Contest Visione Artificiale: Gruppo 18

Vittorio Fina, Giovanni Puzo, Vincenzo Russomanno, Salvatore Ventre.

{s.ventre, g.puzo, v.russomanno, v.fina}@studenti.unisa.it

1 Introduzione

Le applicazioni che coinvolgono l'analisi dei dati tramite i sistemi di visione artificiale al giorno d'oggi sono innumerevoli. Storicamente, stimare l'età di un soggetto a partire dall'immagine che lo ritrae è uno dei problemi più stimolanti nel campo della *facial analysis*. L'aspetto cruciale per questo task è il processo di invecchiamento che ogni individuo subisce e che cambia da soggetto a soggetto, un processo incontrollabile, che comporta una varianza importante per soggetti dello stesso range di età. Ancora, se parliamo di immagini catturate *in the wild*, possiamo avere immagini di soggetti che presentano caratteristiche somatiche che possono trarre in inganno i sistemi di visione, come occhiali, baffi, trucco e altro ancora.

L'obiettivo preposto in questo documento è l'utilizzo del dataset di facce più grande del mondo: **VGGFace2**[1]. Un altro aspetto fondamentale della *Age Estimation*, infatti, è la difficoltà nel reperire un dataset sufficientemente grande ed affidabile per poter essere utilizzato per questo scopo. La difficoltà è appunto nel recuperare una mole di dati sufficientemente grande e annotata in modo affidabile, avere dunque un dataset sufficientemente rappresentativo. La tematica trattata in questo documento è una feature fondamentale per la visione artificiale, utilizzata al giorno d'oggi in numerose applicazioni come la *robotica sociale*, la *privacy preservation* o l'*access control*. Per i nostri scopi, il suddetto dataset è stato annotato con le corrispondenti età in formato *float*. Questo ci permette di poter implementare un'architettura neurale sia come *regressore* che come *classificatore*. Verrà dunque valutato quanto proposto in termini di *pre-processing*, *data augmentation*, *architettura*, *training steps* e *funzione di costo*.

Il nostro contributo è stato, quindi, condurre uno studio sullo stato dell'arte delle reti presenti e sullo sviluppo di un'architettura neurale che, sia con tecnica di regressione o di classificazione, possa sfruttare nel

[1] GitHub Reference: https://github.com/ox-vgg/vgg_face2

miglior modo possibile il dataset a disposizione nel suo processo di apprendimento. È possibile avere dei riferimenti al codice di training, di testing e i vari script di pre-processing del dataset che sono stati realizzati al seguente link GitHub: `https://github.com/dev-guys-unisa/ArtificialVision_FinalContest2020`.

2 Descrizione della soluzione

In questa sezione si riportano gli aspetti fondamentali del nostro studio e dell'architettura realizzata, descrivendo la rete neurale e la procedura di allenamento svolta.

2.1 Convolutional Neural Network

Una delle prime decisioni prese per la realizzazione della rete neurale è stata la scelta tra l'implementazione di un'architettura come *regressore* o come *classificatore*. La scelta è ricaduta sulla realizzazione di un **classificatore**. Sono state dunque considerate **101 classi**[2] in totale dove ognuna di esse rappresenta un'età. Il perché di questa decisione può ricadere in più motivazioni. Innanzitutto, il task preposto per la realizzazione della rete prevede un output in forma intera, rappresentare dunque l'età del soggetto con un numero intero. Il classificatore associa la sua predizione ad una classe e produce in output un numero intero che corrisponde proprio alla sua predizione. Aver avuto un regressore, al contrario, avrebbe previsto la gestione ed approssimazione di un numero reale ottenuto in output per poter comunque avere un intero. Inoltre, è possibile pensare ad applicazione dove è necessario poter classificare un range di età e non dare in output la più probabile, come ad esempio un meccanismo di access control.

L'implementazione dell'architettura utilizzata è stata guidata dallo studio dello stato dell'arte per quanto riguarda la *face analysis*. Tra le reti preaddestrate presenti in letteratura, ed implementate tramite il framework Keras, quella più indicata per il nostro scopo si è rivelata **ResNet50**[3].

Il motivo principale che ci ha spinto verso questa scelta è che il dataset su cui questa rete è stata preaddestrata è proprio **VGGFace2**, che è il dataset a disposizione per implementare l'architettura e svolgere il task di *Age*

[2] Le classi considerate includono l'intervallo [0, 100] con estremi esclusi e cioè 101 classi totali.

