Contest di Visione Artificiale: Gruppo 27

Valitutto Andrea, Gargiulo Michele, Sabini Pietro, Marchesano Riccardo
{a.valitutto9, m.gargiulo30, p.sabini, r.marchesano3}@studenti.unisa.it

1 – Introduzione

L'obiettivo preposto in questo progetto è l'utilizzo del Dataset di volti umani più grande che esista: VGGFace2. Tale dataset è stato proposto al fine di stimare l'età delle persone presenti in esso. Il suddetto Dataset è stato annotato con le corrispondenti età in formato *float.* Questo ha permesso di impostare il problema dell'Age Estimation sia come una regressione e sia come una classificazione. Nel caso in esame, è stato considerato questo problema come un problema di **classificazione**.

L'età verrà classificata come un numero intero da 0 a 100, con un totale di 101 classi. Il nostro contributo è stato quindi quello di condurre uno studio sullo stato dell'arte delle reti presenti e sullo sviluppo di un'architettura neurale che, sia con tecnica di regressione sia di classificazione, possa sfruttare nel miglior modo possibile il dataset a disposizione nel suo processo di apprendimento. È possibile trovare il codice per dividere il dataset, addestrare e testare il modello finale al seguente URL: *https://github.com/andrewvali/ContestArtificialVision*.

2 – Descrizione della soluzione

Nella seguente sezione si riportano gli aspetti fondamentali dello studio effettuato e dell'architettura realizzata, descrivendo la rete neurale e la procedura di allenamento svolta.

2.1 – Convolutional Neural Network

Il compito proposto prevede un output in forma intera che rappresenti l'età del soggetto in questione. Il classificatore associa, infatti, la sua predizione ad una classe e produce in output un numero intero che corrisponde proprio alla sua predizione. Un regressore, invece, avrebbe previsto la gestione e l'approssimazione di un numero reale ottenuto in output per poter avere un intero. Questo è stato uno dei motivi che ha portato a considerare il nostro problema come classificazione. La rete utilizzata è una ResNet-50 utilizzando i pesi di VGGFace2, una delle reti preaddestrate presenti in letteratura e supportate nella libreria VGGFace. Alla rete base, sono stati aggiunti 6 ulteriori livelli.

In particolare, sono stati aggiunti i seguenti livelli:

VGGFace implementata con Resnet50	Descrizione	Numero di parametri
Flatten	Necessaria per riportare il tensore finora processato in monodimensionale. Le features estratte da ResNet-50 sono 2048	0
Dense (2048, relu)	Primo livello di Dense, con attivazione Relu, avente la stessa dimensione della Flatten	4196352
Dropout (0.5)	Layer di Dropout, usato per ridurre l'overfitting sul modello, con probabilità del 50%	0
Dense (512, relu)	Secondo livello di Dense, con dimensione 512 e attivazione Relu	1049088

Dropout (0.5)	Layer di Dropout, usato per ridurre l'overfitting sul modello, con probabilità del 50%	0
Dense (101, Softmax)	Ultimo livello di Dense, con dimensione 101 pari al numero delle classi e attivazione softmax, in modo che l'output rappresenti una Probability Mass Function. L'output finale sarà un vettore composto dalla probabilità per ogni classe.	51813

Da come si può vedere nel seguente summary, aggiungendo 6 livelli alla rete base, si ha un totale di circa 28 milioni di parametri da allenare.

