare, l'asse del dominio mostra l'intervallo dei valori di frequenza del segnale).

Lo ***spettrogramma*** di un segnale tiene traccia del suo spettro nel tempo, il quale è come una "fotografia" del segnale (traccia il *tempo sul suo dominio* e la *frequenza sul suo codomionio*). Per la sua costruzione viene quindi ricavato lo spettro in momenti diversi; successivamente vengono poi uniti tutti gli istanti insieme in un'unica trama. Gli *spettrogrammi* vengono prodotti utilizzando le ***trasformate di Fourier*** per scomporre qualsiasi segnale nelle sue frequenze costituenti. Poichè vi è *corrispondenza biunivoca* tra trasformata di Fourier e la sua antitrasformata, la stessa corrispondenza è presente tra il *dominio temporale* e il *dominio delle frequenze*.

C. La Scala Mel

Gli esseri umani non percepiscono le frequenze su scala lineare; per l'orecchio umano è molto più semplice rilevare le differenze nelle frequenze più basse rispetto alle frequenze più alte (ad esempio, gli esseri umani possono facilmente descrivere la differenza tra 500 e 1000 Hz, ma difficilmente sono in grado di distinguere una differenza tra 10.000 e 10.500 Hz, anche se la distanza tra le due coppie è la stessa e identica).
Nel 1937, Stevens, Volkmann e Newmann proposero un'unità di tono tale che distanze uguali nel tono suonassero ugualmente distanti per l'ascoltatore. Questa è chiamata **"Scala Mel"** e non è altro che il risultato di alcune trasformazioni non-lineari sulla scala delle frequenze. Allo stesso modo, uno ***"spettrogramma Mel"*** è appunto uno spettrogramma in cui le frequenze vengono convertite nella scala Mel.

2. MATERIALI E METODI

A. Il dataset e gli obiettivi dell'analisi

Il dataset ***Ravdess Emotional Speech Audio*** contiene ***1440 file audio*** con estensione ***.wav*** ognuno con determinate caratteristiche che saranno studiate in maniera indipendente, l'una rispetto alle altre, in questo elaborato.
RAVDESS è un dataset *multimodale*, bilanciato rispetto a diverse caratteristiche; tra queste saranno oggetto di studio:

- *Sesso* - Nel dataset sono presenti dodici attori di sesso maschile e 12 attori di sesso femminile, per un totale di 24 attori;
- *ID Attore* - Il codice identificativo di ognuno dei 24 attori che pronuncia la frase in esame;
- *Frase Pronunciata* - Ognuno degli attori pronuncia **due** frasi diverse vocalizzate con accento nordamericano di tipo neutro.
- *Emozione* - Ogni frase viene pronunciata con **otto** tipologie di emozioni diverse.

Inoltre per ogni attore sono presenti 60 diversi file vocali $(60 \cdot 24 = 1440)$

Ciascuno dei 1440 file ha un nome *univoco*. Nello specifico, il nome del file è costituito da un identificatore numerico di 7 parti (ad es. 03-01-06-01-02-01-12.wav) dove ogni parte dell'identificatore si riferisce a una specifica caratteristica del file stesso
Ognuna delle parti dell'identificatore è inserita con un ordine ben preciso e si riferisce a:

- *Modalità* - (01 = full-AV, 02 = solo video, 03 = solo audio);
- *Canale vocale* - (01 = parlato, 02 = cantato);
- *Emozione* - (01 = neutrale, 02 = calmo, 03 = felice, 04 = triste, 05 = arrabbiato, 06 = impaurito, 07 = disgustato, 08 = sorpreso);
- *Intensità Emotiva* - (01 = normale, 02 = forte)[1];
- *Frase Pronunciata* - (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door");
- *Ripetizione della frase* - (01 = prima ripetizione, 02 = seconda ripetizione);
- *Attore* - (L'identificativo va da 01 a 24. Gli attori dispari sono maschi, gli attori pari sono femmine).

[1]Non c'è una forte intensità per l'emozione "neutrale"

B. EDA

La prima fase dello studio è stata quella fare un'indagine più approfondita del dataset al fine di scovare eventuali pattern d'interesse.
Ci si è quindi chiesti quanti fossero gli audio per ogni emozione condizionatamente al sesso degli attori. È emerso che per ogni emozione condizionata al sesso dell'attore sono presenti 96 file audio; l'unica eccezione è rappresentata però dalla categoria *"**neutrale**"* poichè sono presenti 48 file audio sia per quanto riguarda gli attori di sesso maschile che di sesso femminile. Questo fatto potrebbe anticipare una prima problematica nel task in esame: ovvero, essendoci meno emozioni provenienti dalla già citata categoria e quindi meno informazioni su cui apprendere, i modelli potrebbero fare più fatica a discriminare correttamente le emozioni ***neutre***.

Successivamente, poichè non sono stati riscontrati altri dettagli interessanti da evidenziare nella struttura del dataset, si ci è concentrati sull'analisi di quelle che sono le caratteristiche vere e proprie dei segnali audio. Ognuno dei 1440 file audio ha un'estensione *.wav* con una profondità di *16 bit* ciascuno; inoltre i file audio presenti nel dataset possono avere in input sia un solo segnale (file audio in *mono*) che due segnali (file audio in *stereo*). Infine gli audio, essendo stati registrati da persone diverse e in momenti diversi, hanno durata molto simile tra loro ma mai identica.

C. Preprocessing degli audio

Si è iniziato a lavorare sui dati passando alla fase di ***preprocessing dei segnali audio***.
Per prima cosa è stato diviso il dataset originale in modo che potesse essere utilizzabile per la costruzione e la valutazione dei modelli. La suddivisione è avvenuta nel seguente modo:

- **Training Set** - Contiene 1008 file audio (ovvero il 70% del dataset originale);
- **Validation Set** - Contiene 432 file audio (ovvero il 15% del dataset originale)
- **Test Set** - Contiene 432 file audio (ovvero il restante 15% del dataset originale)

Successivamente si è lavorato sulla durata degli audio; le tracce sono state tagliate affinchè presentassero tutte quante la medesima lunghezza. Questa procedura è stata molto utile poichè, all'interno di ogni traccia, sono presenti dei momenti di assenza di segnale prima e dopo che l'attore pronunci la frase. Tagliare i segnali audio, oltre a ridimenionare la complessità del problema, evita eventuali *Bias* prodotti eventualmente all'interno delle *reti neurali* dovuti a parti in cui non è presente alcuna informazione.