[3] Implementazione presente su GitHub: https://github.com/WeidiXie/Keras-VGGFace2-ResNet50

Estimation. La backbone dell'architettura proposta è la **ResNet50**. Si tratta di una Residual Network utilizzata in molte reti neurali. L'efficienza sta nell'aumento delle performance rispetto ad una normale deep network e un processo di allenamento più semplice.

A questo punto possiamo mostrare l'**architettura** della rete completa.

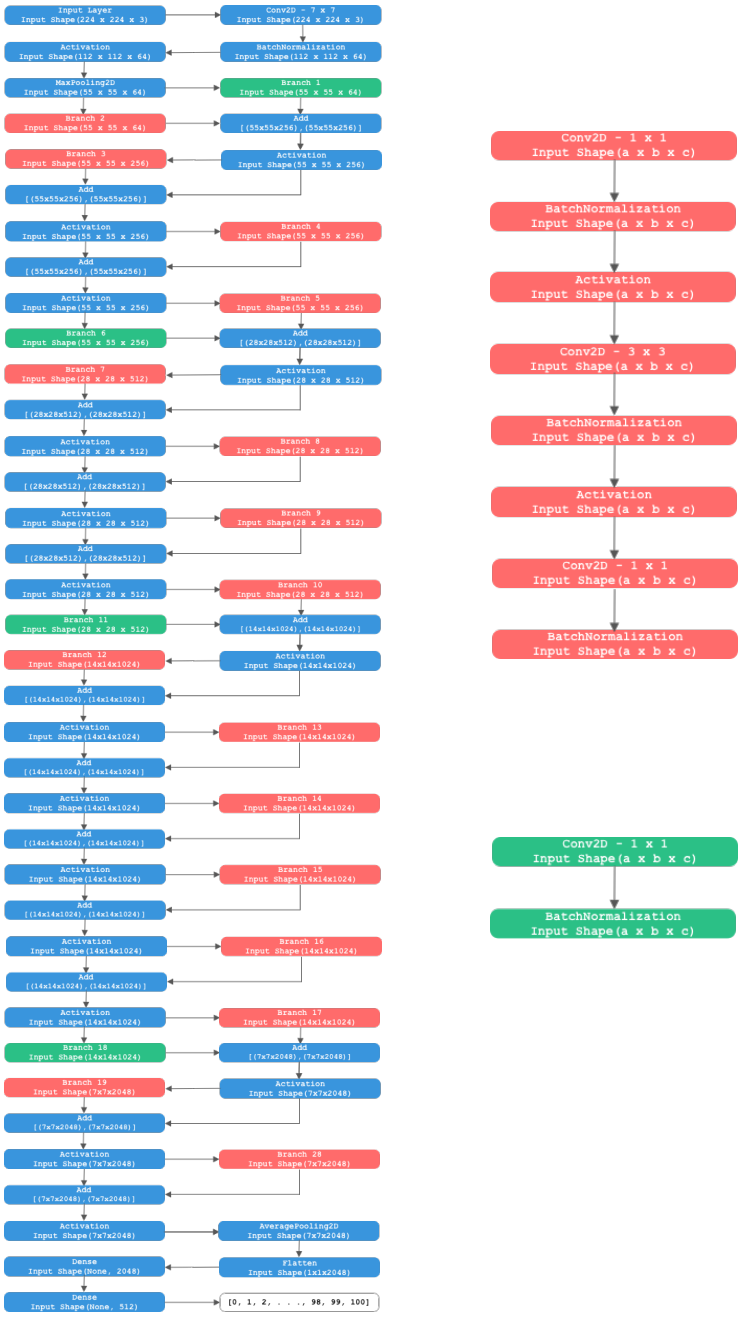


Figura 1 - Architettura della rete neurale proposta come soluzione

Ciò che si vede nel grafico rappresenta ogni livello della rete, considerando che i blocchi rossi e verdi denominati come *Branch* rappresentano le *sottoreti* specificate sul lato destro della figura. Sostanzialmente la rete fa un utilizzo estensivo di livelli **Convolutional2D** e di **BatchNormalization** creando più rami che vengono poi ricongiunti con un Layer **Add**. In tutti i livelli di attivazione, poi, si è utilizzata la funzione di attivazione **ReLU** (Rectified Linear Unit), impiegata in modo intensivo perché riduce il rischio di overfitting della rete. Ciò che notiamo in output della rete, infine, è un livello **Dense** con 101 unità neurali, dove ognuna di esse rappresenta una classe di età come descritto precedentemente. Quest'architettura prevede l'operazione di feature extraction delle immagini per essere processate ed utilizzate in fase di apprendimento e successivamente, sotto test, in fase di predizione.

