Risoluzione dei task di age e gender detection usando tecniche di deep learning

Pier Vincenzo De Lellis PIE.DELELLIS@STUD.UNIROMA3.IT
Dipartimento di Ingegneria Informatica
Università Roma Tre

Francesco Foresi FRA.FORESI@STUD.UNIROMA3.IT
Dipartimento di Ingegneria Informatica
Università Roma Tre

Progetto: Machine Learning (sett. 2021), Laurea Magistrale Ingegneria Informatica Roma Tre

Abstract

I task di *gender detection* e di *age detection* da immagini contenenti volti di esseri umani, ha ormai assunto un ruolo cruciale sia nell'ambito della ricerca che nello sviluppo industriale, in particolare nell'ambito delle piattaforme social e dei social media.
Tuttavia i risultati ottenuti dai metodi dell'attuale stato dell'arte presentano ancora notevoli carenze in termini di prestazioni ed efficienza, causate principalmente dalla difficoltà intrinseca del task in questione. In questa relazione è presentata una soluzione per questi task semplice ma efficace, nella quale si propone un'architettura CNN e si riportano i risultati ottenuti durante la fase di sperimentazione.

Keywords: gender detection, age detection, deep learning, convolutional neural networks

1. Introduzione

Nell'ultimo ventennio le tecniche di Machine Learning hanno preso piede nella risoluzione di molti task che precedentemente venivano risolti manualmente o tramite algoritmi verbosi e spesso fallaci poiché richiedevano continui aggiornamenti. In questo progetto è stato scelto di risolvere uno dei task sociali più gettonati nella vita di tutti i giorni, ovvero quello di identificare il genere e l'età di un individuo attraverso delle immagini contenti volti. L'età e il genere hanno un ruolo fondamentale nelle interazioni sociali, infatti il linguaggio utilizzato da una persona spesso ci aiuta ad identificare il genere ma sopratutto l'età di quest'ultima. La vera sfida infatti risiede nella risoluzione di questo task tramite l'utilizzo di un dataset contenente immagini di volti. In questa relazione sono presentati i due modelli differenti per la risoluzione dei task sopra elencati, i risultati ottenuti rispettivamente da entrambi i modelli e infine il loro comportamento in modalità di inferenza. Abbiamo scelto di utilizzare due modelli distinti poichè abbiamo notato come un unico modello non riuscisse a riportare buoni risultati per entrambi i task.

2. Dataset e preprocessamento

Il dataset utilizzato per la sperimentazione della rete neurale è una partizione di uno dei più grandi dataset di immagini contenente volti di attori famosi, prelevate dai famosi domini *IMDB.com* e *wikipedia.com* (3).
L'intero dataset contiene più di 500 mila immagini, mentre per questa sperimentazione è stato scelto utilizzare due partizioni minori in base al task da risolvere.
Si è notato infatti che dopo una prima scelta di considerare unicamente un sottoinsieme di immagini, derivate dal solo dominio *wikipedia.com*, in fase di valutazione i modelli si sono comportati in maniera differente.
Infatti, mentre il modello per gender detection ha risposto bene a questa scelta iniziale, il modello per age detection ha fornito, invece, prestazioni fortemente negative, a causa della poca distribuzione dei punti nel dataset. Si è optato dunque, in una seconda fase, di utilizzare una seconda partizione(che considera anche immagini derivate dal dominio *IMDB.com*) del dataset per addestrare la rete neurale specializzata nel task di age detection.
Inoltre, la scelta di utilizzare due partizioni ci ha permesso di testare la bontà dei nostri modelli su dataset con immagini e dimensioni diverse.
La distribuzioni dell'età sul dataset complessivo, rispetto alle diverse partizioni, è mostrato in figura seguente