Ognuno dei file audio ridimensionati è stato quindi convertito su ***scala Mel***. Dagli audio convertiti sono stati generati gli ***spettrogrammi Mel***, che saranno fondamentali come input di addestramento per le *reti neurali* che si andranno a costruire.
È stato fatto questo tipo di scelta poichè in letteratura [3] [4] vi è evidenza empirica su quelli che possono essere i miglioramenti evidenti nelle prestazioni dei modelli di machine learning in ambito di *audio classification*; vista quindi la natura stessa del task in esame è sembrata essere un'idea coerente per la risoluzione del problema (soprattutto per quanto riguarda la classificazione delle variabili multinomiali, ovvero il riconoscimento delle *Emozioni* e degli *Attori*).

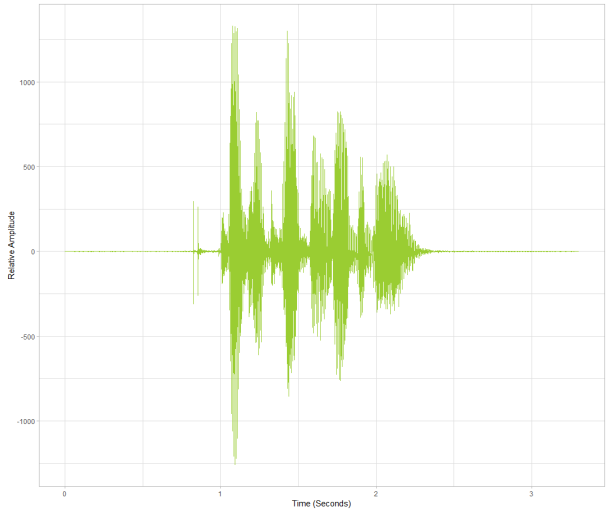


Fig. 3. Esempio di un generico segnale audio presente nel dataset; si può notare l'assenza completa di segnale ai due estremi della figura

Le immagini degli spettrogrammi sono state salvate in scala di grigi (il canale colore ha, dunque, dimensione 1). Questa scelta è dovuta al fatto che uno spettrogramma così creato contiene tutte le informazioni rilevanti nell'intensità dei suoi pixel. Per motivi di visualizzazione nella Figura 4 vengono mostrati due differenti spettrogrammi (relativi a una donna e un uomo) utilizzando una scala colore viridis.

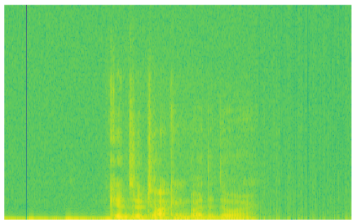
3. MODELLI UTILIZZATI E RISULTATI

Nella seguente sezione verranno discusse le metodologie di Deep Learning utilizzate per la risoluzione dei differenti task.

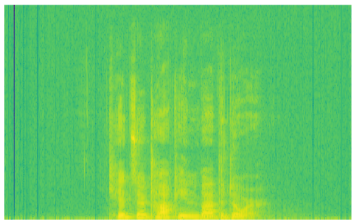
A. Convolutional Neural Network (CNNs)

Una delle scelte principali del lavoro è stata sicuramente quella di allenare ***reti neurali convoluzionnali (CNNs)*** a partire dagli spettrogrammi Mel precendentemente creati a partire dalle tracce audio del dataset.

Le reti convoluzionali (CNN) sono dei particolari tipi di architetture di reti neurali, al giorno d'oggi lo strumento migliore per trattare problemi di *Computer Vision* come *image classification, object localization, object detection e object segmentation*. La ragione principale per cui le CNN hanno riscontrato un così grosso successo nel campo è dovuto al fatto che sono in grado di gestire dati non lineari. Nelle CNNs l'input è un tensore di tre dimensioni; nelle immagini ognuna di queste si riferirisce alla dimensione dell'*altezza*, della *larghezza* e della *profondità* (dove per profondità si intende il numero di canali colore nelle immagini stesse).
Ciò che differenzia le reti convoluzionali dalle tradizionali *artificial neural network* sono i layer che le due architetture utilizzano. La differenza tra un layer *denso* e un layer *convoluzionale* è la seguente: i layer densi apprendono dei pattern globali nello spazio delle features di input, mentre i layer convoluzionali apprendono dei pattern locali: nel caso di immagini, i pattern rappresentano delle porzioni 2D di immagini. La chiave caratteristica delle reti convoluzionali risiede in due importanti proprietà:



(a) Spettrogramma Mel - Maschio Neutrale.



(b) Spettrogramma Mel- Femmina Neutrale.

Fig. 4. Le due immagini mostrano due esempi di ***spettrogrammi Mel***; gli attori sono un uomo e una donna che pronunciano la stessa frase con lo stessa emotività.

- I pattern che individuono sono *invarianti rispetto a traslazioni*: dopo aver appreso un pattern particolare in una zona precisa dell'immagine, un rete convoluzionale può individuarla ovunque.
- Una rete convoluzionale può imparare pattern spaziali in maniera gerarchica. I primi layer convoluzionali possono apprendere una rappresentazione locale (rappresentazioni generiche) come i contorni, mentre i layer successivi sono in grado di specializzarsi su rappresentazioni più complesse (per esempio, occhi, bocca, naso nel caso di una persona).

