UNIVERSITÀ
degli STUDI
di CATANIA

Corso di Laurea in Informatica Magistrale (LM-18)
Relazione di Computer Vision:

FACE BLURRING
BASED ON
AGE DETECTION

Studente:
Pierpaolo Gumina

Docente:
Prof. Sebastiano Battiato

Anno Accademico 2020/21

Abstract

Introduzione

La computer vision, o visione artificiale, è quel campo di studi interdisciplinare che si occupa di capire come i computer possano riprodurre processi e funzioni dell'apparato visivo umano. Non solo, quindi, acquisire le immagini statiche o in movimento (anche oltre lo spettro della luce naturale), identificarle, riconoscerle ed estrarne le informazioni utili per prendere decisioni: l'elaborazione digitale delle immagini (digital image processing) è infatti solo una parte della computer vision. Una parte che rientra, precisamente, nella early vision, l'elaborazione "a basso livello di astrazione" su cui, dagli anni Sessanta in poi, sono stati fatti numerosi passi avanti. La sfida più ambiziosa della computer vision riguarda la visione high level, "ad alto livello di astrazione e comprensione", che dall'immagine in 2D riesca a elaborare, ricostruire ed analizzare l'intero contesto in 3D in cui l'immagine è inserita. Riesca cioè a dare un significato storico e contestuale, quindi anche simbolico, dell'immagine rappresentata.

Negli ultimi anni la Computer Vision, grazie al Deep Learning, ha fatto notevoli passi in avanti consentendo al computer di riconoscere oggetti in immagini con una precisione tale da superare le prestazioni umane. Questo risultato ha impattato notevolmente nella vita di tutti inserendo algoritmi di Computer Vision in qualsiasi dispositivo dotato di camera.

Sebbene l'utilità della Computer Vision è innegabile, gli stessi dati acquisiti costantemente dalla camere presentano anche seri problemi di privacy che portano al bisogno di privacy visiva. La raccolta di foto e video da milioni di individui comporta rischi significativi per la privacy:

- Raccolta permanente. Le aziende che raccolgono dati di solito li conservano per sempre. Gli utenti da cui sono stati raccolti i dati non possono né cancellarli né controllare come saranno usati né influenzare ciò che si apprenderà da essi;
- Informazioni sensibili. Le immagini spesso contengono elementi sensibili catturati accidentalmente come volti, targhe, schermi di computer, indicazioni di posizione e altro. Questi dati visivi sensibili potrebbero essere usati in modo improprio o trapelare attraverso varie vulnerabilità;
- Preoccupazioni legali. I dati visivi conservati dalle aziende potrebbero essere soggetti a questioni legali o spionaggio senza mandato da parte di organizzazioni di sicurezza nazionale e di intelligence.

Obiettivo

Si vuole realizzare un algoritmo di classificazione in grado di riconoscere uno o più volti nella scena (face recognition), stimare l'età (age detection) e adottare tecniche di offuscamento (blurring) con lo scopo di anonimizzare i volti di individui minorenni.

Strumenti utilizzati

Il progetto utilizza librerie di riconoscimento di volti e stima di età pre-allenate in quanto lo scopo del progetto non è allenare una rete neurale bensì mettere insieme gli strumenti per la realizzazione di un algoritmo di face blurring basato sull'età.

Il progetto è sviluppato in Python utilizzando come IDE i notebook Jupyter offerti da Anaconda Navigator. Si tratta di un'applicazione che permette la creazione e la condivisione di documenti web nel formato JSON, che seguono uno schema e una lista ordinata di celle input/output. Queste celle offrono tra l'altro spazio per codici, testi in markdown, formule matematiche ed equazioni o contenuti multimediali (rich media).

L'elaborazione funziona su un'applicazione client basata sul web che si avvia con un browser standard. Basta che sul sistema sia installato e venga eseguito anche il server del Notebook Jupyter. I documenti Jupyter creati si possono esportare come documenti HTML, PDF, Markdown o Python o in alternativa si possono condividere con altri utenti tramite e-mail, Dropbox, GitHub o il proprio Notebook Jupyter.

