

UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA



DIGITAL SIGNAL AND IMAGE MANAGEMENT

Progetto Finale

Autori:

Confalonieri Riccardo, 830404, r.confalonieri5@campus.unimib.it

Mariani Ginevra, 829506, g.mariani34@campus.unimib.it

Mora Lorenzo, 825393, l.mora4@campus.unimib.it

Indice

1 Segnali mono-dimensionali	1
1.1 I dati	1
1.2 Feature Extraction	2
1.2.1 Zero Crossing Rate (ZCR)	2
1.2.2 Root-Mean-Square Energy (RMSE)	2
1.2.3 Spectral Centroids	3
1.2.4 Mel-frequency Cepstrum (MFCC)	3
1.2.5 Combo	4
1.3 Modelli	4
1.3.1 Support Vector Machine	4
1.3.2 Random Forest	5
1.3.3 Convolutional Neural Network	6
1.3.4 Spectrogram Classification	7
2 Segnali bi-dimensionali: face recognition	10
2.1 Il dataset	10
2.2 Approccio metodologico	11
2.2.1 Configurazione del training	11
2.2.2 Data augmentation	11
2.3 VGG16	12
2.4 MobileNet-V2	13
2.5 VGGFace	14
2.6 Risultati e valutazione	15
3 Retrieval	17
3.1 Approccio metodologico	17
3.2 Retrieving con MobileNet-V2	17
3.3 Retrieving con VGGFace	18
3.4 Valutazione dei risultati	18

1 Segnali mono-dimensionali

Il task da portare a termine consiste nell'implementazione di alcuni modelli, sia di Machine Learning che Deep Learning, in grado di determinare correttamente la persona che sta parlando a partire da una registrazione audio.

1.1 I dati

Il dataset contenente i segnali mono-dimensionali è stato realizzato ad hoc per il progetto. In particolare, tramite acquisizione automatica sono stati raccolti 300 audio, 100 per ogni componente del gruppo, da cinque secondi, di cui soltanto una piccola porzione con un leggero rumore di sottofondo. Per la realizzazione delle registrazioni ogni componente ha recitato diversi paragrafi tratti dal libro “L'amore ai tempi del colera” di Marquez e dal libro “Il giovane Holden” di Salinger.

A seguito di alcune analisi e verifiche è emerso che la durata degli audio fosse eccessiva, in quanto i modelli non erano in grado di apprendere o andavano in *overfitting* sulle features in input. Per tale ragione gli audio sono stati divisi in segmenti da due secondi, così che per ogni audio sono stati generati quattro segmenti, ognuno con una sovrapposizione di un secondo rispetto al precedente. In questo modo dai 300 audio di partenza ne sono stati ottenuti 1200. In aggiunta, in una seconda fase, sono state applicate agli audio delle tecniche di **Data augmentation** per incrementare ulteriormente i dati, così da cercare di migliorare le performance dei modelli. Gli audio sono stati *modificati* attraverso due tecniche: *pitch shifting* e *time stretching*. La prima consiste nel modificare il *pitch* originale dell'audio aumentandolo o diminuendolo secondo il parametro `n_steps`, fattore di riduzione del pitch impostato a -0.8 . Il secondo approccio consiste invece nel modificare la *velocità* dell'audio, creando una copia velocizzata (`rate = 1.5`) ed una rallentata (`rate = 0.5`). Per questa ultima modifica è stato necessario applicare nuovamente la funzione di taglio. A seguito di queste modifiche si è arrivati ad un totale di 4800 audio a disposizione.

Di seguito, in figura 1, si riporta una raffigurazione dell'ampiezza di uno degli audio registrati inizialmente, da cinque secondi, in funzione del tempo.

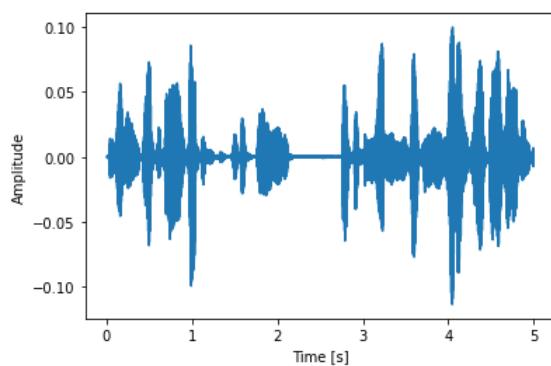


Figura 1: Visualizzazione grafica dell'ampiezza del segnale audio in funzione del tempo

Segue anche la visualizzazione dello spettrogramma dello stesso audio, riporta in figura 2, in quanto uno degli approcci testati per la classificazione ne prevede l'utilizzo.

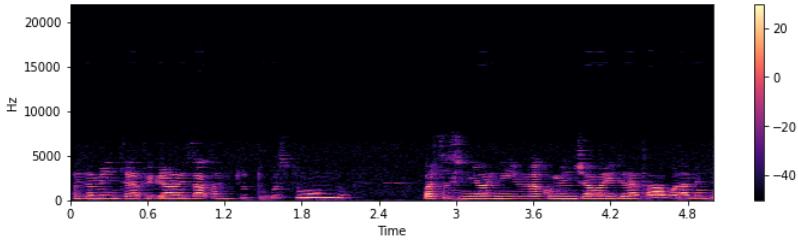


Figura 2: Spettrogramma di un segnale audio contenuto nel dataset

1.2 Feature Extraction

L'estrazione delle features da un audio è un passaggio fondamentale per l'addestramento di qualsiasi modello di Machine Learning e Deep Learning. Esistono diversi metodi di estrazione delle features, perciò se ne sono testati diversi al fine di identificare il più adatto. Spesso una sola feature potrebbe non bastare: si è deciso di combinare diversi metodi al fine di identificare la migliore combinazione. Di seguito vengono descritti gli approcci sperimentati, tra i quali è presente la *deviazione standard* (SD) che non verrà vista nel dettaglio, in quanto non è altro che la deviazione standard del segnale.

1.2.1 Zero Crossing Rate (ZCR)

ZCR è una misura pesata del numero di volte in cui il segnale cambia segno in un frame, cioè quando passa da positivo a zero e poi in negativo o da negativo a zero e poi in positivo. In altre parole, è il numero di volte in cui un segnale attraversa l'asse orizzontale. ZCR è alto per i suoni rumorosi (senza voce) e basso per i suoni tonali (con voce). Viene utilizzato per calcolare la *levigatezza* o *l'uniformità* del segnale. Segue, in figura 3, la visualizzazione delle features estratte con ZCR dell'audio presentato nel precedente paragrafo.

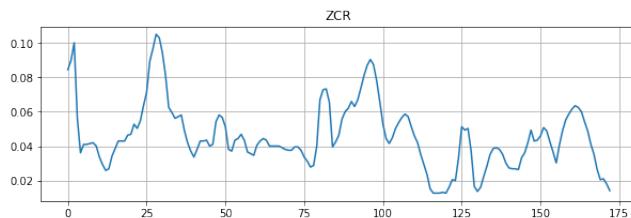


Figura 3: Zero-crossing rate dell'audio selezionato

1.2.2 Root-Mean-Square Energy (RMSE)

L'energia di un segnale corrisponde alla magnitudine totale del segnale. Per i segnali audio, corrisponde più o meno a quanto esso è forte. L'energia in un segnale è definita come $energy = \sum_n |x(n)|^2$, di conseguenza RMSE viene calcolato come:

$$\sqrt{\frac{1}{N} \sum_n |x(n)|^2}$$

In figura 4 è riportato l'RMSE dell'audio presentato nel primo paragrafo.