B. Applicazione delle CNNs al dataset

Le reti neurali convoluzionali sono state addestrate partendo dagli spettrogrammi dei dati precedentemente preprocessati. Gli spettrogrammi mel in input sono stati ulteriormente riconvertiti in tensori riscalando i pixel in un range di valori [0,1].
Dovendo gestire una mole di immagini piuttosto elevata (1440 immagini in formato .png) e non riuscendo a salvare le informazioni di queste in un *array* di R, si è deciso di ricorrere all'utilizzo di un `Data Generator`, uno strumento built-in di keras che permette di lavorare con dataset di dimensione elevata (contenti migliaia o milioni di immagini) senza il rischio di incorrere in problemi di memoria (OOM, Out-Of- Memory). Nelle CNNs sono quindi stati dati in input dei tensori di dimensione: altezza = 256, larghezza = 256 e profondità = 1. Come funzioni di perdita nella fase di addestramento dei pesi delle reti sono state considerate:

- Per i task di classificazione binaria (Sesso degli attori e Frase Pronunciata) la ***"Binary Cross-Entropy"***
- Per i task di classificazione multiclasse (identificazione dell'attore e dell'emozione) la ***"Categorical Cross-Entropy"***

Come ***optimizer*** delle reti implementate si è deciso di utilizzare ***"Adam"*** [5]. Il motivo di questa scelta è dovuta al fatto che la maggior parte delle applicazioni nell'ambito deep learning sfrutta questo metodo di ottimizzazione adattivo che coniuga le caratteristiche di **RMSprop** e **Momentum**.

Model: Sequential		
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 254, 254, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 62, 62, 64)	0
flatten (Flatten)	(None, 246016)	0
dense_1 (Dense)	(None, 128)	31490176
dropout (Dropout)	(None, 128)	0
dense (Dense)	(None, 1)	129
Total params: 31,509,121		
Trainable params: 31,509,121		
Non-trainable params: 0		

Table 1. Architettura CNNs utilizzata per l'allenamento dei task in esame

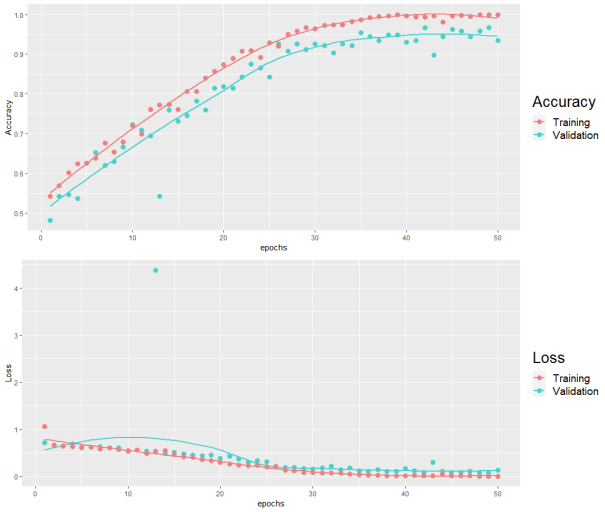
Per ogni rete è stata scelta come metrica di valutazione dei risultati l'***"accuratezza"***. La scelta è giustificata dal fatto che il numero di classi nella variabile risposta è quasi sempre perfettamente bilanciato.

Le reti sono state costruite cercando di ottenere un buon compromesso tra la precisione dei risultati sul validation set e la gestione del sovradattamento del modello sui dati di training (*overfitting*).
La scelta delle diverse architetture, nei diversi task, è stata effettuata provando diverse configurazioni manualmente e valutando le performance di queste sul set di validazione.
Si è giunti a delle architetture molto simili tra di loro anche in contesti di task differenti. La configurazione ottenuta è quella riportata nella Tabella 1.

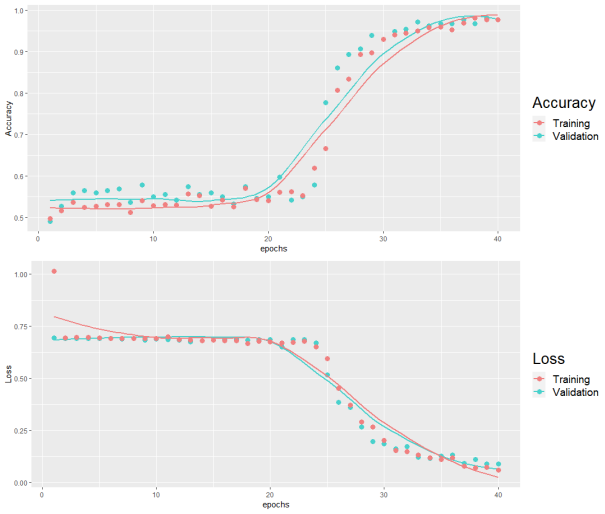
La scelta di utilizzare come funzione di attivazione, per i layer intermedi, una funzione `ReLU` è dettata dal fatto che questa risulta semplice, veloce e, empiricamente, si dimostra che performa bene. Inoltre, è utile per prevenire uno dei problemi ricorrenti delle reti neurali: la *scomparsa di gradiente*.
L'utilizzo di un layer di `Dropout` prima dell'ultimo strato Fully Connected è stato di particolare utilità per mitigare l'overfitting durante la fase di training.
Nelle fasi di addestramento, validazione e valutazione finale dei modelli, è stato sfruttato Google Colab, una piattaforma per eseguire codice sul Cloud, per avere a disposizione l'ausilio di una ***GPU*** al fine di mitigare eventuali problemi computazionali [6]; infatti allenare una CNN può risultare essere un processo particolarmente lento, la cui causa è da ricercarsi nel vasto numero di computazioni richiesto per ogni iterazione.

C. Risultati CNNs

I risultati delle reti neurali convoluzionali sono stati particolarmente soddisfacenti per tutti e quattro i task.