La rete è stata progettata per essere un classificatore, utilizzando una softmax come ultimo layer di attivazione.

```
____________________________________________________________________________________________________
activation_45 (Activation)      (None, 7, 7, 2048)   0           add_14[0][0]
____________________________________________________________________________________________________
conv5_3_1x1_reduce (Conv2D)     (None, 7, 7, 512)    1048576     activation_45[0][0]
____________________________________________________________________________________________________
conv5_3_1x1_reduce/bn (BatchNor (None, 7, 7, 512)    2048        conv5_3_1x1_reduce[0][0]
____________________________________________________________________________________________________
activation_46 (Activation)      (None, 7, 7, 512)    0           conv5_3_1x1_reduce/bn[0][0]
____________________________________________________________________________________________________
conv5_3_3x3 (Conv2D)            (None, 7, 7, 512)    2359296     activation_46[0][0]
____________________________________________________________________________________________________
conv5_3_3x3/bn (BatchNormalizat (None, 7, 7, 512)    2048        conv5_3_3x3[0][0]
____________________________________________________________________________________________________
activation_47 (Activation)      (None, 7, 7, 512)    0           conv5_3_3x3/bn[0][0]
____________________________________________________________________________________________________
conv5_3_1x1_increase (Conv2D)   (None, 7, 7, 2048)   1048576     activation_47[0][0]
____________________________________________________________________________________________________
conv5_3_1x1_increase/bn (BatchN (None, 7, 7, 2048)   8192        conv5_3_1x1_increase[0][0]
____________________________________________________________________________________________________
add_15 (Add)                    (None, 7, 7, 2048)   0           conv5_3_1x1_increase/bn[0][0]
                                                                 activation_45[0][0]
____________________________________________________________________________________________________
activation_48 (Activation)      (None, 7, 7, 2048)   0           add_15[0][0]
____________________________________________________________________________________________________
avg_pool (AveragePooling2D)     (None, 1, 1, 2048)   0           activation_48[0][0]
____________________________________________________________________________________________________
flatten (Flatten)               (None, 2048)         0           avg_pool[0][0]
____________________________________________________________________________________________________
dense (Dense)                   (None, 2048)         4196352     flatten[0][0]
____________________________________________________________________________________________________
dropout (Dropout)               (None, 2048)         0           dense[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                 (None, 512)          1049088     dropout[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 512)          0           dense_1[0][0]
____________________________________________________________________________________________________
Logits (Dense)                  (None, 101)          51813       dropout_1[0][0]
====================================================================================================
Total params: 28,858,405
Trainable params: 28,805,285
Non-trainable params: 53,120
```

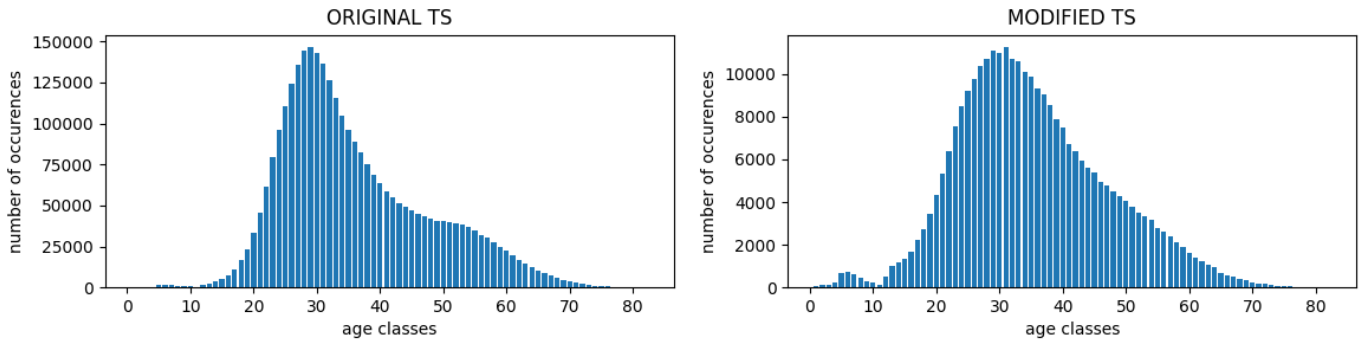
2.2 – Procedura di addestramento

2.2.1 – Dataset

Il dataset originale di VGGFace2 consiste in 3.31 milioni di immagini suddivise in 9131 identità. Tale dataset è già diviso in un training set, composto da 8631 identità, e un test set che include le rimanenti 500 identità. A causa dei tempi e delle risorse ridotte è stato deciso di utilizzare solo un sottoinsieme del dataset fornito. In particolare, è stata calcolata la distribuzione del numero di immagini per ogni età di ogni identità. Ci siamo assicurati che i sottoinsiemi del dataset utilizzato come training e validation seguissero la stessa distribuzione, e che tutte le età venissero considerate. Si può notare che prendendo un solo sample, per ogni età di ogni identità, si ottengono circa 250.000 samples. Per il training, questa rappresenta la soglia minima.