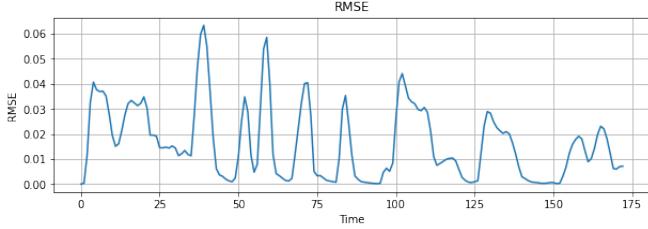


Figura 4: RMSE dell’audio selezionato

1.2.3 Spectral Centroids

Il centroide spettrale è una misura utilizzata nell’elaborazione del segnale digitale per caratterizzare uno spettro e indica dove si trova il centro di massa dello spettro. Percettivamente, ha una connessione robusta con l’impressione di brillantezza di un suono. Come per le altre features si riporta in figura 5 l’estrazione ottenuta per l’audio presentato precedentemente.

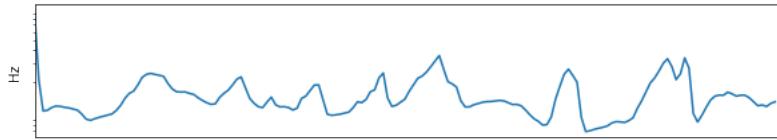


Figura 5: Centroide spettrale dell’audio selezionato

1.2.4 Mel-frequency Cepstrum (MFCC)

Si tratta di una delle tecniche di features extraction più utilizzate. Nell’elaborazione del suono, il *cepstrum* a frequenza mel è una rappresentazione dello spettro di potenza a breve termine di un suono, basato su una trasformata coseno lineare di uno spettro di potenza logaritmica su una scala *mel* non lineare di frequenza. Gli MFCC producono una rappresentazione compatta di un segnale audio e si differenziano da altre caratteristiche *cepstrali* nelle bande di frequenza che sono sulla scala mel, tuttavia i valori MFCC non sono molto robusti in presenza di rumore additivo.

Al fine di ottenere delle features compatte, adatte per essere gestite da classificatori diversi, si è deciso di limitare il numero di componenti a 13, come è visionabile nell’immagine 6 sottostante.

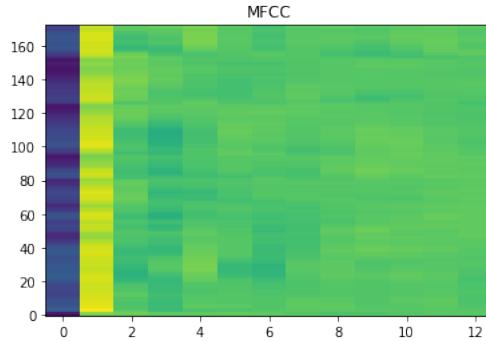


Figura 6: Spettro MFCC dell’audio selezionato

1.2.5 Combo

Come anticipato sono state combinate insieme, tramite concatenazione, le diverse features per ottenere una rappresentazione più robusta del segnale audio. Più specificatamente sono state testate le seguenti combinazioni:

- `combo1` è la combinazione di deviazione standard, RMSE e MFCC;
- `combo2` concatena deviazione standard, ZCR e spectral centroids;
- `combo3` combina spectral centroids e MFCC.

1.3 Modelli

Per completare il task di riconoscimento audio sono stati testati i seguenti approcci:

- Support Vector Machine (SVM);
- Random Forest (RF);
- Rete convoluzionale implementata *from scratch*;
- Spectrogram classification.

I primi tre approcci richiedono come input le features estratte con i metodi precedentemente presentati. Per il terzo approccio si è scelto di utilizzare `mfcc` in quanto permette l'implementazione di una rete neurale convoluzionale. Per i primi due invece sono stati testati tutti gli estrattori al fine di trovare quello che garantiva la migliore accuratezza. I risultati, per dati aumentati, sono raccolti in tabella 1.

Tabella 1: Accuratezza dei modelli con diversi feature extractors

	SVM	RF
<i>Combo 1</i>	0.83	0.79
<i>Combo 2</i>	0.48	0.55
<i>Combo 3</i>	0.53	0.79
<i>MFCC</i>	0.83	0.8
<i>Spectral Centroids</i>	0.48	0.51
<i>RMSE</i>	0.53	0.38
<i>SD</i>	0.43	0.38
<i>ZCR</i>	0.37	0.31

Basandosi su questi risultati `mfcc` è stato scelto sia per SVM che RF. Lo stesso risultato è stato ottenuto anche per i dati non aumentati, per cui non sono stati riportati i risultati.

1.3.1 Support Vector Machine

Il primo approccio utilizzato prevede l'addestramento di una Support Vector Machine con *kernel radiale* e come anticipato è stato utilizzato `mfcc` per estrarre le features dagli audio. Per garantire le migliori performance possibili sono state diverse configurazioni dei parametri C e γ , concludendo che la migliore fosse data da $C = 18$ e $\gamma = 10^{-9}$. In aggiunta, l'addestramento è avvenuto in diversi step: prima sui dati semplicemente segmentati, poi su quelli segmentati e *normalizzati* ed infine sui dati segmentati ed *aumentati*. Questo perchè si è voluta identificare la miglior configurazione possibile dei dati col modello. La miglior configurazione trovata è quella che prevede l'addestramento di dati segmentati e aumentati.

Si è deciso di non usare la normalizzazione dei dati, nonostante i risultati migliori, in quanto si verificava un chiaro *overfitting*.



Figura 7: Confusion Matrix ottenuta con SVM

Il modello raggiunge un'accuratezza sul test set pari a 0.83, infatti anche osservando la figura 7 si osserva come la maggior parte delle predizioni siano corrette. Mentre sul training set si è raggiunta un'accuracy leggermente maggiore pari a 0.86. In particolare l'algoritmo riesce a classificare più abilmente la voce di Ginevra, mentre fa più fatica con Lorenzo e Riccardo che spesso vengono confusi. Questo viene confermato anche in fase di test, dove la voce di Ginevra viene sempre predetta correttamente al contrario di quella di Lorenzo e Riccardo.

1.3.2 Random Forest

Il secondo modello ha visto l'addestramento di un algoritmo di Random Forest, un metodo di *bagging* molto usato per la classificazione. Ancora una volta è stato usato `mfcc` per l'estrazione delle features, in quanto otteneva l'accuratezza più alta. Dopo vari test è emerso che il numero ottimale di *stimatori* fosse 100 e con una profondità di 12. Come per SVM l'addestramento è stato suddiviso in step e si è concluso che i dati più adatti fossero quelli segmentati e aumentati in quanto garantivano le migliori performance.