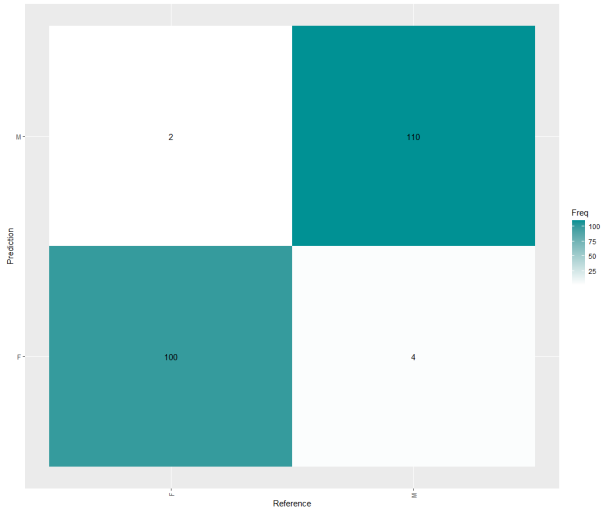


(a) Performances CNN Sesso dell'attore.

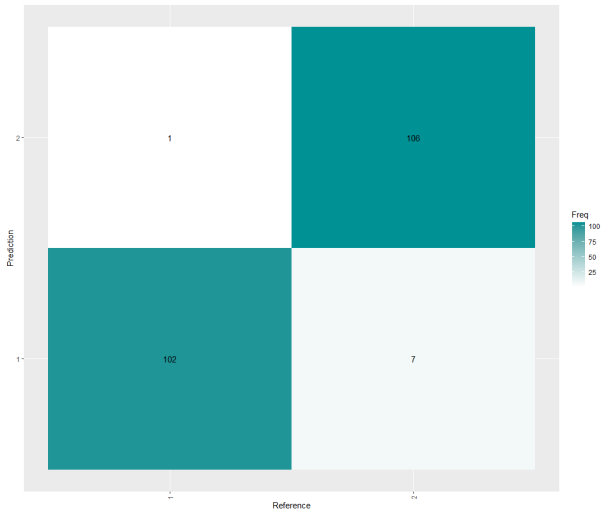


(b) Performances CNN Frase Pronunciata

Fig. 5. Le due immagini mostrano l'andamento dell'*accuratezza* e della *funzione di perdita* per quanto riguarda le CNNs nei task di classificazione binaria



(a) Confusion Matrix - Risultati sul test-set del modello CNN sul Sesso dell'attore.



(b) Confusion Matrix - Risultati sul test-set del modello CNN sulla Frase Pronunciata.

Fig. 6. Dalle due matrici di confusione non sembrano emergere particolari problemi nel classificare una determinata categoria in particolare.

Per quanto riguarda i task di **classificazione binaria** (Sesso e Frase Pronunciata) si è arrivati a performances di accuratezza particolarmente elevate (l'accuratezza converge, quasi, al 100%) sia sul training set che sul validation set, come si osserva dalla Figura 5.
Il problema riguardante la classificazione del *Sesso* dell'attore è stato risolto allenando la rete neurale su 50 epoche; per quanto riguarda invece la classificazione delle *Frasi Pronunciate* ne sono bastate 40.

Per quanto riguarda le performances sul ***test set***, in entrambi i task è stato confermato il trend del validation; nella fattispecie:

- Per la ***Frase pronunciata*** l'accuratezza sul test-set è del 94.91%;
- Per il ***Sesso*** dell'attore l'accuratezza sul test-set è del 97.22%.

Sono state create infine ***le matrici di confusione*** (Figura 6) per verificare i risultati sul test set e da queste non sono emerse particolari anomalie.

Per quanto concerne invece i task di **classificazione multiclasse** (Id Attore e Emozione) risulta invece di fondamentale importanza fare un discorso specifico per entrambi.
Gli attori che recitano una frase, presenti all'interno del dataset, sono 24; ci si trova, dunque, per la prima volta, di fronte ad un problema di classicazione non banale (la risposta non è binaria). Il problema di classificazione è bilanciato (tutti e 24 gli attori recitano lo stesso numero di volte). La baseline del modello, in termini di accuracy, è di 0.04166667 ($\frac{1}{24} \approx 4\%$) Questo è il risultato che ci si aspetta da un *classificatore triviale* (un classificatore che identifica tutte le osservazioni come appartenenti ad un'unica etichetta).
Per questo tipo di task, seppur l'architettura della rete e quasi identica a quelle addestrate in precedenza, è stato utilizziamo un modello leggermente più complesso; la maggiore complessità

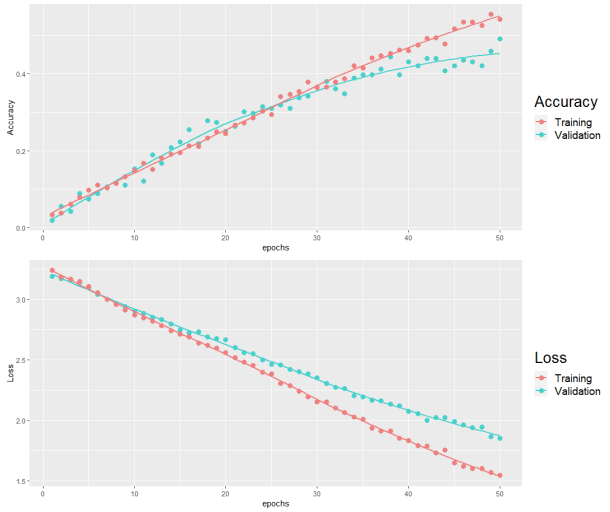


Fig. 7. L'immagine mostra l'andamento dell'*accuratezza* e della *funzione di perdita* per quanto riguarda la CNN dell'ID dell'Attore