2.2 Procedura di addestramento

Dataset

Il dataset originale di **VGGFace2** consiste in *3.31 milioni* di immagini suddivise in *9131 identità*; le immagini sono collezionate da Google Image Search e hanno una grande variazione in posa, età, illuminazione, etnia e professione. Tale dataset è già diviso in un training set, composto da *8631 identità*, e un test set che include le rimanenti *500 identità*.

Per i nostri scopi, abbiamo scelto un sottoinsieme del training set, composto da 1261462 elementi, vale a dire un massimo di *150 immagini per ogni identità*. Tale set è stato successivamente diviso in:

- **70%** per il training, ovvero 790488 campioni.
- **20%** per la validation, ovvero 344795 campioni.
- **10%** per il test, ovvero 126179 campioni.

La scelta del sottoinsieme del training set originale è stata effettuata nel seguente modo: innanzitutto è stato calcolato il range di età per ogni identità, considerando l'età massima e l'età minima delle immagini associate ad essa. Tale range è stato diviso in *4 fasce*, a seguito di diverse prove effettuate con un numero di fasce maggiore o minore, come si può vedere nei seguenti grafici:

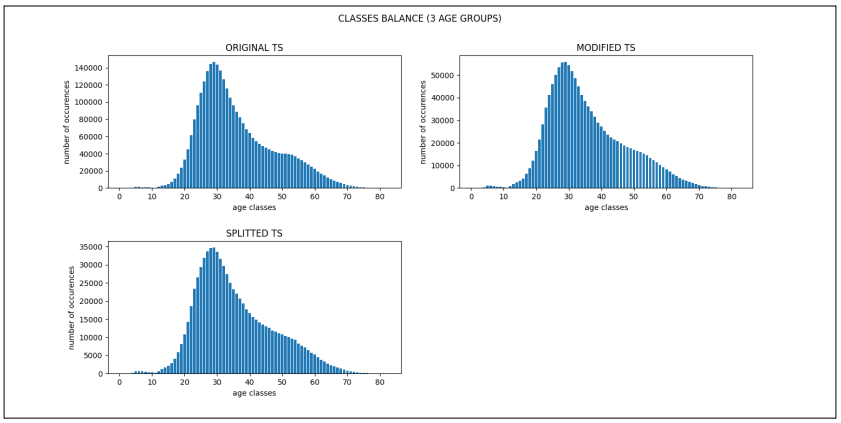


Grafico n.1 - Plot della distribuzione dei campioni per classe nel training set originale, nel sottoinsieme selezionato con divisione a 3 fasce e in quello finale ottenuto dallo split

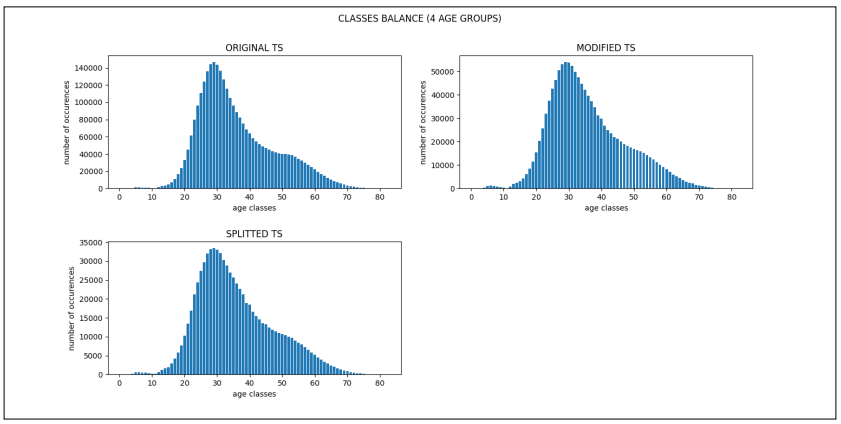


Grafico n.2 - Plot della distribuzione dei campioni per classe nel training set originale, nel sottoinsieme selezionato con divisione a 4 fasce e in quello finale ottenuto dallo split