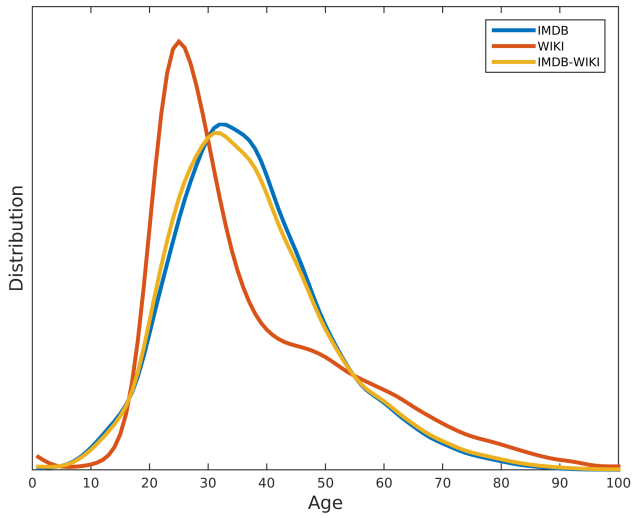


Figure 1: distribuzione dell'età rispetto ai dataset

La prima partizione contiene 42,255 immagini, mentre la seconda partizione contiene 100,122 immagini. Una volta trovato il dataset da utilizzare è seguita una fase di preprocessamento e preparazione delle immagini da utilizzare per addestrare la rete neurale (4).
Nella prima fase del preprocessamento sono state scaricate le immagini da un URL di download, successivamente è stato creato una variabile in cui sono presenti tutti i metadati delle immagini e salvate in un file chiamato wiki.dat in formato MatLab, convertito successivamente in una struttura dati di Python(*e.g.* dictionary).

Nella seconda fase del preprocessamento è stata utilizzata la rete neurale *CNN_face_detection_model_v1*, della libreria *Dlib*, per rilevare ed estrarre i volti dalle immagini del dataset.
Infine le immagini sono ridimensionate a $32x32$, poichè la dimensione dell'immagine non ha importanza per l'addestramento di qualsiasi modello CNN.Infine, Sono state archiviate le immagini sul disco seguendo il seguente schema:

```
meta_data = {
    "images" : images,
    "name" : name,
    "age" : age,
    "gender" : gender
}
```

3. Il modello

Per la risoluzione di entrambi i task di *age* e *gender* detection, si è optato per l'utilizzo di una rete neurale convoluzionale (*CNN*), ottimizzata considerando le best practice note per il deep learning. Sono stati sviluppati due modelli, uno per la predizione dell'età e uno per il riconoscimento del genere, che differiscono in qualche particolare. Per lo sviluppo del modello sono state utilizzate le librerie *TensorFlow* e *keras*, più in particolare la funzione *tf.keras.Sequential()* crea la rete neurale come uno stack sequenziale di strati, ognuno dei quali riceve in input e restituisce in output esattamente un tensore.
Per quanto riguarda la fase di addestramento di entrambi i modelli, si è optato per suddividere il dataset come segue:

- 80% suddiviso in:
  - 80% training
  - 20% validation
- 20% test

Nelle prossime sottosezioni si discute nel dettaglio della struttura di ogni modello.

3.1 genderDetectionCNN

La CNN dedicata al task di gender detection presenta uno strato di input, che istanzia un tensore Keras di dimensioni $32x32x3$:

```
layers = [Input(shape=(32,32,3))]
```

Successivamente, si concatenano 4 blocchi convoluzionali alla rete neurale:

```
no_of_conv_layers = (16, 32, 64, 128)
for i in no_of_conv_layers:
    layers += [
```

```
            Conv2D(i, padding='same', kernel_size=(2,2)),
            Activation('relu'),
            BatchNormalization(),
            MaxPooling2D(pool_size=(2,2), strides=2, padding='same')
    ]
```