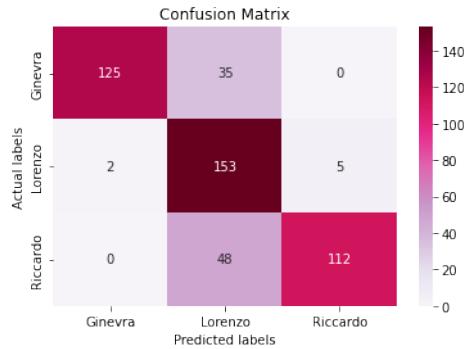


Figura 8: Confusion Matrix ottenuta con Random Forest

Tuttavia, uno dei principali svantaggi degli alberi decisionali è che sono molto inclini a un adattamento eccessivo: funzionano bene sui dati di addestramento, ma non sono così flessibili per fare previsioni su campioni nuovi. Questo è stato riscontrato anche dal particolare modello implementato in cui il training set raggiunge un'accuracy pari ad 0.98 mentre il test set, i cui risultati sono osservabili in figura 8, raggiunge un'accuracy di 0.81. Quest'ultima seppur buona riporta a emergere l'*overfitting* del training. Inoltre in fase di test RF ha commesso molti più errori rispetto a SVM, ragione per cui è stata escluso dalla demo live.

1.3.3 Convolutional Neural Network

Il terzo approccio sperimentato ha riguardato l'implementazione di una rete neurale convoluzionale che prende come input le features estratte con MFCC sui dati segmentati. Più specificatamente l'input è una matrice di dimensione 13×173 . L'implementazione della rete è visibile nell'immagine 9 sottostante.

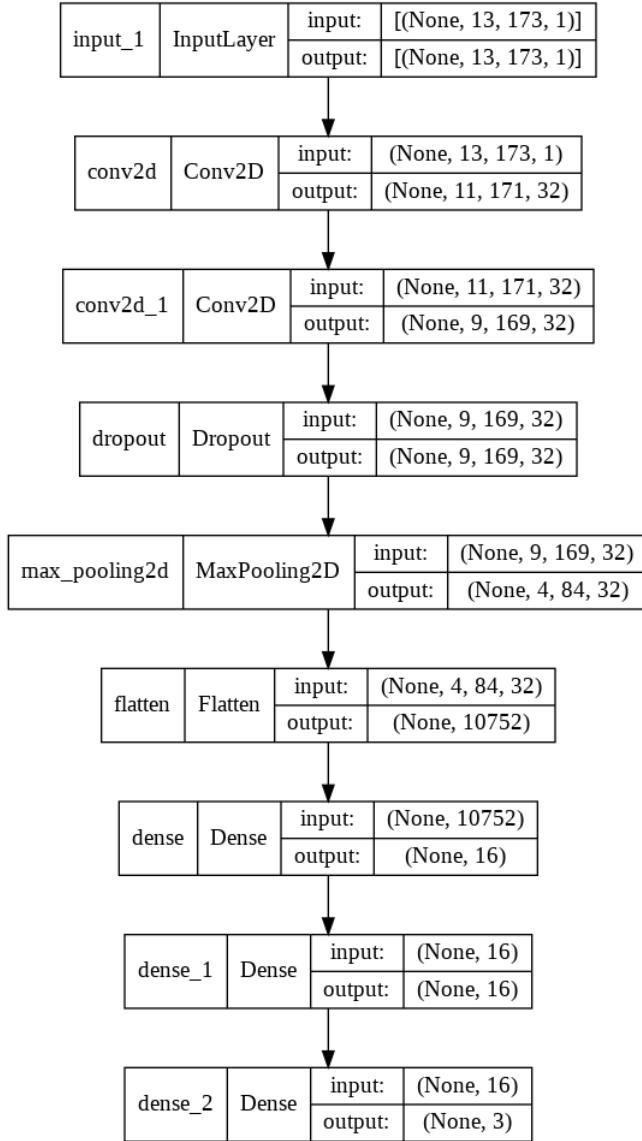


Figura 9: Rete *from scratch* basata su MFCC

La funzione di attivazione scelta per i layer convoluzionali e densi, ad eccezione dell'ultimo, è la *relu*. Per l'ultimo layer fully connected viene usato la funzione *softmax*, poichè il problema di classificazione è multi classe. La rete viene ottimizzata da *Adam* avente learning rate pari a 0.0001 e la funzione di loss calcolata è la Categorical CrossEntropy.

Il modello è stato addestrato con Early Stopping che ha permesso di ridurre il numero di epoche inizialmente impostato, 100, a sole 15 epoche ottenendo un valore massimo di accuratezza sul validation set pari a 0.9583 e uno minimo di loss pari a 0.1807.

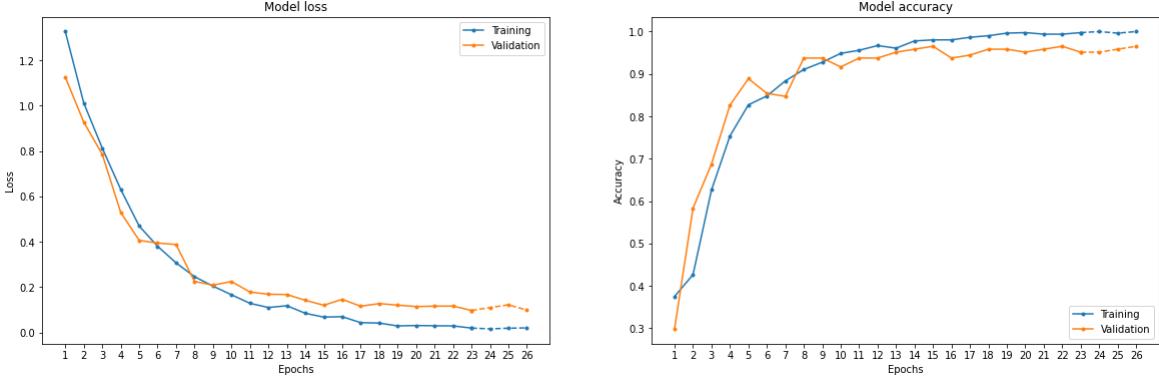


Figura 10: Addestramento della rete *from scratch* basata su MFCC

Dalla figura 10 sovrastante si osserva che il modello soffre di un leggero overfitting nelle ultime fasi dell’addestramento, globalmente però l’addestramento sembra buono, infatti l’accuratezza su test è di 0.97 su test set. Infine osservando la matrice di confusione nella figura 11 si può concludere che risultati ottenuti siano molto validi. Infatti Ginevra viene classificata sempre correttamente, Lorenzo e Riccardo invece vengono confusi soltanto poche volte.

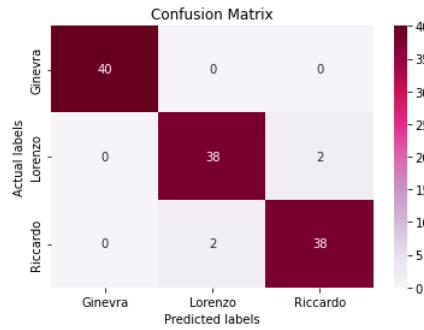


Figura 11: Confusion matrix ottenuta con la rete *from scratch* basata su mfcc

La stessa rete è stata sviluppata per i dati aumentati, ma le performance sono leggermente peggiorate per cui si è deciso di escluderla dal report.