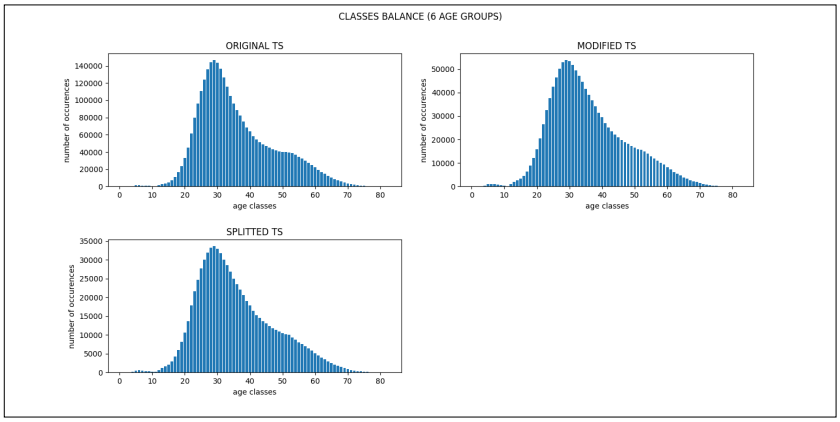


Grafico n.3 - Plot della distribuzione dei campioni per classe nel training set originale, nel sottoinsieme selezionato con divisione a 6 fasce e in quello finale ottenuto dallo split

Analizzando attentamente i grafici, la scelta è ricaduta su un numero di fasce pari a 4 in quanto, oltre a diminuire sensibilmente il numero di campioni rispetto al dataset originale, garantisce una *rappresentanza leggermente maggiore* per tutte le fasce d'età, senza però rinunciare alla distribuzione dei dati che si otteneva in partenza.

La scelta di usare tale metodo di selezione dell'insieme di campioni da usare è stata dettata dal voler mantenere tutte le identità del dataset originario in modo da catturare quanta più variabilità di features significative, dato che ogni identità ha un volto con determinate caratteristiche intrinseche, magari indipendenti dall'età.

Per ciascuna fascia, sono state selezionate *casualmente* un massimo di 30 immagini, tranne che per la terza da cui ne abbiamo selezionate 60. Il numero di immagini massimo per ogni gruppo di età è stato scelto considerando due fattori:

- *fattore tempo*, dovuto alle tempistiche di consegna del progetto e ai limiti di utilizzo della piattaforma Google Colab su cui abbiamo effettuato l'addestramento e il testing della DCNN.

- cercare di mantenere la *distribuzione dei campioni nel dataset originale*, ma allo stesso uniformare il più possibile il numero di campioni per classe, evitando spike di campioni di una determinata età.

Dal secondo fattore dipende in particolare la scelta di 60 campioni per la terza fascia di età in quanto, analizzando la distribuzione originale dei campioni, si può notare un numero maggiore di elementi compresi all'incirca nella fascia [25, 35], che in media corrisponde al secondo raggruppamento di età (considerando un range [0, 100]); quindi, prendendo meno campioni rispetto alla fascia successiva, tale picco tenderebbe ad attenuarsi. Naturalmente, avremmo ottenuto una maggiore attenuazione prendendo anche più campioni della prima e dell'ultima fascia; tuttavia ciò non sarebbe stato possibile per tutte le identità (specialmente quelle con range di età più stretto) ed, inoltre, al fine di non aumentare le dimensioni del dataset, ciò avrebbe significato prendere meno campioni nelle altre fasce e quindi alterare la distribuzione originale.

Per stimare le performance del modello, abbiamo utilizzato il **validation set approach** dando in pasto alla rete l'intero validation set (scelto come specificato precedentemente) in modo da avere una quantità importante di dati non visti durante il training su cui testare la capacità di generalizzazione della rete stessa, anche perché il dataset a disposizione permetteva di applicare una divisione in 3 sets senza rischiare di avere

pochi dati su cui effettuare il training, compromettendo quindi le prestazioni.

Infine, per rendere più pratico l'utilizzo di Google Colab col Drive e ridurre i tempi, gran parte della fase di elaborazione del dataset (scelta del sottoinsieme di campioni, divisione in training-validation-test e face detection) è stata svolta in locale e solo successivamente caricata su Google Drive usando il formato *TFRecord*. In particolare abbiamo utilizzato due diverse tipologie di record:

- per il *training set* ed il *validation set*, per ogni immagine, sono stati salvati il path relativo, le dimensioni, l'età associata ed una sua rappresentazione come stringa di bytes;

- per il *test set*, per ogni immagine, sono stati salvati solo il path relativo e la rappresentazione come stringa di bytes (*pre-processata*); questo perché le labels da usare nel test sono state salvate in un file *.csv* da confrontare con quello delle predizioni.