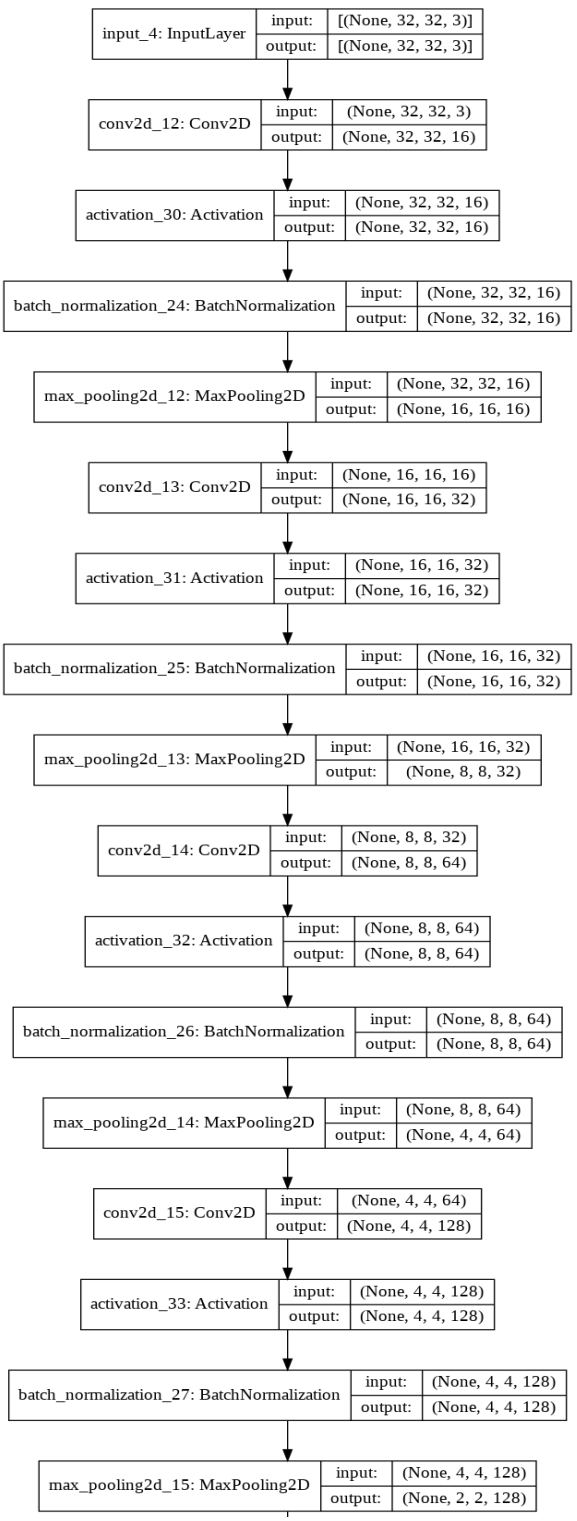
ogni blocco comprende lo strato convoluzionale *Conv2D* che comprende un numero di filtri diverso per ogni strato (potenza di 2 compresa in [16,128]), una funzione di attivazione *ReLu* che garantisce la non linearità, una batch normalization e uno strato di pooling *MaxPooling2D* che opera il campionamento dei parametri in 2D, con stride uguale a 2. Successivamente al core della rete convoluzionale, si concatena immediatamente uno strato *Flatten*, per appiattire l'input, e una serie di strati densi separati batch normalization e *Droput*(con tasso 0.25) per ridurre l'overfitting e il problema dell'*exploding/vanishing gradient*:

```
layers += [
           Flatten(),
           Dense(512),
           Activation('relu'),
           BatchNormalization(),
           Dropout(0.25),
           Dense(256),
           Activation('relu'),
           BatchNormalization(),
           Dropout(0.25),
           Dense(128),
           Activation('relu'),
           BatchNormalization(),
           Dropout(0.25),
           Dense(64),
           Activation('relu'),
           BatchNormalization(),
           Dense(16),
           Activation('relu'),
           Dense(2),
           Activation('softmax')
]
```

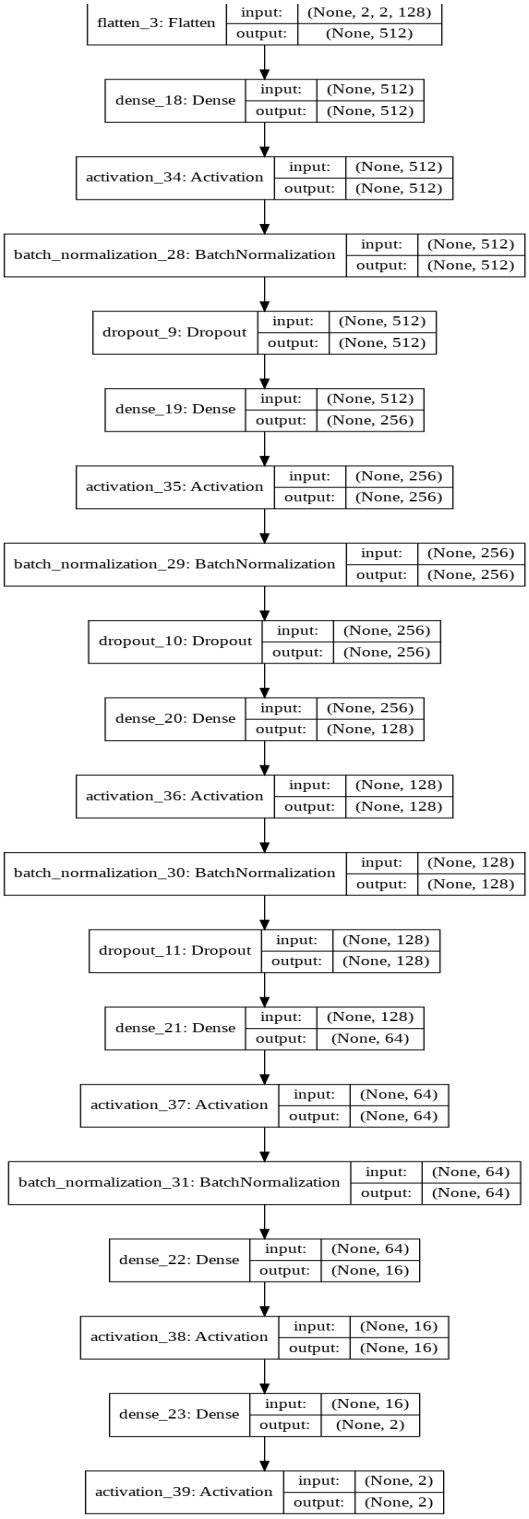
Il modello, infine, è compilato settando un ottimizzatore *Adam*(tf.keras.optimizers.Adam()), come funzione di loss la *categorical crossentropy* e come metrica l'*accuracy*. Durante l'addestramento si è scelto di adottare la tecnica dell'*Early Stopping*, che ,monitorando il valore della loss in validazione, prevede l'interruzione forzata del train se tale valore cessa di migliorare nel corso delle varie epoch:

```
tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)
```