1.3.4 Spectrogram Classification

L’ultimo approccio sperimentato consiste nella classificazione di immagini relative allo spettrogramma dei segnali audio; si passa quindi da segnali mono-dimensionali a segnali bi-dimensional. Lo *spettrogramma* è una rappresentazione visiva di come lo spettro delle frequenze di un segnale audio vari nel tempo. L’analisi dello spettro fornisce una vasta gamma di informazioni sui segnali che possono essere utilizzati per le attività di classificazione e discriminazione. Esso viene calcolato usando la trasformata di Fourier: i dati campionati digitalmente, nel dominio del tempo, vengono suddivisi in *chunks*, solitamente sovrapposti. Infine viene calcolata la trasformata di Fourier per determinare l’entità dello spettro di frequenza per ciascun chunk.

Una volta convertiti tutti gli audio (segmentati e aumentati) in spettrogrammi e convertiti in scala di grigi, sono stati suddivisi in training set (90%, 4320 immagini) e testing set (10%, ovvero 480 immagini). A questo punto è stato possibile implementare una rete convoluzionale che è visionabile in figura 12.

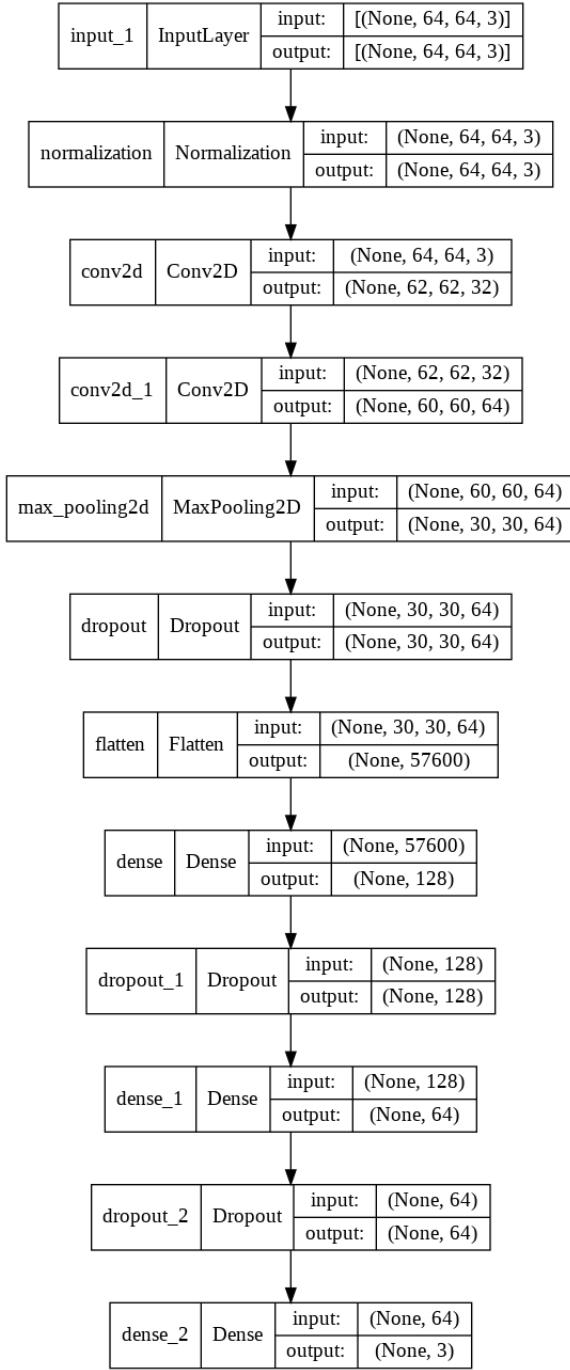


Figura 12: Rete convoluzionale per la classificazione di spettrogrammi

I layer convoluzionali e densi, ad eccezione dell'ultimo sono attivati dalla funzione *relu*; l'ultimo invece usa la *softmax*. Il modello viene ottimizzato da *adam* e la loss che viene calcolata è la *categorical crossentropy*, essendoci tre classi da predire. Infine il modello viene regolarizzato tramite Early Stopping, una tecnica che permette di bloccare l'addestramento in assenza di miglioramenti, salvando i valori dei pesi ottimali. Il modello, dalle 25 epoche impostate inizialmente, grazie all'Early Stopping, ne ha eseguite solo 11: già dalla sesta epoca non ci sono stati miglioramenti. Il valore migliore di accuratezza raggiunto sul validation set è pari a 0.9583, quello di loss invece è uguale a 0.1671.

Nella figura 13 sono visionabili i risultati ottenuti su test set.



Figura 13: Confusion matrix ottenuta con la rete *from scratch* per gli spettrogrammi

Dall’osservazione della matrice di confusione si nota come gli audio di Ginevra sono quelli che il modello riesce a riconoscere più abilmente, a differenza di quelli di Lorenzo che spesso vengono confusi con quelli di Riccardo. Infatti testando nuove registrazioni raccolte in un secondo momento, la rete reagisce bene con audio di Ginevra, mentre fatica a riconoscere correttamente quelli di Lorenzo e Riccardo. Questo può essere dovuto al fatto che le due voci maschili possono avere dei tratti simili che il classificatore non riesce a distinguere perfettamente. Un’altra motivazione può essere legata alle condizioni di registrazione degli audio e al ritmo del parlato che potrebbe essere diverso da quello usato per registrare gli audio di partenza.

2 Segnali bi-dimensionali: face recognition

Lo scopo di questa parte di progetto è di realizzare dei modelli abili nel riconoscere i volti dei diversi componenti del gruppo di lavoro. Dopo alcuni tentativi con classificatori classici si è deciso di utilizzare delle reti neurali e, dato il particolare contesto con immagini, la scelta è ricaduta sulle CNN. Per lo sviluppo si è scelto di considerare variazioni nell'espressione facciale dei soggetti e nelle condizioni di luce.

2.1 Il dataset

Il dataset utilizzato per lo sviluppo di questa parte di progetto è stato realizzato ex novo a partire dalle foto dei componenti del gruppo. Nello specifico, tramite un'acquisizione automatica, sono state scattate per ogni componente del gruppo il seguente numero di foto:

- 80 foto da utilizzarsi come training.
- 20 foto per il validation set.
- 17 foto come test set.

Tutte le immagini sono state scattate in un ambiente casalingo, con però l'introduzione di variazioni a livello di luci ed espressioni del volto come si può notare dagli esempi mostrati in Figura 14. Inoltre le diverse foto sono state caricate in sottocartelle, il cui nome corrisponde al nome del componente presente nella foto, per ottimizzarne il caricamento tramite i metodi appositi di Keras [1].

Ottenute le immagini necessarie per lo sviluppo del progetto si è proceduto con una fase di preprocessing volta ad estrarre il *crop* dei volti dalle immagini. Per farlo sono stati diversi **face detector**, tuttavia la scelta finale è ricaduta su di uno su un modello neurale offerto dalla libreria *dlib* [2]. Estratti i crop per tutte le immagini si è potuto procedere con l'effettivo caricamento delle stesse in ambiente Python [3]. Durante questa fase le immagini sono state caricate in batch di 16 e riscalate ad una dimensione pari a 224x224px per *matchare* correttamente con gli input dei modelli neurali utilizzati successivamente. Si segnala inoltre che la label di riferimento è estratta in modo automatico sfruttando la struttura a sottocartelle appositamente creata e che i batch sono stati riordinati in modo randomico così da evitare che contenessero solo immagini sequenziali di una sola persona.