Face Detection

Il rilevamento del volto ha occupato un ruolo determinante nella realizzazione del modello da addestrare. Fin dalla fase di studio del dataset, abbiamo notato che alcune delle immagini presentavano un numero di volti maggiore ad uno, rendendo complicata la stima dell'età di un soggetto in foto. Quindi, fatta questa premessa, abbiamo deciso di implementare una soluzione che rilevasse i volti all'interno delle immagini, scegliendo quello con **area maggiore**, e che quindi ci indicasse il *soggetto in primo piano*. La soluzione adottata è ricaduta sulla scelta del detector **MTCNN**, ovvero un framework sviluppato come soluzione sia per il rilevamento che per l'allineamento dei volti. Questo processo consiste in *stages* di reti convoluzionali che sono in grado di riconoscere i volti e la posizione del punto di riferimento del volto stesso come occhi, naso e bocca.

Nella pratica, però, questo detector è stato solo implementato, in quanto si è utilizzato un approccio che prevede l'utilizzo di informazioni sui volti già estratti con il framework del **MiviaLab**[4]. Infatti, abbiamo elaborato le informazioni presenti all'interno dei file *.csv* del framework, andando ad estrapolare le coordinate dei riquadri di delimitazione dei volti e la posizione dell'intero viso nelle immagini. Avendo questi dati, è stato

[4] Link al framework presente su GitHub: `https://github.com/MiviaLab/GenderRecognitionFramework`

possibile eseguire un ritaglio sulle immagini del dataset, ottenendo nuove immagini che raffigurassero solo i volti su cui fare la stima dell'età. Questo approccio è stato utilizzato sia per lavorare sulle immagini che formano il dataset di train, sia per le immagini che compongono il test.

"This is a state-of-the-art deep learning model for face detection, described in the 2016 paper titled "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks."

Face Pre-Processing

La fase di pre-processamento delle immagini è stata fondamentale per la normalizzazione dei volti individuati durante la fase di rilevamento precedente. Come suggerito dagli autori di VGGFace2, la tecnica da applicare è quella di sottrarre i valori medi calcolati su tutto il dataset iniziale, da tutti i canali delle immagini del training set. Da sottolineare che, come implementazione, abbiamo usato il tool *mean_std_normalize*" messo a disposizione all'interno del framework del MiviaLab, nella sezione *dataset_tools*.

Data Augmentation

Per quanto riguarda la data augmentation, è stata utilizzata una tecnica che aumentasse la quantità di dati aggiungendo copie delle immagini modificate. Tale tecnica ha visto l'implementazione delle seguenti trasformazioni:

- **random_brightness_contrast**: applicazione di una certa intensità di contrasto e luminosità all'immagine; questo è utile perché differenti condizioni di illuminazione possono alterare notevolmente la percezione dell'età di una persona, infatti un volto maggiormente illuminato può apparire più giovane di quanto sia nella realtà. Per quanto riguarda il contrasto, una variazione potrebbe mettere in risalto o meno, i *key points* del volto e consentire un migliore apprendimento e riconoscimento di essi.

- **random_monochrome**: alterazione dell'immagine secondo una scala di grigi; utile per esaltare determinate caratteristiche dei volti che altrimenti non verrebbero valorizzati nell'immagine a colori. Inoltre, permette di ridurre l'overfitting sulle immagini a colori.

- **random_flip**: capovolgimento dell'immagine; utile per osservare la stessa immagine da diversi punti di vista.

Di seguito è possibile vedere il risultato delle trasformazioni descritte finora ed applicate su un'immagine di test. È possibile vedere la differenza con l'immagine originale, considerando soprattutto la resize applicata, dovuta all'**input size** della rete utilizzata, cioè **224 x 224 x 3**. Si è dunque scelto di mantenere questa dimensione per non eccedere con i tempi di training nel caso di size più grandi o per non avere cali di accuracy nel caso di size più piccole.