Di seguito si riporta il diagramma completo del modello *genderDetectionCNN*(1)



(a) prima parte(filtri convoluzionali)



(b) seconda parte(strati densi e output)

Figure 2: illustrazione(in due parti) del modello *genderDetectionCNN*

3.2 ageDetectionCNN

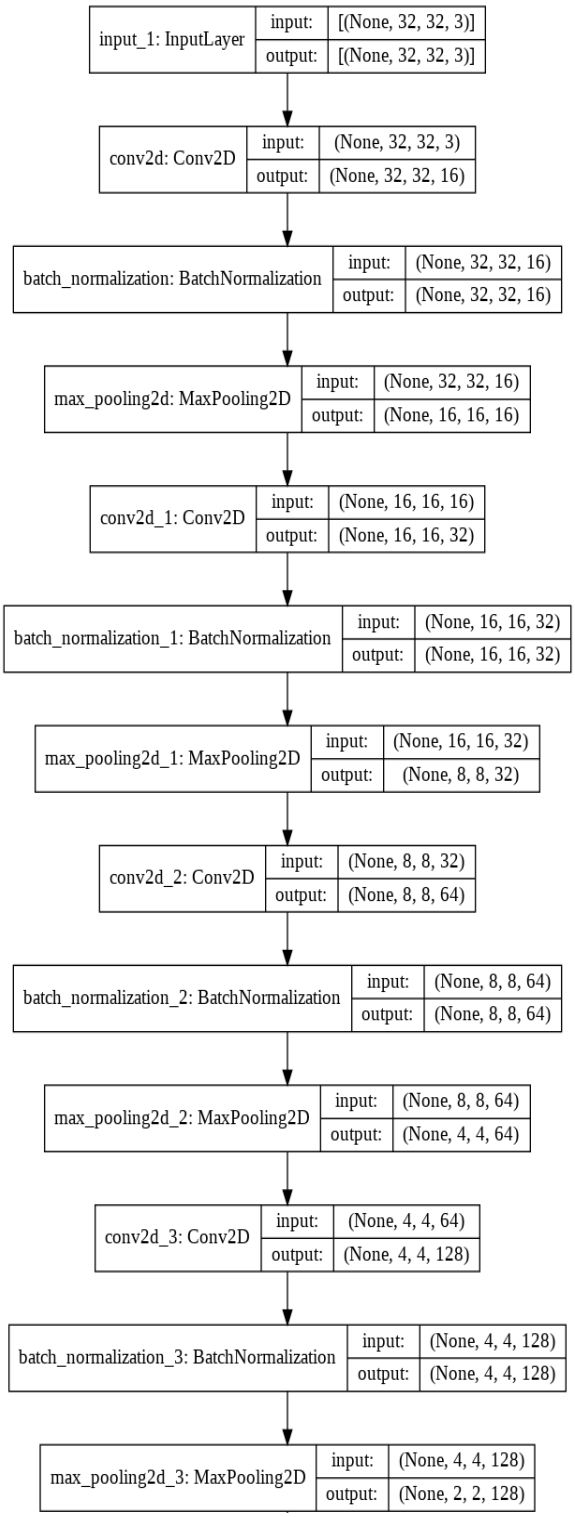
Il secondo modello è stato in realtà sviluppato in 3 diversi modi per via delle circostanze di sperimentazione(*e.g.* dataset). In particolare, il dataset utilizzato si è rivelato distribuito correttamente per la feature *gender*, ma decisamente meno per la feature *age*. Si è provato in prima fase a risolvere il task con un approccio di tipo regressione. Così come *genderDetectionCNN*, anche questo modello è una rete neurale convoluzionale con uno strato di input di dimensioni $32x32x3$, e 4 blocchi convoluzionali analoghi al modello precedentemente descritto, con la differenza che per quanto riguarda lo strato *Conv2D* è stata scelta una inizializzazione di *Xavier e He* e di utilizzare la *L2 regularization*:

```
Conv2D(i, padding='same', kernel_size=(2,2),
       kernel_initializer=tf.keras.initializers.HeNormal(),
       kernel_regularizer=tf.keras.regularizers.L2(1e-5))
```