Infine sono state utilizzate anche ulteriori immagini dei diversi soggetti scattate durante gli anni in diverse ambientazioni e con diverse caratteristiche per testare, in un secondo momento, quanto i modelli siano abili a riconoscere il soggetto con immagini reali e non realizzate ad hoc.

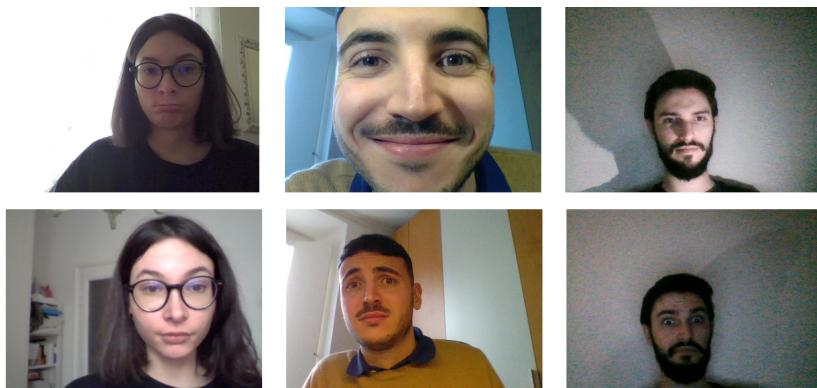


Figura 14: Esempio di immagini utilizzate per il task di face-recognition

2.2 Approccio metodologico

Date le poche immagini a disposizione per il training dei modelli la scelta per l'implementazione delle architetture e ricaduta sin da subito su approcci di fine tuning. Quest'approccio infatti permette di utilizzare reti precedentemente addestrate per un altro task di riconoscimento di immagini, opportunamente modificate per adattarsi al nuovo task di classificazione. In particolare si è scelto di utilizzare tre diverse reti:

- VGG116
- MobileNet-V2
- VGGFace

Delle quali le prime due sono state addestrate su un task piuttosto generico (*i.e.* Imagenet), mentre l'ultima rete è stata scelta proprio perché addestrata su un task di riconoscimento del volto e quindi ci si aspetta che le features estratte dai primi layer siano molto più specifiche. Tutti i modelli proposti sono stati tagliati alla fine dei layer convoluzionali, i quali non verranno addestrati nuovamente per il nuovo task. Per ognuno dei modelli creati è stato inoltre applicato il preprocessing specifico, in modo che le immagini potessero essere correttamente trattate dai layer predefiniti. Infine, per poter utilizzare correttamente questi modelli, è stato necessario effettuare un reshaping dei crop dei volti ad una dimensione pari a 224x224px, mantenendo comunque i tre canali colore. Questa modifica è dovuta al fatto che tutte e tre le reti scelte sono state preaddestrate su immagini di queste dimensioni, si potrebbero quindi generare errori con input diversi soprattutto nei layer convoluzionali. Nelle sezioni successive si analizzerà più nel dettaglio le scelte ed i risultati ottenuti per le singole architetture.

2.2.1 Configurazione del training

Per quanto concerne l'addestramento dei modelli si è deciso di fissare un massimo di 100 epoch e, utilizzando l'approccio dell'EarlyStopping per interrompere il processo dopo un certo numero di epoch senza alcun miglioramento della validation loss. Tale approccio, oltre a fermare l'apprendimento, ripristina anche i pesi corrispondenti all'epoca migliore individuata, di fatto eliminando il training delle ultime tre epoch. Mentre per quanto riguarda il caricamento delle immagini, utilizzando i metodi ad hoc forniti dalla libreria Keras [1], si sono caricati tutti gli esempi a disposizione in batch di dimensione 16. Inoltre in tutti i test set si sono mantenute le tre classi bilanciate.

2.2.2 Data augmentation

Avendo a disposizione poche immagini, e la maggior parte con caratteristiche molto simili, si è deciso di implementare delle tecniche di data augmentation. Tali tecniche hanno permesso sia di ampliare il dataset a disposizione senza effettivamente raccogliere nuovi elementi, creando delle copie modificate, sia di creare dei modelli più robusti e flessibili a possibili cambiamenti di luce o scene di acquisizione.

Nello specifico si è scelto di introdurre le seguenti distorsioni alle immagini:

- *Random flip*: che capovolge casualmente l'immagine in orizzontale. Non è stato considerato il caso del flip verticale in quanto considerato fuori contesto per il particolare task. Si suppone infatti di avere immagini con il viso non capovolto.
- *Random rotation*: che ruota l'immagine in un range di gradi predefinito.

- *Random Zoom*: che effettua uno zoom dell’immagine in un certo range predefinito sia in altezza che in larghezza. In particolare, con i parametri scelti, si effettuano sia degli zoom-in che degli zoom-out utili quando il soggetto è molto lontano o molto vicino alla camera in fase di acquisizione.
- *Random contrast*: regola il contrasto di un’immagine in base a un fattore casuale. Il contrasto viene regolato indipendentemente per ciascun canale di ciascuna immagine durante il training.

Queste tecniche sono state applicate alle sole immagini di training mentre non vengono applicate in fase di inferenza. I parametri specifici utilizzati verranno poi analizzati singolarmente per i singoli modelli.

2.3 VGG16

VGG16 è una delle prime architetture CNN proposte, nonché una delle più famose ed utilizzate. L’architettura della rete è caratterizzata da stack di layer convoluzionali, con dimensione del kernel 3x3, applicati sequenzialmente. Questo modello è caratterizzato da un numero molto elevato di parametri e di operazioni, e permette di raggiungere una top-1 accuracy pari circa allo 0.73 sul task originale imagenet.

Come menzionato precedentemente, questa rete è stata tagliata alla fine dei blocchi convoluzionali, rimuovendo dunque i tre layer fully connected finali che sono stati sostituiti dai seguenti layer aggiuntivi:

- Fully connected con 32 neuroni e dropout rate 0.2;
- Fully connected con 16 neuroni;
- Fully connected finale con 3 neuroni che mappa alla dimensione dell’output.

Per tutti i layer aggiuntivi è stata usata come funzione di attivazione *relu*. Durante la fase di addestramento soltanto i layer aggiuntivi sono stati impostati come trainabili mentre i pesi dei layer relativi alla struttura base di VGG16 sono stati bloccati. Mentre per quanto riguarda i layer di data augmentation sono stati individuati, a seguito di alcune prove, i parametri ottimali riportati in tabella:

Tabella 2: Parametri dei layer di data augmentation per VGG16

Layer	Factor
Random flip	Horizontal
Random rotation	$[-0.35, 0.35]$
Random zoom in altezza	$[-0.6, 0.6]$
Random zoom in larghezza	-
Random contrast	$[0.2, 1.8]$

Complessivamente si è dunque arrivati a realizzare una struttura con: 15.518.115 parametri totali di cui trainabili 803.427.