Il numero di samples presi dal dataset originale è di **319.998,** equivale a 38 immagini per ogni identità. La tecnica di divisione per ogni età di ogni identità è stata la seguente:

- Per ogni identità è stato calcolato il range di età (età massima – età minima) ed è stato diviso in 4 gruppi. 38 immagini per identità divise in 4 gruppi vale a dire 9 immagini (divisione intera) per gruppi di età.
- Le restanti 2 immagini sono state prese, in modo sperimentale, dal gruppo che avesse portato ad una distribuzione simile al dataset originale. Come Banalmente si può vedere dal grafico, la fascia di età considerata in questo caso è la terza in quanto presenta un numero molto maggiore rispetto alle restanti fasce. Tuttavia, per avere un maggiore equilibrio andavano prese anche quelle della prima e l'ultima fascia, ma così facendo non sarebbero state più disponibili per il test set.
- A questo punto è stato preso l'80% (**255.998)** del nuovo dataset per effettuare il training della rete e il restante 20% (**63.999**) per il validation, seguendo lo stesso principio di divisione effettuato prima, in modo tale da avere la stessa distribuzione sul training e validation set.
- La parte di test per valutare la nostra rete in termini di **MAE** (*Mean absolute error*) è del 10% della dimensione del dataset creato (31999 samples).



Per stimare le performance del modello, abbiamo utilizzato il validation set approach, utilizzando un validation set (scelto come specificato sopra) in modo da avere una quantità importante di dati non visti durante il training su cui testare la capacità di generalizzazione della rete stessa.

2.2.2 – Face Detection

Il rilevamento del volto ha occupato un ruolo determinante nella realizzazione del modello da addestrare. Infatti, durante l'analisi dei dati sul dataset originale, sono state individuate una grossa quantità di immagini contenente più di un soggetto, di conseguenza rendendo molto complicata la stima dell'età del

soggetto d'interesse. Quindi, fatta questa premessa, la soluzione sarebbe stata l'implementazione di un detector che estraesse soltanto il volto su cui effettuare la stima dell'età. La soluzione a tale problema è stata quella di utilizzare i risultati delle crop già effettuate da un detector, reso disponibile dal MiviaLab, disponibile al seguente link (https://github.com/MiviaLab/GenderRecognitionFramework). Infatti, sono state utilizzate le coordinate all'interno dei file .csv del framework, estrapolando i volti dei soggetti all'interno delle immagini su cui effettuare la stima dell'età. Questo approccio è stato utilizzato sia per lavorare sul training, validation e test.

2.2.3 – Face pre-processing

La fase di pre-processing delle immagini è stata fondamentale per la normalizzazione dei volti. È stato seguito lo stesso approccio suggerito dagli autori di VGGFace2, la tecnica applicata è stata quella di sottrarre, per ogni immagine, la media dei 3 canali di colore. Anche in questo caso, è stata utilizzata la funzione "mean_std_normalize" messa a disposizione all'interno del framework del MiviaLab, nella sezione dataset_tools.

2.2.4 – Data Augmentation

Come Data Augmentation è stata specificata la modalità "vggface2" che effettua variazioni causali in termini di flip, luminosità, contrasto e la conversione in grayscale.

- **random_brightness_contrast**: applicazione di una certa intensità di contrasto all'immagine.
- **random_monochrome**: alterazione dell'immagine secondo una scala di grigi.
- **random_flip**: capovolgimento dell'immagine.

Anche in questo caso, abbiamo fatto riferimento alle funzioni di data augmentation presenti all'interno del framework. Altri tipi di augumentation sono stati testati, ma i risultati sono stati peggiori o migliori di una quantità trascurabile, ma aumentando i tempi di addestramento. Tutto ciò ha portato a scegliere una data augumentation che offrisse un buon trade-off tra performance e tempi di addestramento.

2.2.5 – Training from scratch o Fine Tuning

È stata utilizzata una tecnica mista tra Fine Tuning e Training from Scratch.
Il training è stato effettuato in due fasi:

1) Warm-Up: inizialmente è stato fatto un fine tuning sugli ultimi livelli della rete per 30 epoche per capire i tempi di addestramento e le performance ottenute soltanto sugli ultimi livelli della rete.
2) Ai modelli più promettenti sono stati sbloccati tutti i livelli e sono stati addestrati partendo dai pesi trovati precedentemente.

2.2.6 – Procedura di training

La prima fase è stata quella di effettuare il training per 30 epoche soltanto sugli ultimi livelli aggiunti (circa 5 milioni di parametri), per calcolare bene i tempi a disposizione. Questa prima fase è stata cruciale per capire i modelli più promettenti su cui continuare il training, sbloccando tutti i livelli e facendolo ripartire dagli ultimi pesi salvati (i migliori sul validation). Dopo questa fase, sono stati analizzati i vari modelli costruiti e sono stati presi in considerazione soltanto quelli con l'errore e il **MAE** più basso sul validation. In particolare, su ResNet-50 con un totale di 28 milioni di parametri. I parametri di addestramento della rete sono stati i seguenti:

Funzione di Loss: è stata considerata la ***Ordinal Categorical Crossentropy Loss Function***. Tale funzione massimizza l'errore quando la distanza è maggiore tra la classe predetta e quella vera. L'output della rete proviene da una Activation Layer con Softmax, che restituisce un vettore *one-hot encoded* di 101 elementi. Ogni indice del vettore contiene la probabilità di appartenere alla corrispondente classe (esempio: [0.01,0.13,0.03,0.5,0.8,…..]) Con questa considerazione, ha senso considerare un errore tanto maggiore

quanto più diversi sono il vettore stimato e quello vero. Ulteriori informazioni sulla funzione di errore sono disponibili al seguente URL: *https://github.com/JHart96/keras_ordinal_categorical_crossentropy*

Metrica: Custom *MAE* adattata al nostro caso. Non è stata utilizzata la funzione di *mean absolute error* messa a disposizione da Keras, ma è stata modificata in modo tale da calcolare il MAE in base alla distanza tra la classe predetta e quella vera. Utilizzare quella di keras, non era coerente con il nostro problema poiché veniva effettuata la differenza dei due vettori, così facendo non sarebbe stato considerato di quanto si sbagliasse la classe.

Optimizer: SGD con momento 0.9. È stato scelto in base allo studio effettuato in letteratura sul task in questione.

Callback lists:

- **Riduzione del learning rate:** è stata utilizzata la funzione messa a disposizione da keras "*ReduceLROnPlateau*" con monitoraggio sul 'val_loss', factor=0.2, patience=5 e min_lr=0.001. In quanto è stato notato che intorno alla 15esima epoca il MAE sul validation oscillava e diminuendo il lr, si sono ottenuti ulteriori miglioramenti, passando da 2.74 a 2.54.
- **TensorBoard:** Utilizzata per plottare a posteriori i grafici sul traning e sul validation.
- **ModelCheckpoint:** Utilizzata per poter salvare, quando necessario, il modello migliore in base alla metrica di valutazione scelta.
- **EarlyStopping:** Molto utile in questo contesto, in quanto i tempi di addestramento, soprattutto su tutti i livelli della rete, erano di circa 50 min ad epoca. È stato utilizzato un delta di 0.002 e un patience di 15, per garatire che il training si stoppasse dopo 15 epoche di non miglioramento sul validation. È stato deciso di mettere 15 epoche perché dopo 5 epoche veniva diminuito il lr così da fermare l'addestramento se dopo la seconda diminuzione del lr non ci fossero stati miglioramenti.

3.0 – Risultati sperimentali

È stata utilizzata una pratica di *warm-up*, ovvero sono stati considerati vari modelli, allenandoli e valutando le performance dopo 30 epoche di addestramento. Ciò ci ha permesso di capire quali fossero i modelli più promettenti e poterli addestrare su tutti i parametri per 50 epoche. I parametri migliori di tutti i modelli sono stati poi salvati tramite una metrica MAE (Mean Absolute Error) personalizzata, per tenere conto del formato di output (one-hot encoded vector di lunghezza pari a 101).

Le reti prese in considerazione sono state le seguenti:

- Vgg16 con pesi di Imagenet
- MobileNet con i pesi di Imagenet
- Senet con i pesi di Imagenet
- ResNet50 con i pesi di VggFace2

Ciascuna rete è stata testata con parametri differenti di augmentation e di top-layer. In particolare, è stato osservato come molte delle reti prese in considerazione andassero in overfitting dopo circa 15 epoche in assenza dei 2 layer di Dropout in coda alla rete. Per questo motivo, dopo alcuni test, sono state scelte 3 layer di Dense con attivazione "relu", intervallati da layer di Dropout.

I risultati su Training e Validation Set per i modelli considerati sono stati i seguenti:

Modello	Pesi	Training MAE	Validation MAE
Vgg16	Imagenet	1.9	4.5
MobileNet	Imagenet	2.3	4.8
Senet	imagenet	2.2	3.7
ResNet50	VggFace2	1.30	2.51

4.0 – 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 particolar modo da un punto di vista applicativo. È stato affrontato un problema molto discusso negli ultimi tempi, in letteratura questo ambito è molto trattato, ma gli algoritmi utilizzati richiedono una potenza di calcolo incompatibile con gli strumenti utilizzati in questo progetto. Nonostante ciò, è stata implementata una rete di age estimation con delle performance buone sulle classi più comuni nel dataset originali e discrete sulle altre classi. Considerando che attualmente in letteratura le performance su questo task non sono elevatissime, confrontando i vari modelli sono comunque risultati accettabili.