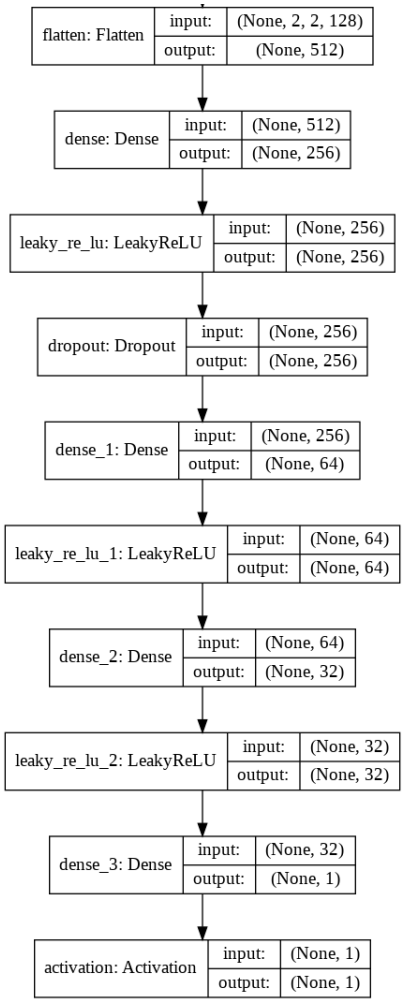
Successivamente, dopo lo strato Flatten troviamo i soliti strati densi, con funzione di attivazione Leaky ReLu, separati da batch normalization, per concludere con uno strato denso di dimensione 1 e funzione di attivazione lineare,:

```
Flatten(),
Dense(256, input_dim=1,
      kernel_regularizer=tf.keras.regularizers.L2( 0.1 ),
      bias_regularizer=tf.keras.regularizers.L2( 0.1 )),
LeakyReLU( alpha=0.2 ),
Dropout(0.25),
Dense(64, input_dim=1,
      kernel_regularizer=tf.keras.regularizers.L2( 0.1 ),
      bias_regularizer=tf.keras.regularizers.L2( 0.1 )),
LeakyReLU( alpha=0.2 ),
Dense(32, input_dim=1,
      kernel_regularizer=tf.keras.regularizers.L2( 0.1 ),
      bias_regularizer=tf.keras.regularizers.L2( 0.1 )),
LeakyReLU( alpha=0.2 ),
Dense(1),
Activation('linear')
```

Anche per questo modello, l'addestramento prevede early stopping e ottimizzatore Adam, ma la funzione di loss utilizzata è la *Mean Absolute Error*, usata anche come metrica. Altre implementazioni di questo modello prevedono un approccio più verso un problema di classificazione, per valutare le prestazioni relativamente ad un dataset poco distribuito nei confronti della feature dell'età. In primo luogo si è provato a considerare l'output come una predizione di probabilità di appartenenza a una classe nel caso di multi-classe(8 classi di età: [1-20], [21-32], [33,44], [45-56], [57-68], [69-80], [81-92], [92+]), e nel caso binario([1-60], [60+]). Per entrambi i casi, la struttura della rete è fortemente simile a quella del task di gender detection, a differenza di qualche strato denso omesso. Di seguito è riportato il diagramma completo del modello *ageDetectionCNN*



(a) prima parte(filtri convoluzionali)



(b) seconda parte(strati densi e output)