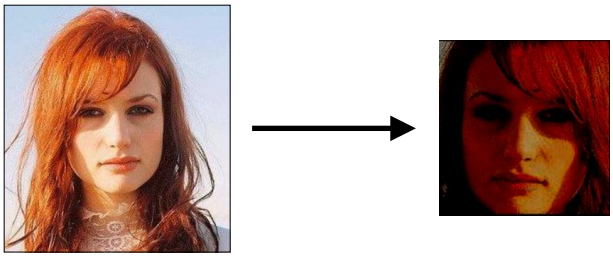


Figura 2 - Processo di trasformazione dell'immagine

Per le ultime due trasformazioni, la probabilità di eseguire la loro applicazione viene calcolata a partire dalla generazione di un numero casuale. Anche in questo caso, abbiamo fatto riferimento alle funzioni di data augmentation presenti all'interno del framework. Avendo a disposizione la loro implementazione, ci siamo concentrati nel creare una logica che applicasse queste trasformazioni in modo da generare una determinata quantità di dati.

La logica è stata la seguente: per ogni immagine è stato deciso, innanzitutto, se mettere in pratica l'augmentation, attraverso la generazione di un *numero casuale tra 0 o 1*. In caso affermativo, l'applicazione della augmentation ha previsto l'esecuzione, in successione, della *random_brightness_contrast*, *random_monochrome* e della *random_flip*. Da mettere in evidenza, come potenzialmente tutte e tre le tecniche possono essere applicate su un'immagine, ma, di fatto, la presenza della probabilità rende meno attuabile questo scenario, anzi diversifica le possibilità di combinazione sull'aumento dei dati. Infine, come conseguenza, oltre all'aumento di dati, questo ci ha permesso di ridurre l'overfitting sul modello addestrato.

Training from scratch o fine tuning

L'addestramento del modello per stimare l'età è partito dall'utilizzo di una rete pre-inizializzata. Nel nostro caso, i pesi sono stati inizializzati

attraverso **ImageNet**[5] e la motivazione è stata dettata dal fatto che, partire da modelli pre-addestrati, ci ha garantito sia la partenza con i migliori pesi esistenti e, sia di risparmiare tempo. In generale, una volta che la rete è stata addestrata, il tempo richiesto per la classificazione di un nuovo pattern - *forward propagation* - è in genere veloce.

Inoltre, va evidenziata l'applicazione di un ultimo livello di output sulla rete pre-addestrata; questo, perché bisogna garantire che le uscite del modello siano adeguate al numero di classi relativo al problema: nel nostro caso, 101 classi. Il layer di output definito è stato il **Dense**, utilizzando come funzione di attivazione la **Softmax**[6], che ci ha permesso di gestire un vettore di uscita normalizzato di 101 elementi, dove ogni elemento può valere da 0 a 1 e la somma di tutti gli elementi è pari a 1.

Procedura di training

La procedura di training ha richiesto, prima di essere effettuata, uno studio in cui sono stati regolati alcuni dei parametri fondamentali per iniziare l'addestramento del modello.

- **n_epochs** = 25
- **batch_size** = 128
- **initial_learning_rate** = 0.005
- **learning_rate_decay_factor** = 0.2
- **learning_rate_decay_epochs** = 20
- **momentum** = 0.9
- **patience** = 5
- **num_classes** = 101
- **weight_decay** = 1e-4

---

[5] ImageNet è un database di immagini realizzato per l'utilizzo nel campo della visione artificiale.

[6] Funzione di attivazione che permette di gestire le probabilità delle classi ed estrarre la più probabile.

Inoltre questi parametri sono stati tarati in modo tale da avere l'output della rete il più possibile vicino a quello desiderato. Tra i vari parametri presentati, si distingue il *weight_decay*, un indice di regolarizzazione già implementato nella rete preaddestrata, che permette di evitare l'overfitting del modello. La funzione di *loss* selezionata è stata la *categorical_crossentropy*, a cui abbiamo affidato il compito di calcolare la perdita di entropia tra le etichette (età) vere e quelle predette. In più, l'addestramento è stato controllato attraverso l'uso di due metriche: **categorical_accuracy** e **mae.** Della funzione *mae* è stata realizzata una opportuna implementazione, e la scelta è motivata dal fatto che tale metrica sarà la base per la valutazione dei risultati del contest. Da sottolineare ancora una volta che, tutti questi aspetti, sono stati valutati anche tenendo conto di quanto fosse già stato implementato all'interno del framework del MiviaLab.