Questa rete ha ottenuto dei buoni risultati, sul validation set, raggiungendo una loss di 0.07 e un’accuracy di 0.966. Inoltre il modello ha richiesto solo 16 epoche di training, di cui le ultime tre eliminate per via dell’EarlyStopping che ripristina i pesi dell’epoca migliore. La funzione di ottimizzazione utilizzata è *Adam* con un learning rate di $1e - 04$. L’andamento del training è riportato in Figura 15 e non segnala particolari problemi né di underfitting né di overfitting.

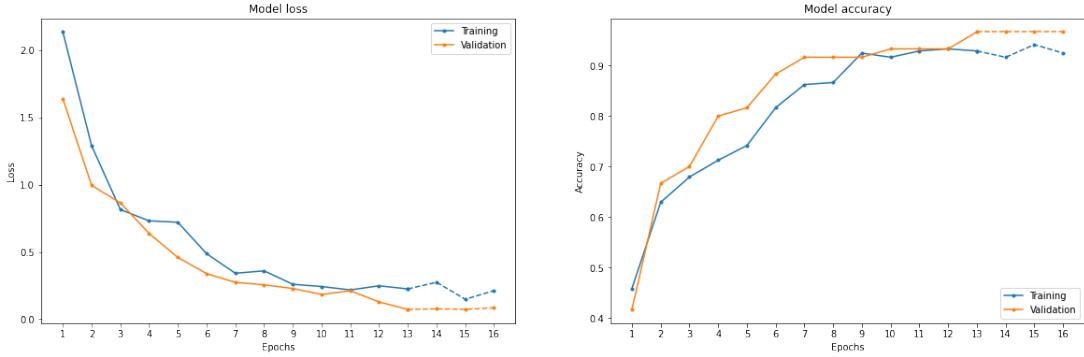


Figura 15: Andamento di loss e accuracy durante l’addestramento della rete VGG-16

2.4 MobileNet-V2

Questa architettura è stata sviluppata nel 2017 da un gruppo di ricercatori di Google [4]. È fortemente ottimizzata per essere utilizzata su dispositivi mobili, il numero di parametri e operazioni richieste è infatti minimo e, inoltre, anche il peso del modello, 14MB, è ideale per essere utilizzato su dispositivi con poca memoria. Nonostante questo, sul task originale Imagenet, il modello riesce a raggiungere quasi la stessa accuracy di altri modelli più complessi come VGG. Anche in questo caso la rete preaddestrata è stata tagliata prima dei blocchi fully connected, e i layer rimasti sono stati bloccati per evitare il loro addestramento. I layer finali sono stati sostituiti da un solo fully connected che mappa alla dimensione di output. Il modello finale è quindi composto da: 2.261.827 parametri totali di cui solo 3.843 trainabili. La rete ottiene risultati leggermente migliori rispetto alla rete VGG16, con una validation loss pari a 0.058 e una validation accuracy pari a 0.983. Tuttavia il numero di epoche complessive richieste aumenta, considerando le tre eliminate successivamente dall’EarlyStopping, pari a 28. Anche i parametri dei layer di data augmentation, riportati in Tabella 3, sono leggermente diversi:

Tabella 3: Parametri dei layer di data augmentation per MobileNet-V2

Layer	Factor
Random flip	Horizontal
Random rotation	$[-0.4, 0.4]$
Random zoom in altezza	$[-0.5, 0.5]$
Random zoom in larghezza	$[-0.5, 0.5]$
Random contrast	$[0.2, 1.8]$

Per l’addestramento di questa rete è stata utilizzata come funzione di ottimizzazione RMSProp con un learning rate pari a $1e - 03$ in quanto si è dimostrata essere più performante rispetto ad Adam. Analizzando l’andamento globale del training, riportato in figura 16, sembra poter esserci un leggero underfitting se si considera la metrica della loss.

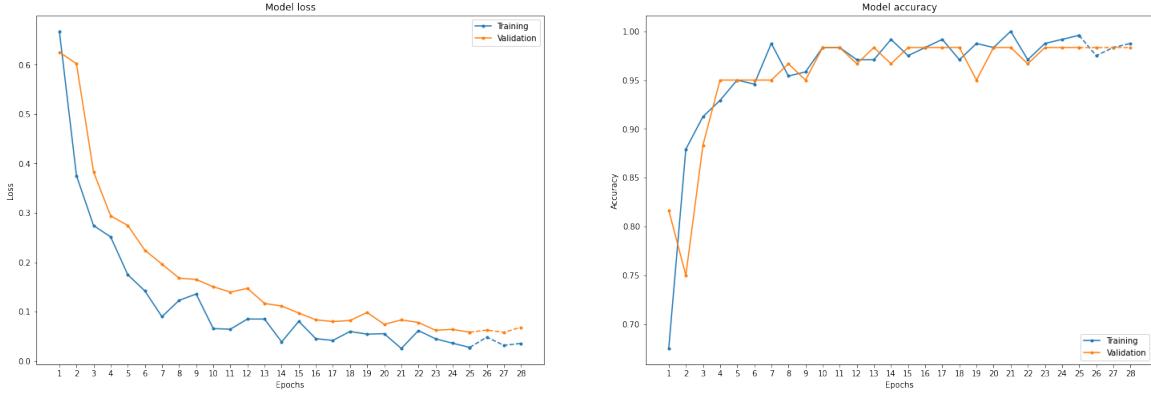


Figura 16: Andamento di loss e accuracy durante l’addestramento della rete MobileNet-V2

2.5 VGGFace

Questa particolare rete utilizza la struttura base della rete VGG ed è stata proposta nel 2015 da Omkar M. Parkhi et al. [5]. A differenza delle precedenti i pesi sono stati addestrati su un dataset specifico contenente i volti di più di 2.6 milioni di persone. Questo fa sì che sin dai primi layer convoluzionali le features siano molto più specifiche, rispetto alle altre architetture analizzate per il task di riconoscimento del volto.

Per l’addestramento di questa rete è stata seguita la medesima logica, ovvero sono stati tagliati i layer finali e bloccati tutti i layer rimanenti per preservare i pesi già esistenti. Infine si sono aggiunti due nuovi layer fully connected rispettivamente con 64 e 32 neuroni. In totale la rete così costruita è composta da 145.172.929 parametri di cui 170.051 addestrabili: la rete ottenuta è molto grossa e pesante. Come funzione di ottimizzazione invece si è utilizzato *Adam* con un learning rate pari a $1e - 04$.

Infine si riporta che in questo specifico caso, sebbene testati, non sono stati inseriti i layer di data augmentation, in quanto si è osservato che la rete avesse già delle performance ottimali anche con immagini fuori dagli esempi di training. Date le alte prestazioni è stato necessario cambiare leggermente la configurazione dell’EarlyStopping, ritenendo valide anche epoche con miglioramenti minimi, 0.0001, di validation loss, per questo motivo invece delle solite tre epoche si è scelto di far terminare l’apprendimento dopo solo due epoche peggiorative.

L’andamento del training, riportato in figura 17, mostra come già dalle primissime epoche i risultati fossero ottimali. Nel complesso l’addestramento è prorogato per 16 epoche, di cui le ultime due eliminate per via dell’EarlyStopping, raggiungendo una validation loss pari a 0.00098 ed una accuracy massima di 1.