Figure 3: illustrazione(in due parti) del modello *ageDetectionCNN*

4. Esecuzione e ambiente di sviluppo

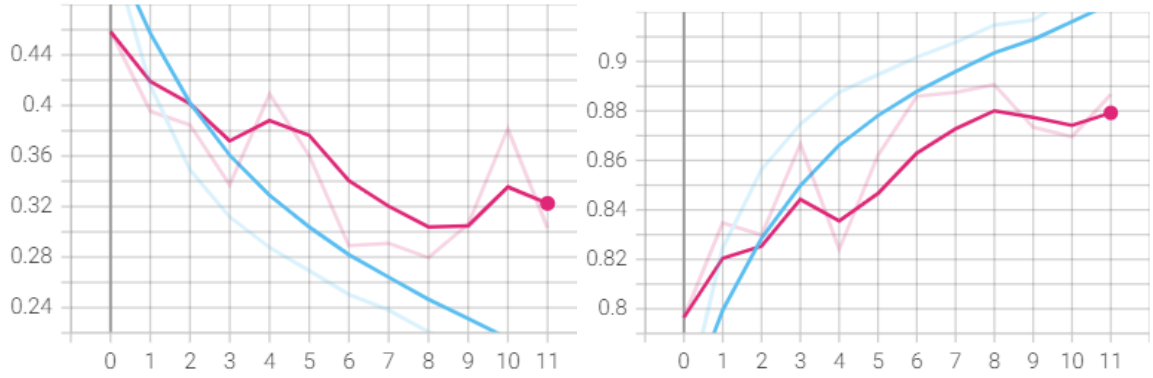
I modelli descritti sono stati scritti in python, più in particolare con l'ausilio delle librerie *TensorFlow* e *keras*, che permettono un'implementazione rapida ed efficiente di modelli complessi di deep learning. Si è scelto di sviluppare ed eseguire entrambi i modelli nell'ambiente computazione su cloud *Google Colab*, prevedendo l'utilizzo di GPU, 12.69GB di RAM, 73.27GB di disco. I tempi di addestramento sono stati prevalentemente brevi e sostenibili, la prossima sezione approfondisce i risultati sia della fase di training, che della fase di inferenza.

5. Risultati

In questa sezione sono riportati e descritti i risultati dell'addestramento della rete neurale, con conseguente valutazione del modello, e dell'utilizzo del modello in inference mode, modalità di produzione descritta in una delle prossime sottosezioni

5.1 Training, testing e validation

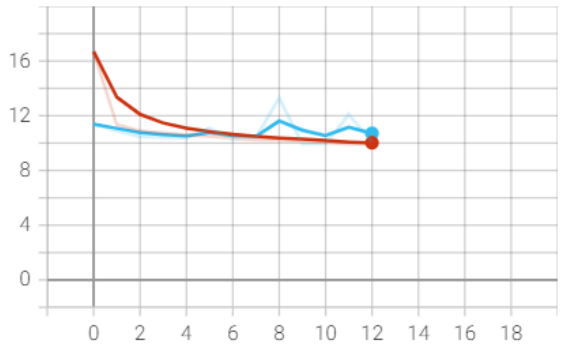
L'addestramento dei due modelli ha riscontrato risultati più o meno soddisfacenti. La prima rete neurale addestrata è stata *genderDetectionCNN*, su un dataset di 42,255 immagini in totale processato in batch da 64 unità(considerando 27,042 immagini di train dopo aver suddiviso il dataset come descritto precedentemente). Il numero di epoch scelto è 20, ma adottando la tecnica dell'early stopping quando il *validation loss* non cambia per 3 epoch consecutive. Considerando la *categorical crossentropy* come funzione di loss e l'*accuracy* come metrica, l'addestramento del modello ha mostrato i seguenti risultati:



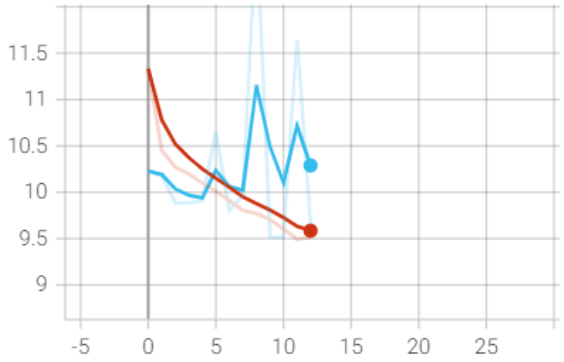
(a) genderDetectionCNN grafico epoch_loss (b) genderDetectionCNN grafico epoch_accuracy