La procedura di addestramento è stata realizzata attraverso la piattaforma cloud **Google Colab**, ed è stata caratterizzata da un numero di epoche pari a 25, che è possibile esaminare in due fasi:

- le prime **18** epoche, con un addestramento svolto sugli ultimi 11 livelli del modello:
    - **Total params**: 24.662.053
    - **Trainable params**: 4,513,893
    - **Non-trainable params**: 20,148,160
- le ultime **7** epoche, con un addestramento svolto su tutti i livelli che compongono il modello:
    - **Total params**: 24.662.053
    - **Trainable params**: 24,608,933
    - **Non-trainable params**: 53,120

Altri dettagli riguardanti la procedura di training, interessano l'uso delle funzioni di **EarlyStopping** e del **ModelCheckpoint**, presenti all'interno della libreria Keras. La prima, ha permesso di fermare l'addestramento non appena la metrica monitorata smetteva di migliorare. La seconda, è stata utile per salvare i pesi del modello con una certa frequenza, andando a salvaguardare il lavoro fatto nel caso Google Colab andasse a limitare o rimuovere la potenza di calcolo.

3 Risultati sperimentali

Di seguito sono riportati i risultati ottenuti con le diverse reti, sia in fase di addestramento che di testing, relativi all'ultima epoca di addestramento. A titolo informativo vengono riportate le metriche utilizzate per valutare le reti neurali addestrate.

Per la fase di addestramento sono state valutate 2 tipi di data augmentation; oltre quella descritta in precedenza, è stata eseguita anche la seguente: dopo aver deciso randomicamente se effettuare o meno l'augmentation, viene scelta una delle 3 di cui sopra con la differenza che flip e monochrome vengono applicate in modo deterministico e non probabilistico.

Loss Function

La funzione che riporta il *costo* come un numero reale. Un problema di ottimizzazione cerca di minimizzare questa funzione. Il nostro è un problema *multi-classe* e, come nella maggior parte di problemi come questo la funzione utilizzata maggiormente è la *categorical crossentropy*. I nostri sperimenti hanno visto l'utilizzo di questa metrica per due categorie di dati:

- *Training Loss*: calcolo del costo sul training set.
- *Validation Loss*: calcolo del costo sul validation set.

Accuracy

La metrica principale di valutazione di una rete convoluzionale. In genere è riportata in percentuale ed attesta la precisione con la quale la rete riesce a predire la label corretta per un campione. Nel nostro caso è stata utilizzata la *categorical accuracy*. I nostri esperimenti anche in questo caso hanno visto l'utilizzo di questa metrica per:

- *Training Categorical Accuracy*: calcolo dell'accuracy sul training set.
- *Validation Categorical Accuracy*: calcolo dell'accuracy sul validation set.

Mae

La metrica utilizzata in letteratura scientifica per i task di *face analysis*. Questo genere di problemi prevede la ricerca della minimizzazione dell'errore medio assoluto. Ad esempio, avere un *Mae* di 1.2 su uno specifico modello indica che i suoi valori predetti avranno un errore

medio di ± 1.2. Nel nostro caso tutti i set utilizzati e descritti nelle precedenti sezioni sono stati coinvolti nel calcolo di questa metrica.

Risultati: di seguito una tabella riassuntiva dei modelli addestrati e dei test effettuati.

NET	VGG16	MOBILENET	SENET	C3AE	XCEPTION	RESNET50
T-LOSS	3,223	2,451	2,845	3,063	2,280	2,282
T-ACC	0,072	0,159	0,109	0,093	0,188	0,244
T-MAE	7,452	2,095	3,213	4,148	1,692	1,184
V-LOSS	3,281	2,817	3,282	3,303	2,436	2,650
V-ACC	0,065	0,124	0,077	0,069	0,164	0,184
V-MAE	8,731	2,867	4,265	4,497	1,853	1,872
TEST-MAE	9,373	5,654	6,286	7,733	4,037	3,940

• VGG16 - 7° Epoca

Scartata - per la rete in questione, è stata applicata l'augmentation descritta in questa sezione; si è proceduto all'addestramento degli ultimi 5 livelli per un totale di 7 epoche. Effettuando il test sul test set ricavato dal training set originale di VGGFace2, si è ottenuto un valore del **mae** alto e ciò ha portato a virare su un' altra rete.

• MOBILENET 224 - 9° Epoca

Scartata - dato il basso numero di pesi della rete in questione, si è proceduto con l'addestramento dell'intera rete per 9 epoche; è stato stoppato in quanto iniziava a dare segni di overfitting.

• SENET - 16° Epoca

Scartata - il training effettuato è stato diviso in due fasi: nella prima sono stati addestrati solo gli ultimi 15 livelli per 9 epoche, successivamente si è proceduto ad addestrare l'intera rete per le restanti 7 epoche. È stata scartata in quanto cominciava ad overfittare i dati e non forniva buoni risultati eseguendo il test.