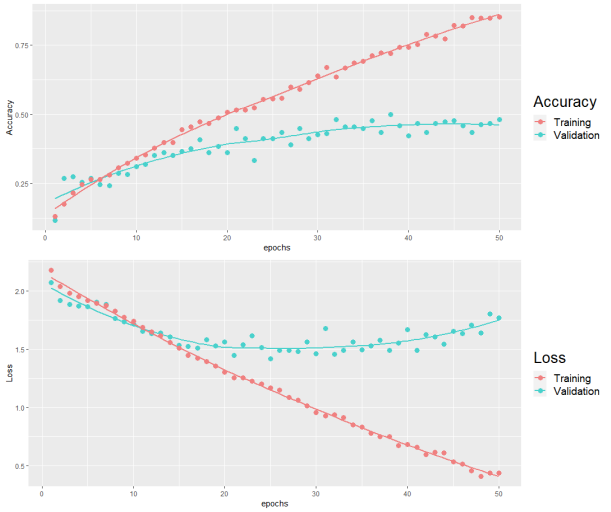


Fig. 9. L'immagine mostra l'andamento dell'*accuratezza* e della *funzione di perdita* per quanto riguarda la CNN dell'Emozione dell'Attore; risulta evidente un forte overfitting

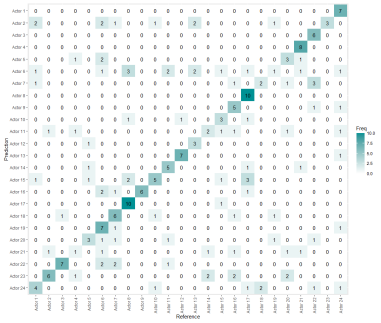


Fig. 8. Confusion Matrix - Risultati sul test-set del modello CNN sull'ID Attore.

risiede nel numero diverso di neuroni nel primo layer `FULLY CONNECTED` (**512**). Inoltre, trattandosi di un problema di classificazione multimodale, il layer FULLY CONNECTED finale avrà 24 neuroni (tanti quanti sono gli attori) e una funzione di attivazione `softmax`.
La rete neurale è stata addestrata su 50 epoche e, a partire dalla 40esima, si osserva un leggero overfitting (tenuto comunque sotto controllo dalla presenza del layer di `Dropout`).

Sul **test-set** il modello performa piuttosto bene; l'accuratezza è del 55.56% (ovvero più di 10 volte migliore rispetto al classificatore triviale). Considerando che alcuni attori, del medesimo sesso, hanno una voce molto simile, soprattutto quando recitano utilizzando lo stesso tono emotivo, si ritiene di essere arrivati, anche in questo caso, ad un risultato più che soddisfacente.
In task come questo, infatti, sarebbe difficile anche per l'orecchio umano discriminare le voci degli attori con una bassa percentuale di errori.
Per completezza si riporta anche in questo caso la matrice di confusione in Figura 8.

Il discorso relativo alle *emozioni* invece risulta essere completamente diverso rispetto a tutti gli altri, sia teoricamente che per quanto riguarda le performance della CNN addestrata.
Come detto in precedenza, riconoscere le emozioni risulta essere notoriamente una sfida particolarmente complicata in letteratura poichè l'unica discriminante risulta quasi sempre essere solo ed unicamente il *tono vocale*; le strutture di frequenza tra gli audio di uno stesso attore che dice sempre la stessa frase ma con toni emotivi differenti sono quasi sempre identiche tra loro.
Avendo a che fare con 8 tipi di emozioni diverse, ci si aspetta da un classificatore triviale un'accuratezza, in termini di previsione, pari a $\frac{1}{8} \approx 12.5\%$.
Si ricorda inoltre che il task riguardante le emozioni risulta essere l'unico non completamente bilanciato (in quanto le emozioni neutrali sono presenti la metà delle volte rispetto a tutte quante le altre, sia per quanto riguarda gli attori di sesso maschile che quelli di sesso femminile). In fase di addestramento della rete, anche in questo caso, l'unico cambiamento apportato rispetto alle reti precedentemente considerate è stato modificare il numero di neuroni nel primo layer `FULLY CONNECTED` (**256**) e nel layer di FULLY CONNECTED finale avrà 8 neuroni (tanti quante sono, in questo caso, le emozioni) con una funzione di attivazione *softmax*.
La rete è stata addestrata su 50 epoche e, come ci si aspettava, le performance sul set di validazione, in termini di accuratezza, non sono così alte come nei modelli implementati per i task precedenti. E' inoltre presente un evidente overfitting come si osserva dalla Figura 9.

Sul **Test-set** il modello è in grado di discriminare correttamente il 42.59% delle osservazioni (circa 3,5 volte maggiori rispetto a quelle del classificatore triviale).
L'utilizzo degli ***Spettrogrammi Mel*** hanno permesso alla rete di è riuscire ad arrivare comunque a performances accettabili, seppur non altissime, in un task notoriamente molto complesso.
Anche in questo caso si riporta la matrice di confusione in Figura 10.
Come previsto, la confusion matrix evidenzia le basse performances per il riconoscimento delle emozioni di tipo neutro (vista sostanziale mancanza di audio riguardanti questa categoria).

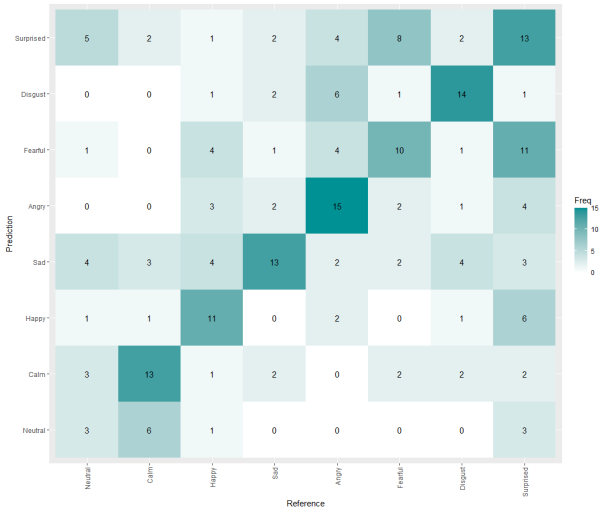


Fig. 10. Confusion Matrix - Risultati sul test-set del modello CNN sulle Emozioni.