ove la linea blu rappresenta il comportamento del modello, nel corso delle epoche, durante il train, mentre la linea rossa riguarda la validazione. Si noti come il primo grafico illustra un comportamento degli errori di validazione e di train consono con un modello no overfitting. Analogamente il secondo grafico mostra un comportamento notevole dell'accuracy che raggiunge, dopo 11 epoch per early stopping, il valore di 0.92 per il training e 0.88 per la validazione. Per quanto riguarda il modello *ageDetectionCNN*, come per il primo, l'addestramento ha previsto batch da 64 e 20 epoch con la tecnica dell'early stopping,

considerando però circa 100,122 immagini nel dataset, suddiviso nella stessa maniera. La funzione di loss è la *mean absolute error*, usata anche come metrica e spesso conosciuta anche come L1 Loss,che rappresenta la distanza matematica tra il valore predetto e quello effettivo. I risultati riportati dal modello sono i seguenti:



(a) ageDetectionCNN grafico epoch_loss



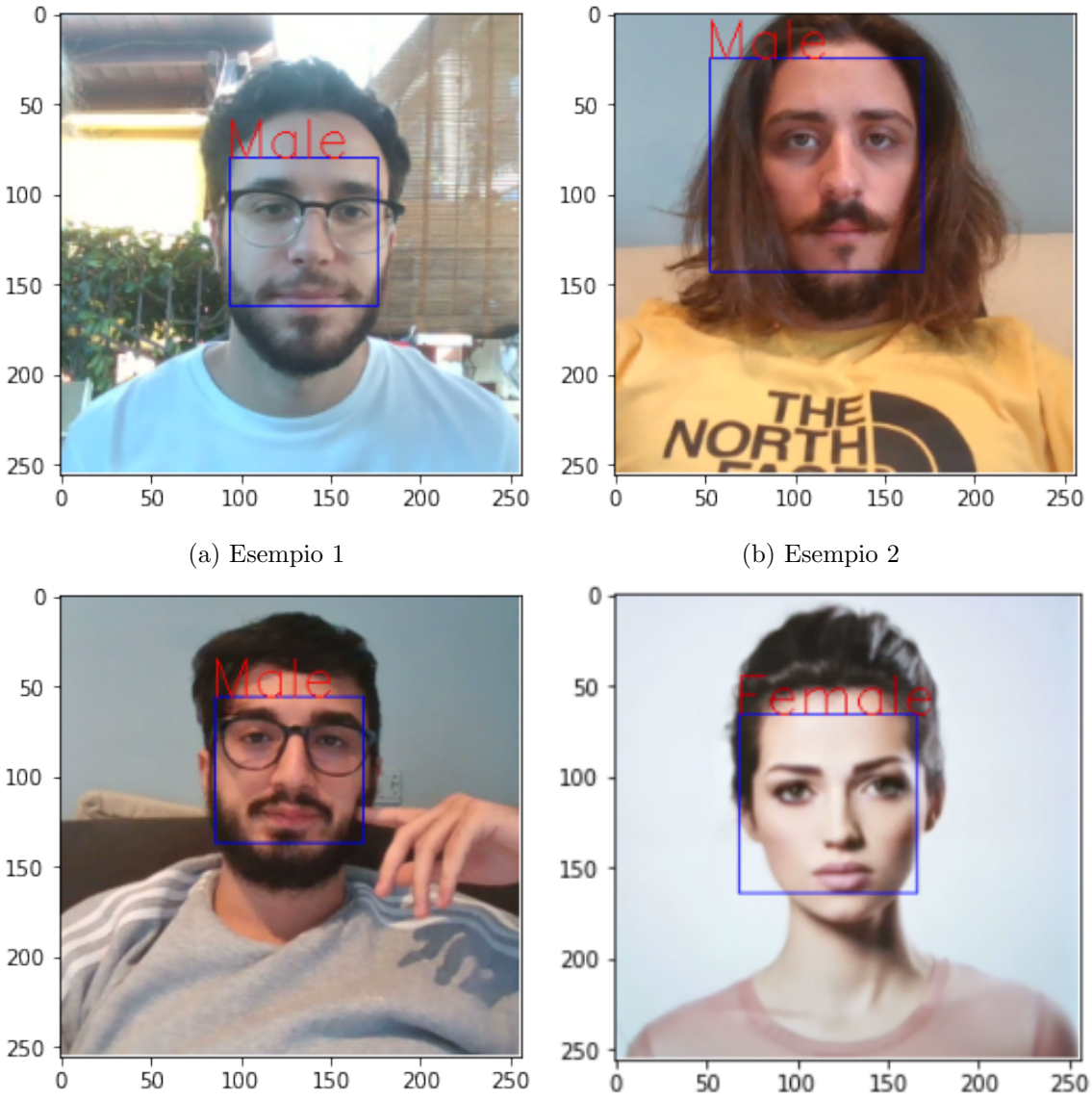
(b) ageDetectionCNN grafico graphic epoch_mae