Capitolo 1: Introduzione

1.1 OpenCV

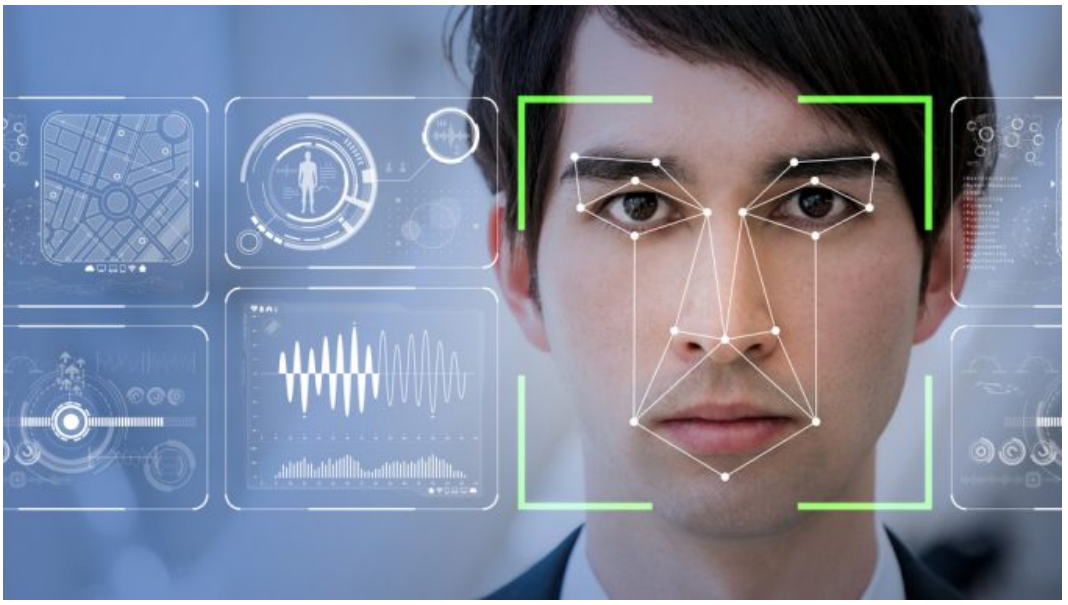
OpenCV [1] (Open Source Computer Vision Library) è una libreria software open source di computer vision e machine learning. OpenCV è stata costruita per fornire un'infrastruttura comune per le applicazioni di visione artificiale e per accelerare l'uso della percezione automatica nei prodotti commerciali. Essendo un prodotto con licenza BSD, OpenCV rende facile per le aziende utilizzare e modificare il codice.

La libreria ha più di 2500 algoritmi ottimizzati, che include un set completo di algoritmi di computer vision e machine learning sia classici che all'avanguardia. Questi algoritmi possono essere utilizzati per rilevare e riconoscere volti, identificare oggetti, classificare le azioni umane nei video, tracciare i movimenti della telecamera, tracciare oggetti in movimento, estrarre modelli 3D di oggetti, produrre nuvole di punti 3D da telecamere stereo, unire immagini per produrre un'alta risoluzione l'immagine di un'intera scena, trova immagini simili da un database di immagini, rimuovi gli occhi rossi dalle immagini scattate con il flash, segui i movimenti degli occhi, riconosci uno scenario e stabilisci marcatori per sovrapporlo alla realtà aumentata, ecc.

Insieme ad aziende affermate come Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota che utilizzano la libreria, ci sono molte startup come Applied Minds, VideoSurf e Zeitera, che fanno ampio uso di OpenCV. Gli usi implementati di OpenCV spaziano dall'unione di immagini di streetview, al rilevamento di intrusioni nei video di sorveglianza in Israele, al monitoraggio delle attrezzature minerarie in Cina, all'aiuto dei robot a navigare e a raccogliere oggetti al Willow Garage, al rilevamento di incidenti di annegamento in piscina in Europa, all'esecuzione di opere d'arte interattive in Spagna e New York, controllando le piste per detriti in Turchia, ispezionando le etichette sui prodotti nelle fabbriche di tutto il mondo fino al rilevamento rapido dei volti in Giappone.

Inoltre possiede API per linguaggi come C++, Python, Java e MATLAB e supporta Windows, Linux, Android e Mac OS.