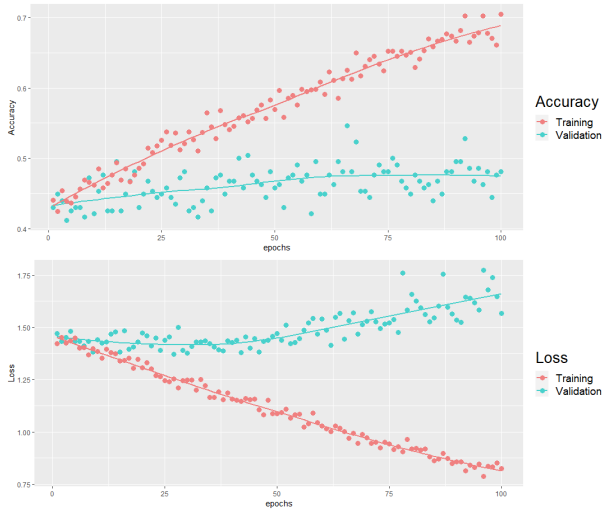


Fig. 11. L'immagine mostra l'andamento dell'*accuratezza* e della *funzione di perdita* per quanto riguarda la CNN dell'Emozione dell'Attore **(Data Augmentation)**; l'overfitting sembra essersi mitigato

D. Approcci Alternativi per il task sulle emozioni

Viste le prestazioni migliorabili per quanto riguarda il task delle emozioni, non ci si è fermati solo e unicamenta alla costruzione di CNNs da zero ma sono stati esplorati una serie di approcci alternativi, con la finalità di migliorare l'overfitting sul trainig set e/o provare a migliorare l'accuratezza della classificazione.

D.1. Data Augmentation

Il problema dell'overfitting è causato dalla presenza di un numero insufficiente di campioni da cui imparare, rendendo impossibile il training di un modello che possa generalizzare a nuovi dati.
La tecnica di ***Data Augmentation*** è una potente metodologia utilizzata per mitigare l'overfitting nell'ambito della Computer Vision.
Data Augmentation adotta l'approccio di generare nuovi dati di training partendo dal campione di addestramento già esistente; vengono infatti aumentati i campioni di training tramite una serie di trasformazioni casuali che producono immagini dall'aspetto simile. Così facendo il modello non apprenderà mai dalla stessa e identica immagine bensì da immagini simili. Questo dovrebbe consentire al modello di generalizzare meglio riducendo in qualche misura l'overfitting.
Non avendo a che fare con immagini convenzionali, ma con spettrogrammi, il processo di data augmentation del training set è stato svolto secondo il criterio di non stravolgere troppo i dati applicando troppe trasformazioni (applicare troppe trasformazioni agli spettrogrammi avrebbe permesso alla rete di controllare l'overfitting ma avrebbe abbassato moltissimo le performances del modello).
A questo proposito sono state fatte solo le seguenti trasformazioni sugli spettrogrammi:

- *zoom range* - parametro che serve per ingrandire in modo casuale le immagini nel range fissato. É stato impostato pari a **0.05**;
- *width shift* - parametro che serve a traslare l'immagine in orizzontale (il valore si riferisce una frazione di spostamento rispetto al totale). É stato impostato pari a **0.02**;
- *height shift* - parametro che serve a traslare l'immagine in verticale (il valore si riferisce una frazione di spostamento rispetto al totale). É stato impostato pari a **0.02**.

È stata così nuovamente addestrata la rete utilizzata in precedenza. Nella nuova fase di addestramento è stato deciso di aumentare il numero di epoche rispetto alla rete precedente poichè è stato inserito del Bias nei dati (dovuto ad immagini artefatte).
Dalle prime 50 epoche, nella Figura 11 si osserva come, attraverso l'uso di data augmentation, si riesce a ridurre il forte overfitting della rete (seppur non si riesce ancora ad eliminarlo completamente).
A livello di prestazioni sul **test set** il modello presenta performance leggermente più alte rispetto alla rete nel quale non si è fatto Data Augmentation, avendo un'accuratezza pari al 43.98%.
La scelta di effettuare data augmentation si è rivelata quindi vincente in quanto si è riusciti a mitigare leggermente l'overfitting e a migliorare, seppur leggermente, le prestazioni del modello sul test set.
Anche in questo caso si riporta la matrice di confusione in Figura 12.

D.2. Utilizzo di reti pre-allenate: Fine-tuning sulle Emozioni (VGG-16)

Un approccio comune e altamente efficace nel mondo del deep learning, quando si ha a che fare con set di immagini molto piccoli, consiste nell'utilizzare una rete pre-addestrata.
Una rete pre-addestrata è una rete neurale che è stata precedentemente addestrata su un set di dati di grandi dimensioni (in genere per un'attività di classificazione delle immagini su larga scala). Se il set di dati originale sul quale è stata pre-addestrata la rete è sufficientemente **ampio** e sufficientemente **vario**, la gerarchia spaziale delle caratteristiche apprese dalla rete pre-addestrata può effettivamente fungere da modello generico per un qualsiasi task visivo; per questo motivo le sue caratteristiche possono rivelarsi utili per molti diversi problemi di computer vision (anche se questi possono coinvolgere classi completa-

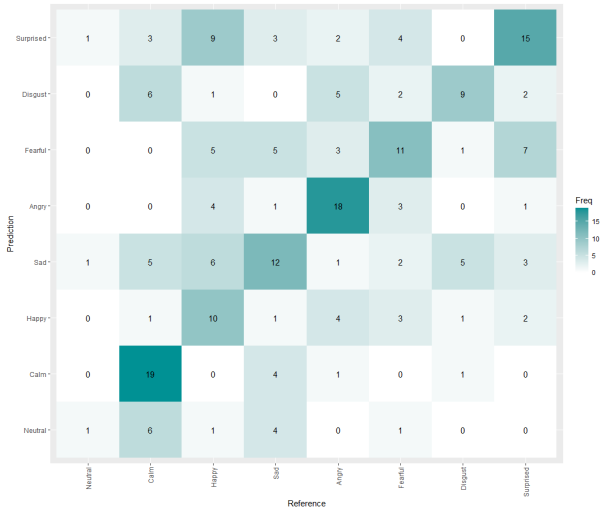


Fig. 12. Confusion Matrix - Risultati sul test-set del modello CNN sulle Emozioni **(Data Augmentation)**.