• C3AE - 9° Epoca

Scartata - anche per questa rete è stato utilizzato l'augmentation descritto in questa sezione; l'addestramento è stato effettuato sull'intera rete ma, dopo 9 epoche, è stato stoppato a causa del valore troppo alto ottenuto in fase di test e dell'overfitting che iniziava a mostrare continuando con le epoche.

• XCEPTION - 5° Epoca

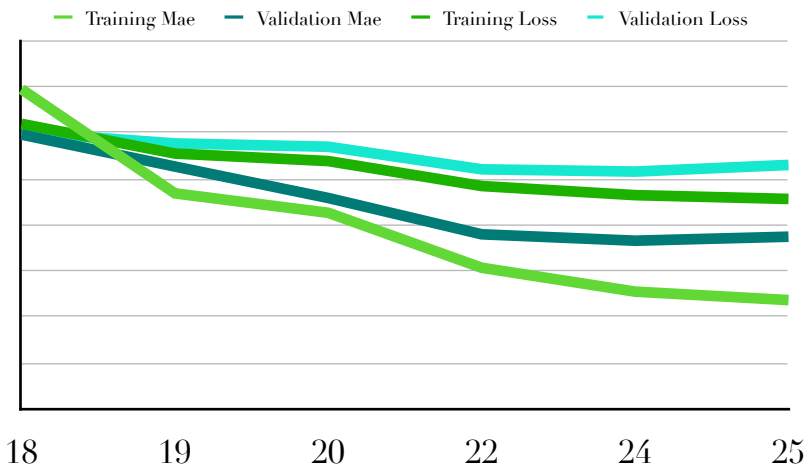
Scartata - si è proceduto con l'addestramento di tutti i livelli della rete per un totale di 5 epoche; è stato poi stoppato a causa dei tempi troppo lunghi nel portare a termine tali addestramenti.

• RESNET50 - 25° Epoca

La scelta del modello da utilizzare per la stima dell'età è ricaduta sulla ResNet50, considerando i risultati ottenuti sia durante la fase di training e sia durante la fase di test. In particolare, andando ad analizzare i dati ottenuti sulle ultime epoche dell'addestramento, è emerso un netto miglioramento dei valori.

	loss	cat_accuracy	mae	val_loss	val_cat_accuracy	val_mae
Epoca 18	3,0966	0,1067	3,4728	3,0052	0,1132	2,9803
Epoca 19	2,7744	0,1431	2,3423	2,8844	0,1288	2,6335
Epoca 20	2,6928	0,1559	2,1311	2,8484	0,1305	2,2919
Epoca 22	2,4219	0,2103	1,5348	2,6034	0,1791	1,8979
Epoca 24	2,3229	0,2330	1,2756	2,5787	0,1857	1,8284
Epoca 25	2,2819	0,2439	1,1839	2,6506	0,1845	1,8719

La scelta di fermare l'addestramento alla venticinquesima epoca è dovuto, invece, alla possibilità di sovra adattamento del modello, determinato dall'eccessivo training. Come possiamo vedere dalla figura seguente, il parametro *mae* si attesta su un valore costante verso le ultime due epoche, suggerendo un inizio di calo delle prestazioni.



4 Conclusioni

Giunti alla conclusione di questo lavoro possiamo analizzare quanto riportato nelle precedenti sezioni.

In primis, abbiamo compreso l'importanza del task della *Age Estimation* in particolare modo da un punto di vista applicativo. È fondamentale poter realizzare architetture che possano essere eseguite anche su dispositivi non tecnologicamente avanzati e che possano essere in egual modo efficienti. In letteratura questo ambito è molto trattato ma gli algoritmi utilizzati richiedono un utilizzo estensivo di operazioni computazionali.

In secondo luogo, è stata trattata la nostra proposta di soluzione per il task. Innanzitutto si è studiato il dataset utilizzato e si sono poi applicate le dovute operazioni di riduzioni coerentemente con le risorse a disposizione per sviluppare la nostra soluzione (i.e. Google Colab). Si è quindi analizzata la distribuzione dei dati per ogni identità presente nel dataset, ridotto per le motivazioni di cui sopra.

Abbiamo, infine, proposto un modello per il task che ci è stato posto cercando di raggiungere performance accettabili e competitive, confrontando i vari modelli addestrati e, in particolar modo, quello finale con le performance che si trovano in letteratura.