In questo caso, la linea blu rappresenta il comportamento durante la validazione, mentre quella il comportamento durante il training. Si noti come questo modello ha riportato risultati meno soddisfacenti dal punto di vista dell'errore, con un minimo di 9.94 per il train e 10.05 per la validazione. Questo è probabilmente dovuto alla poca distribuzione dell'età rispetto agli esempi contenuti del dataset, nonostante sia stato scelto un dataset più grande rispetto alla sperimentazione dell'altro modello. Inoltre l'age detection è un problema di regressione lineare, più complesso di una classificazione binaria come gender detection.

5.2 Inference mode

Una volta terminato l'addestramento del modello e la relativa fase di calibrazione, abbiamo rilasciato la rete neurale in fase di "produzione" al fine di testare le capacità di quest'ultima fornendogli degli input creati manualmente direttamente dalla fotocamera dei dispositivi a nostra disposizione. In primo luogo è stata definita una funzione chiamata *rgba2rgb*, utilizzata per convertire un'immagine con formato RGBA in RGB (il codice è stato estratto da una risposta di stackOverflow di Feng Wang). Successivamente è stata definita una seconda funzione *take_photo*, che ha la funzione di catturare l'immagine utilizzando la webcam del computer. Poichè stiamo eseguendo il nostro notebook di Google Colab su un browser, è stato scelto di scrivere la funzione utilizzando il linguaggio JavaScript, per gestire nel modo più semplice possibile l'accesso alla webcam e l'acquisizione dell'immagine, per poi inviarla al server per l'elaborazione. A seguito di ciò è stata definita una terza funzione chiamata *extract_faces*, la quale riceve la nostra immagine come input e come avvenuto nella fase del preprocessamento del dataset viene filtrata utilizzando la CNN *CNN_face_detection _model _v1* che ci permette di riconoscere e ritagliare soltanto il volto dell'immagine. Infine, è stata definita un ultima funzione chiamata *show _output*, al quale vengono passate in input l'immagine, il volto ritagliato dalla funzione precedente e mostra in output il risultato ot-

tenuto dalla predizione effettuata dal modello. A seguire è riportato un esempio di output ottenuto nella modalità di inferenza del modello *genderDetectionCNN*.



(a) Esempio 1

(b) Esempio 2

(c) Esempio 3

(d) Esempio 4

Figure 6: Esempio di output del modello *genderDetectionCNN* in inference mode

6. Conclusioni

In questa relazione sono state descritte due soluzioni proposte in merito ai task di *gender* ed *age* detection. I modelli utilizzati sono delle reti neurali convoluzionali, adottando però le tecniche principali di ottimizzazioni per il deep learning, come ad esempio il *Dropout* o *Batch Normalization*. L'addestramento è stato eseguito su dataset contenenti immagini di volti di attori famosi, di cui sono riportate età e genere. Basandosi sui risultati, il modello più performante è *genderDetectionCNN*, per via del dataset che meglio distribuisce il genere piuttosto che l'età. Per migliorare l'accuratezza dei modelli si può sicuramente considerare un dataset molto più grande, oltre che aumentare la complessità dei modelli stessi, aggiungendo ad esempio più strati convoluzionali *Conv2D* ed aumentando il numero delle feature map generate dagli hidden layer. 2

References

Build a Gender Classifier using TensorFlow and Keras. towardsdatascience.com/build-a-gender-classifier-in-google-colab-using-tensorflow-keras\-and-tensorboard-2cd6f952d8aa.

Age and Gender Estimation in TensorFlow Workbook 1, Age Estimation. colab.research.google.com/github/shubham0204/Google_Colab_Notebooks/blob/main/Age_Estimation_(W1).ipynb/.

IMDB-WIKI – 500k+ face images with age and gender labels. data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/.

Preprocess and prepare a face dataset ready for CNN models. towardsdatascience.com/preprocess-and-prepare-a-face-dataset-ready-for-cnn-models-885867907eb0/.