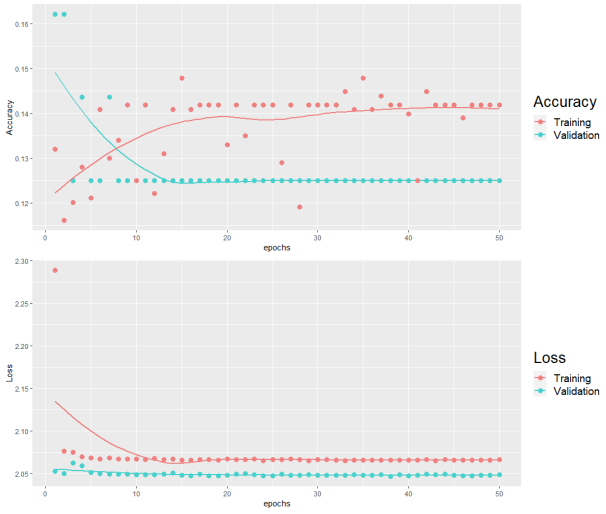


Fig. 13. L'immagine mostra l'andamento dell'*accuratezza* e della *funzione di perdita* per quanto riguarda l'identificazione dell'Emozione dell'attore, utilizzando fine-tuning su VGG-16; le prestazioni risultano deludenti.

mente diverse da quelle presenti nel dataset sul quale è stata pre-addestrata la rete).
Il ***Fine-Tuning*** consiste nel tenere congelati i primi layer convoluzionali della rete pre-addestrata (congelando di conseguenza anche i pesi ad essi associati) scongelando soltanto gli ultimi strati di questa; una volta fatta questa operazione si possono definire altri layer aggiuntivi e si inseriscono i Fully-Connected. Infine vengono date in input alla rete le immagini del dataset su cui si vuole lavorare (non il dataset su cui è stata già pre-allenata al rete).
Solitamente vengono congelati i primi layer poichè i primi livelli codificano funzionalità più generiche e riutilizzabili, mentre i livelli più in alto codificano funzionalità più specializzate. Risulta infatti di maggiore utilità riaddestrare le funzionalità più specializzate, perché queste sono quelle che devono essere riproposte per il nuovo problema.
Con l'obiettivo di ottenere dei risultati migliori in termini di accuracy, l'idea è stata quella di esplorare anche metodi alternativi rispetto alla costruzione di una rete da zero. Si è per questo motivo tentato un approccio di **fine-tuning** utilizzando una rete altamente performante come **VGG-16** (la quale è stata precendetemente allenata sul dataset *Imagenet*).
Si è deciso di congelare la rete ai primi blocchi convoluzionali scongelando i pesi dal layer **"block3 conv1"** (questo perchè il tipo di task in esame è molto diverso dalle immagini su cui la rete è stata allenata). L'idea è, quindi, che il modello vada ad estrarre delle features generiche a partire dalle nostre immagini nei primi layer della rete.
I risultati, come lecito aspettarsi vista la natura stessa del task, sono piuttosto deludenti. Le immagini degli spettrogrammi sono molto diverse rispetto a quelle su cui è stata allenata Imagenet (immagini di animali, oggetti, frutta, ...). La rete non riesce ad apprendere correttamente le diverse emozioni (le performance sono paragonabili a quelle di un classificatore triviale).

D.3. AutoEncoder e FFNN

Per risolvere il problema dell'overfitting, in riferimento al task sulle emozioni, si è deciso di ricorrere ad un approccio alternativo. Si è deciso di ridurre la dimensione dei dati originali (immagini degli spettrogrammi) sfruttando un particolare tipo di architettura di rete neurale: gli **AutoEncoder**.

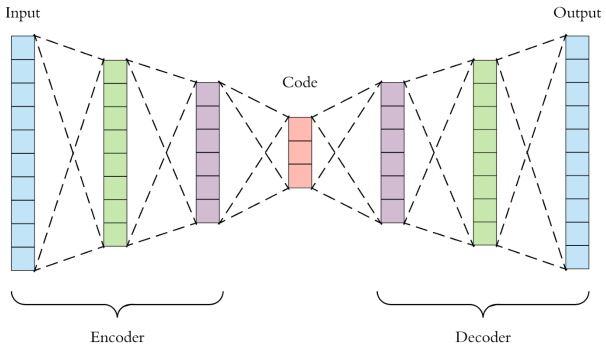


Fig. 14. L'immagine mostra la forma tipica di un autoencoder (non necessariamente, tuttavia, la rete deve essere simmetrica)

Gli Autoencoders sono reti neurali artificiali capaci di rappresentare in modo efficiente dati in input imparando i cosiddetti **codings**, sfruttando un approccio non supervisionato.
I codings costituiscono dunque un'efficiente rappresentazione dei dati, ridotti a una dimensionalità inferiore. Questo fa degli autoencoders dei potenti strumenti di *dimensionality reduction*.

Un autoencoder è sempre costituito da due parti, come osservato dalla Figura 14:

- Una rete di riconoscimento, *encoder* che converte gli input nelle rappresentazioni interne.
- Una rete generativa chiamata *decoder* che converte la rappresentazione interna in un output.