1.2 Face detection



Un sistema di riconoscimento facciale (in inglese face recognition) è una soluzione tecnologica in grado di identificare una persona attraverso la valutazione dell'immagine del suo volto basandosi su una o più immagini che la ritraggono. L'applicazione di algoritmi di Intelligenza Artificiale ha reso queste soluzioni più sofisticate, rendendo possibile l'identificazione dei volti anche attraverso le variazioni della loro apparenza e il riconoscimento di una serie di informazioni aggiuntive alla mera identificazione, come età, genere, etnia, stato d'animo manifestato, e altro. Secondo alcuni ricercatori, grazie alle reti neurali, oggi un elaboratore può essere in grado di riconoscere una persona in mezzo a milioni di volti meglio di un essere umano.

La maggior parte dei sistemi di riconoscimento facciale attualmente in commercio, spiegano da TechTarget, funzionano con codici numerici chiamati "faceprints". Tali sistemi identificano un determinato numero di punti chiave o "nodali" su un volto umano. In questo contesto, i punti chiave o nodali sono punti di riferimento utilizzati per misurare le variabili del volto di una persona, come la lunghezza o la larghezza del naso, la profondità degli occhi e la forma degli zigomi. Questi sistemi funzionano catturando dati per i punti nodali su un'immagine digitale del volto di un individuo e memorizzando i dati risultanti come una impronta facciale. L'impronta facciale può quindi essere utilizzata come base per il confronto con i dati acquisiti da facce in un'immagine o un video.

I sistemi di riconoscimento facciale basati sulle "impronte facciali" possono identificare rapidamente e con precisione gli individui "obiettivo" quando le condizioni sono favorevoli. Tuttavia, se il volto del soggetto è parzialmente

oscurato o in un profilo anziché rivolto in avanti o se la luce è insufficiente, il software è meno affidabile. La tecnologia sta rapidamente evolvendo per superare questi limiti e ci sono diversi approcci, come la modellazione 3D e l'applicazione di algoritmi di Machine Learning che si basano su reti neurali artificiali, che possono accelerarne il progresso.

I sistemi di riconoscimento facciale vengono da sempre e comunemente usati per scopi di sicurezza, e sono sempre più utilizzati anche in una varietà di altre applicazioni.
La videosorveglianza può beneficiare enormemente degli avanzamenti compiuti nell'ambito dell'Intelligenza Artificiale, della computer vision, e del riconoscimento facciale.
Alcuni sistemi di pagamento mobile utilizzano il riconoscimento facciale per autenticare gli utenti in modo sicuro, vedasi il recente rilascio da parte di Apple del sistema di pagamento tramite Face ID.

1.2.1 Haar cascades

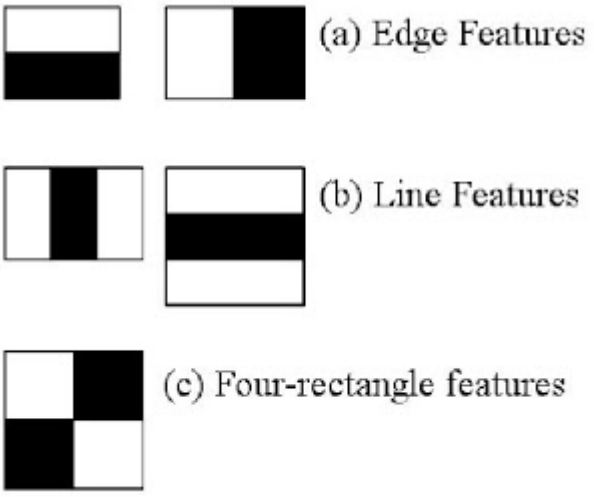
Esistono vari algoritmi pronti all'uso che permettono di realizzare un sistema di face detection, in particolare per questo progetto è stato utilizzato l'algoritmo Haar cascades.