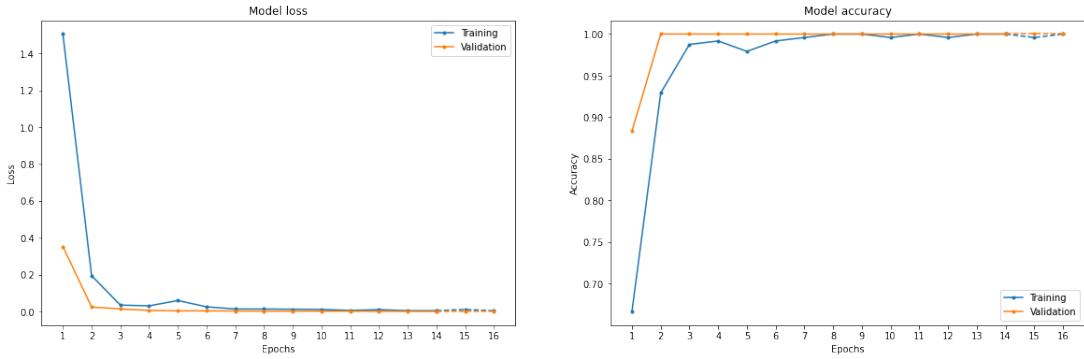


Figura 17: Andamento di loss e accuracy durante l’addestramento della rete VGGFace

2.6 Risultati e valutazione

Per valutare le capacità predittive dei diversi modelli creati sono stati utilizzati gli appositi test set creati in fase iniziale di acquisizione dati. In particolare le diverse architetture sono state testate sia su un test set con immagini simili a quelle utilizzate per il training, ovvero scattate in ambiente casalingo, ed un secondo test set con immagini reali acquisite in contesti vari.

Le figure 18, 19 e 20 riportano rispettivamente gli scores ottenuti sul test set standard dalle reti MobileNet-V2, VGG16 e VGGFace. È possibile notare come entrambe le reti con i pesi di *imagenet* abbiano alcuni errori di predizione, anche se minimi. Analizzandoli più nel dettaglio si è notato che questi errori sono dovuti a piccoli cambiamenti, come ad esempio la presenza o meno di occhiali da vista. Mentre la rete VGGFace come ci si aspettava dato l'ottimo training riesce a prevedere correttamente tutti gli esempi forniti.

Questi risultati sono stati ulteriormente confermati dal secondo test set con le immagini scattate fuori dal contesto di training, nello specifico MobileNet-V2 classifica erroneamente 6 immagini su 17, VGG16 predice in modo errato 5 immagini su 17, infine la rete VGGFace commette un solo errore di predizione riuscendo a classificare correttamente anche le immagini più complesse.

Questo porta a concludere che i pesi ottenuti con *imagenet* possano essere utilizzati per questo task solo se si è sicuri di ricevere in input immagini molto controllate, altrimenti è più opportuno utilizzare i pesi derivati da task più simili come VGGFace che permettono di costruire modelli molto più robusti e flessibili ai cambiamenti.

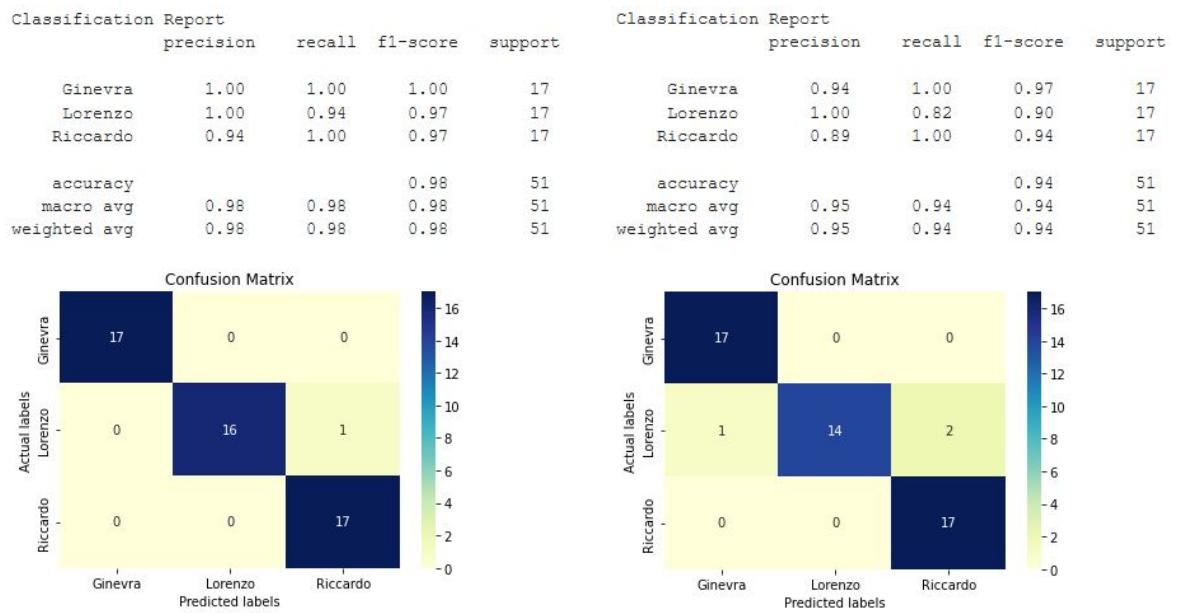


Figura 18: Risultati della rete MobileNet-V2 sul test set

Figura 19: Risultati della rete VGG16 sul test set

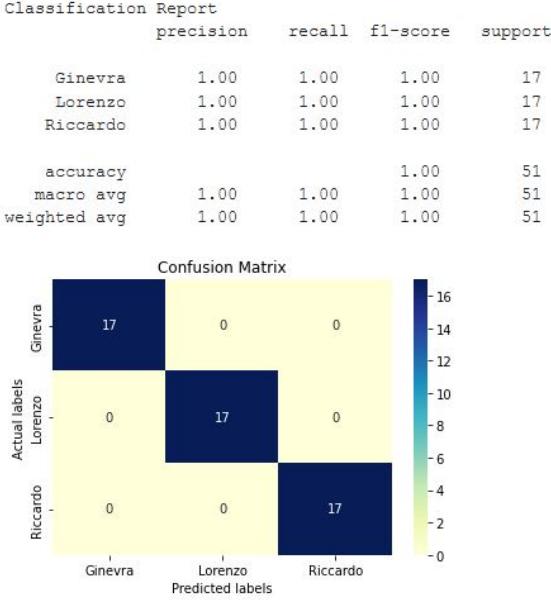


Figura 20: Risultati della rete VGGFace sul test set

Un'altra importante metrica che si è deciso di valutare è stata il tempo di predizione per una singola immagine¹. Si è scelto di valutare anche questa metrica in quanto in alcuni casi potrebbe essere richiesto di effettuare la predizione in real-time, ad esempio per lo sblocco del telefono. In tabella 4 sono riportati i tempi di predizione per i diversi modelli. Rispetto a questa metrica è difficile trarre una conclusione unica e sempre valida, infatti dipende molto dal contesto applicativo e dalle risorse. Ad esempio MobileNet-v2 risulta essere quella più ideale per task real-time, anche se i suoi errori non la renderebbero valida, ad esempio, per sbloccare telefoni in quanto permetterebbero l'accesso anche a persone sconosciute che ovviamente non è desiderabile.