L'originalità del metodo sta nel fatto che, tra l'encoder e il decoder, vi è uno spazio, che nella presente implementazione sarà di 128 neuroni, chiamato *'latent space'*, in cui verranno sintetizzate le informazioni dei tensori originali.
L'idea è quella di allenare l'AutoEncoder in modo tale da minimizzare una qualche funzione di perdita (*reconstruction loss*), (tipicamente si tratta o di *binary crossentropy* o di *Mean Squared Error*, nella presente implementazione abbiamo utilizzato

Fig. 15. L'immagine mostra l'andamento di Accuracy e Loss sul set di training e validazione dell'Artificial Neural Network a partire dalle features estratte dallo spazio latente dell'AutoEncoder

il secondo perchè ci permette di ottenere una ricostruzione migliore).
Successivamente, sfruttando la functional API di keras, si è fatta previsione sul training, validation e test set nello spazio latente per ottenere le 'nuove features' che contengono una parte dell'informazione originaria (notevolmente ridotta dimensionalmente).
L'AutoEncoder implementato ha la struttura evidenziata nella Tabella 2.

Una volta estratte le features di interesse, queste diventano l'input di una rete Feed Forward semplice dalle caratteristiche riportate nella Tabella 3.

Dalla Figura 15 si osserva come le due curve (quella relativa al set di training e al set di validazione) sono molto vicine. L'overfitting, utilizzando questa procedura, è stato fortemente mitigato, tuttavia la potenza predittiva del modello è calata drasticamente (l'accuratezza sul set di validazione è $\sim 20\%$).

4. CONCLUSIONI E POSSIBILI SVILUPPI

I risultati ottenuti sono, in generale, piuttosto soddisfacenti. Attraverso l'utilizzo degli spettrogrammi di Mel e di Reti Neurali Convoluzionali implementate da zero, si è riuscito ad ottenere dei modelli che discriminano correttamente la maggior parte delle osservazioni in riferimento ai diversi task.
In particolare, i modelli implementati per i due problemi di classificazione binaria, raggiungono delle performance estremamente elevate in termini di *accuracy*. Per quanto riguarda, invece, i due problemi di classificazione multiclasse, è stato necessario trovare una soluzione per mitigare le forme di sovradattamento che i modelli mostravano (sia per quanto riguarda l'identificazione dell'attore, in forma lieve, sia, soprattutto, per quanto riguarda l'emozione da discriminare).

L'utilizzo del linguaggio `R` ha posto grossi limiti alla realizzazione del progetto:

- Nella fase di preprocessing e processing dei dati si è dovuti

Model: Model

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 1)]	0
conv2d_1 (Conv2D)	(None, 254, 254, 32)	160
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d (Conv2D)	(None, 64, 64, 16)	2064
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 128)	524416
dense_1 (Dense)	(None, 4096)	528384
reshape (Reshape)	(None, 16, 16, 16)	0
conv2d_4 (Conv2D)	(None, 16, 16, 16)	1040
up_sampling2d_1 (UpSampling2D)	(None, 64, 64, 16)	0
conv2d_3 (Conv2D)	(None, 64, 64, 32)	2080
up_sampling2d (UpSampling2D)	(None, 256, 256, 32)	0
conv2d_2 (Conv2D)	(None, 256, 256, 1)	129

Total params: 1,058,273
Trainable params: 1,058,273
Non-trainable params: 0

Table 2. Configurazione dell'architettura di Autoencoder

Model: Sequential_4

Layer (type)	Output Shape	Param #
dense_12 (Dense)	(None, 256)	33024
dense_11 (Dense)	(None, 128)	32896
dense_10 (Dense)	(None, 64)	8256
dense_9 (Dense)	(None, 8)	520

Total params: 74,696
Trainable params: 74,696
Non-trainable params: 0

Table 3. Configurazione della Feed Forward Neural Network

ricorrere a delle strategie efficienti per la manipolazione dei file audio e lo stoccaggio delle immagini all'interno dei Data Generator di keras.
Questo, tuttavia, non ha permesso di lavorare con il segnale audio grezzo, poichè immagazzinare, all'interno di un *array*, 1440 segnali 20'000 dimensionali, portava, inevitabilmente, ad un esaurimento della memoria RAM disponibile.

- Non esistono, in `R`, librerie mature per il processing dei segnali audio (a differenza di Python che presenta diverse alternative tra cui `Librosa`, `Pyo`, `PyAudio`, ...)

Uno dei possibili sviluppi futuri da citare è quello di sfruttare un approccio **Multi-Task Learning** per controllare l'overfitting della rete. [7]
L'idea di questo approccio è quella di allenare una medesima rete su diversi problemi di classificazione. I primi layer della rete saranno condivisi tra i diversi task, controllando, così facendo, l'eccessiva specializzazione della rete nella prima fase di apprendimento (favorendo la generalizzazione dei risultati). Purtroppo, lavorando con i Data Generator di Keras, non è stato possibile implementare questo approccio.

Un metodo alternativo e piuttosto recente, ha a che fare con approcci di Deep Learning Probabilistici. Le *CNN Bayesiane*, implementabili attraverso la recente libreria `tfprobability` [8], sono delle varianti rispetto alle CNN che possono ridurre la possibilità di overfitting durante la fase di apprendimento per quanto riguarda dataset di dimensione *medio-piccola*.
L'idea alla base delle reti bayesiane risiede nella possibilità di modellare i pesi delle reti come una distribuzione (anzichè stimarli puntualmente), riuscendo ad ottenere, in output, non solo una stima, ma anche una misura di incertezza associata alla stima di previsione.

REFERENCES

1. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. journal computer vision **60**, 91–110 (2004).
2. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision,* (Ieee, 2011), pp. 2564–2571.
3. B. Thornton, "Audio recognition using mel spectrograms and convolution neural networks," (2019).
4. M. Dörfler, R. Bammer, and T. Grill, "Inside the spectrogram: Convolutional neural networks in audio processing," in *2017 international conference on sampling theory and applications (SampTA),* (IEEE, 2017), pp. 152–155.
5. Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS),* (IEEE, 2018), pp. 1–2.
6. H. Kim, H. Nam, W. Jung, and J. Lee, "Performance analysis of cnn frameworks for gpus," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS),* (IEEE, 2017), pp. 55–64.
7. S. Ruder, "An overview of multi-task learning in deep neural networks," arXiv preprint arXiv:1706.05098 (2017).
8. J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "Tensorflow distributions," arXiv preprint arXiv:1711.10604 (2017).