L'Haar cascades[2] è stato introdotto da Viola e Jones nella loro pubblicazione del 2001 denominata Rapid Object Detection using a Boosted Cascade of Simple Features[3]. Negli ultimi anni sono stati sviluppati molteplici algoritmi che risultano più precisi come HOG + Linear SVM, SSD, Faster R-CNN, YOLO.
Il vantaggio ma l'Haar cascades risulta più rapido di quest'ultimi, ma di contro è soggetto al rilevamento di falsi-positivi risultando di conseguenza meno accurato.
L'algoritmo di face detection ideato da Viola e Jones si articola in un approccio basato sull'apprendimento automatico in cui una funzione a cascata viene addestrata da molte immagini positive e negative. Viene poi utilizzata per rilevare gli oggetti in altre immagini (testing).

Inizialmente, l'algoritmo ha bisogno di molte immagini positive (immagini di volti) e negative (immagini senza volti) per addestrare il classificatore.

OpenCV dispone di default una Haar cascades pre-addestrata per eseguire il rilevamento di un volto out-of-the-box.

Successivamente si procede estraendo le caratteristiche dalle immagini (vedi immagine sotto), tale procedure fa uso di kernel convoluzionali.

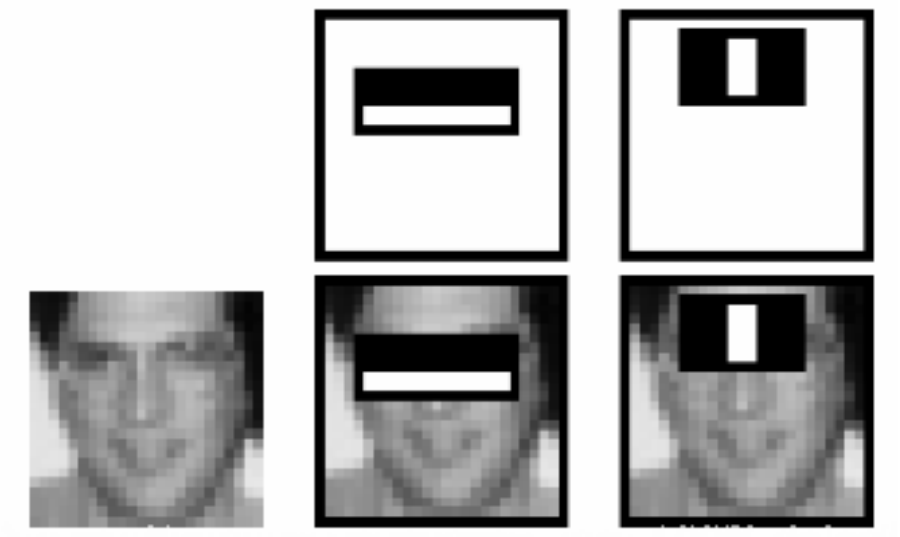


Per ottenere le caratteristiche per ciascuna di queste cinque aree rettangolari, si sottrae la somma dei pixel sotto la regione bianca dalla somma dei pixel sotto la regione nera. È interessante notare che queste caratteristiche hanno una reale importanza nel contesto del rilevamento dei volti:

- Le regioni degli occhi tendono ad essere più scure delle regioni delle guance.
- La regione del naso è più luminosa della regione degli occhi.

Calcolate le suddette cinque regioni rettangolari e la loro corrispondente differenza di somme, è possibile formare delle caratteristiche che possono classificare le parti di un viso.

Inoltre, per un intero set di caratteristiche, si utilizza l'algoritmo AdaBoost per selezionare quali corrispondono alle regioni facciali di un'immagine.