Tabella 4: Tempo di predizione di una singola immagine

Modello	Tempo (secondi)
VGG16	1.694
MobileNet-V2	0.125
VGGFace	1.868

¹Il tempo di predizione è stato calcolato come media rispetto al tempo necessario per predire un intero batch a parità di potenza computazionale. Inoltre è strettamente dipendente dalle risorse computazionali a disposizione.

3 Retrieval

L’obiettivo di questo task è individuare i volti dei vip che risultano essere più simili ad una data immagine di **query**. In questo caso sono stati nuovamente utilizzati i volti dei componenti del gruppo come query, mentre per il dataset di retrieving è stato utilizzato quello consigliato per il progetto. Quest’ultimo contiene 1580 vip ed oltre 200.000 immagini in varie pose, dimensioni e in diversi contesti. Quindi per questo task non è stato creato alcun nuovo dataset.

3.1 Approccio metodologico

Per quest’ultima parte si è deciso in primo luogo di estrarre le features per le immagini utilizzando due CNN: MobileNet-V2 e VGGFace. Sono state volutamente scelte due architetture addestrate su dataset molto diversi per poter analizzare quanto questo aspetto impattasse nella qualità del retrieving. Ci si aspetta infatti che MobileNet-V2 non dia buoni risultati in quanto non particolarmente adatta a questo task.

Una volta scelte le architetture per la creazione delle features ci si è posti il dubbio su come processare correttamente le immagini. Si è quindi scelto di applicare prima un face detector alle immagini dei vip; questa scelta è dovuta al fatto che lo scopo è individuare i vip il cui volto è il più simile ad una data query e quindi calcolare le features su tutta l’immagine potrebbe portare a delle distorsioni e restituire similitudini dovute a sfondo e/o altre caratteristiche indesiderate. Quindi prima di calcolare le features, sia per le immagini dei vip ma anche successivamente per le query, è stato applicato il face detector *dlib* estraendo il crop del solo volto. Successivamente, avendo scelto dei modelli CNN preaddestrati, l’immagine è stata ridimensionata per *matchare* opportunamente con l’input di dimensioni 224x244px caratteristico per entrambi i modelli.

Infine, dopo aver effettuato alcuni test, si è deciso di estrarre le features soltanto per un subset di volti, in quanto il processamento richiedeva un grosso sforzo computazionale e tempistiche molto lunghe. Per questo motivo si è scelto di estrarre per ogni vip al più 20 immagini, questo permette di creare le features di riferimento per tutte le persone famose garantendo un retrieving il più possibile valido. Forzando infatti il numero massimo di immagini si è visto che si sarebbero esclusi a priori alcuni vip, questi però potrebbero essere proprio i più simili alla query in input. In totale si sono dunque ottenute le features di riferimento rispetto a 31.496 volti appartenenti a tutti i vip. Come ultima considerazione si segnala che per alcuni vip non si hanno a disposizione tutte le 20 immagini in quanto il face detector non è riuscito ad estrarre correttamente il volto della immagini disponibili. Questo è dovuto a molteplici fattori, tra cui volti molto lontani o in alcuni casi modificati, ad esempio ci sono immagini di copertina di album musicali in cui il volto del cantante contiene sovrapposizioni con altri oggetti.

Una volta costruite tutte le features si è utilizzata una struttura ad albero per effettuare una ricerca rapida ed ottimizzata delle immagini più somiglianti. Nello specifico ci si è avvalsi della libreria *Sklearn* [6] che permette di implementare la struttura denominata *kd-tree*.

3.2 Retrieving con MobileNet-V2

MobileNet-V2 ha permesso di estrarre 1024 features di riferimento per ogni singola immagine richiedendo circa cinquanta minuti per processare tutte le 31.496 immagini dei vip. Successivamente queste features sono state utilizzate come riferimento per il retrieving dei 10 volti più simili ad una query in input. Tuttavia, come ci si attendeva date le caratteristiche della rete, osservando i volti individuati per i diversi componenti del gruppo si è notato che non vi era molta somiglianza tra il volto in input e quelli restituiti, se non per la prima del-

le dieci immagini. Questo è dovuto al fatto che MobileNet-V2 è preaddestrata su un task completamente diverso rispetto al riconoscimento dei volti.

3.3 Retrieving con VGGFace

La libreria utilizzata per VGGFace permette di scegliere tra diversi modelli di base come architettura, dopo alcune prove effettuate si è scelto di utilizzare VGG come architettura di base e l'*average* come pooling. Questo ha permesso di estrarre 512 features di riferimento per ogni singola immagine, tuttavia, nonostante la minor dimensionalità, il tempo richiesto per la creazione di tutte le features è maggiore rispetto al caso precedente di circa trenta minuti. Questo incremento di tempo può essere spiegato dalla maggior complessità architettonica della rete.

Questa rete ha permesso un miglioramento nella qualità dei volti individuati come più simili; è possibile infatti notare diverse somiglianze rispetto alla query anche per le immagini successive rispetto alla prima. Analizzando la figura 21 riportante i cinque volti più simili per i diversi componenti del gruppo, è possibile notare che le caratteristiche principali siano simili. Ad esempio si nota la presenza di occhiali neri per ‘Ginevra’, ma anche similitudini nella forma del naso e nella posizione sopracciglia. Le medesime osservazioni sono valide anche per i componenti ‘Lorenzo’, i cui retrieval hanno un’espressione sorridente e spesso presentano baffi, e ‘Riccardo’ in cui i volti individuati presentano gli occhiali e barba.



Figura 21: Top-5 risultati per il task di retrieving

3.4 Valutazione dei risultati

Il task di retrieving è risultato computazionalmente più pesante del previsto nella fase di creazione iniziale delle features dei volti dei personaggi famosi. Questo, congiuntamente a problemi di eccesso di memoria e di tempo², ha portato alla scelta di ridimensionare il dataset iniziale considerando soltanto un subset di immagini per ogni singolo vip.

Per quanto concerne invece l’analisi dei risultati ottenuti, concentrandosi su VGGFace, ci si ritiene abbastanza soddisfatti. MobileNet-V2 ha infatti mostrato alcuni limiti dovuti, come ci si attendeva per via dei pesi pretrainati su altre tipologie di immagini rispetto ai volti. Un’ulteriore considerazione sui risultati riguarda il retrieving ottenuto per ‘Ginevra’ nel quale è presente più volte la stessa persona, *Nana Mouskouri*. Questo, in base alle specifiche richieste, può essere un risultato non voluto in quanto all’atto pratico non individua 10 personaggi famosi distinti e può quindi essere oggetto di miglioramenti futuri. Infine, durante i diversi tentativi, si è notato come la qualità delle immagini dei vip e della query in input incida molto sui risultati. Anche questo aspetto può essere un importante fattore per migliorare le performance del retrieving.

²Se si utilizza Google Colab dopo un certo intervallo di tempo la sessione viene terminata automaticamente

Riferimenti bibliografici

- [1] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [2] D. E. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.
- [3] P. C. Team, *Python: A dynamic, open source programming language*, Python Software Foundation, 2015. [Online]. Available: <https://www.python.org/>.
- [4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” 2017.
- [5] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in *British Machine Vision Conference*, 2015.
- [6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [7] S. Bianco, “Dispense e slide del corso digital signal and image management,” 2021.