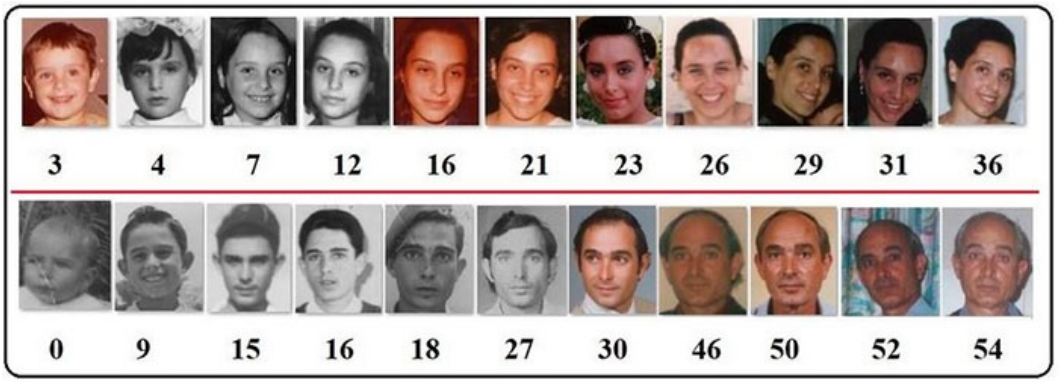


Tuttavia, usare una finestra scorrevole fissa e farla scorrere su ogni coordinata (x,y) di un'immagine, seguita dal calcolo di queste caratteristiche Haar-like, e infine eseguire la classificazione effettiva può essere computazionalmente costoso.

Per ovviare a ciò, Viola e Jones hanno introdotto il concetto di cascate o stadi. Ad ogni fermata lungo il percorso della finestra scorrevole, la finestra deve superare una serie di test dove ogni test successivo è più costoso computazionalmente del precedente. Se uno qualsiasi dei test fallisce, la finestra viene automaticamente scartata.

Alcuni vantaggi delle cascate Haar sono che sono molto veloci nel calcolare le caratteristiche Haar-like grazie all'uso di immagini integrali (chiamate anche tabelle di aree sommate). Sono anche molto efficienti per la selezione delle caratteristiche attraverso l'uso dell'algoritmo AdaBoost.
L’aspetto più importante dell’uso di Haar è che possono rilevare i volti nelle immagini indipendentemente dalla posizione o dalla scala del volto, inoltre, l'algoritmo Viola-Jones è in grado di funzionare in tempo reale.

1.3 Age detection



Il rilevamento dell'età è il processo di discernimento automatico dell'età di una persona esclusivamente da una foto del suo viso.
In genere il rilevamento dell'età è implementato come un processo in due fasi:

1. Rileva volti nell'immagine in ingresso/streaming video;
2. Estrazione della regione di interesse (ROI) del viso e applicazione l'algoritmo di rilevamento dell'età per prevedere l'età della persona.

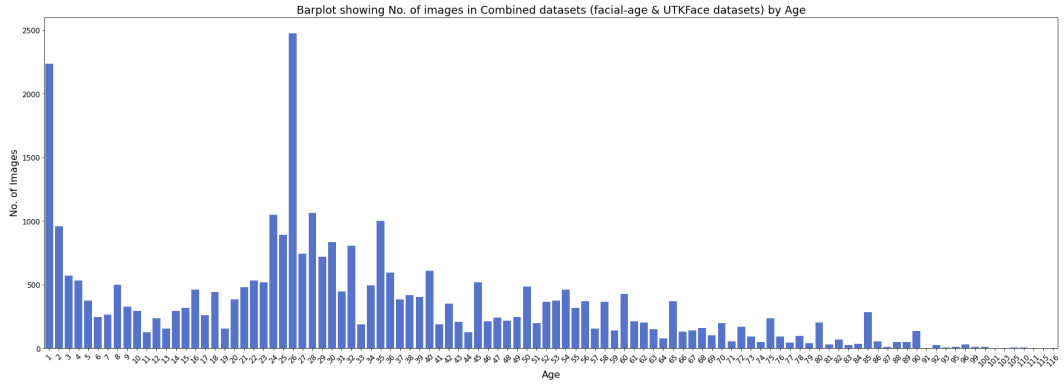
Per quanto riguarda il primo punto è stato discusso nel capitolo precedente.

Per il rilevamento dell'età è stato usato un modello pre-addestrato, in particolare il lavoro svolto da Prerak Agarwal nell'articolo "Age Detection using Facial Images: traditional Machine Learning vs. Deep Learning[4]".

Capitolo 2: Fase di allenamento

2.1 Raccolta dati e analisi esplorativa

Per allenare il modello di age detection, come spiegato nell'articolo di Prerak Agarwal, sono state reperite immagini con le relative etichette da UTKFace[5] e facial-age from kaggle[6], per un totale di 33.486 immagini RGB in formato JPG di dimensioni 200x200 pixel ciascuno.
La distribuzione dei dati raggruppati per età è la seguente:

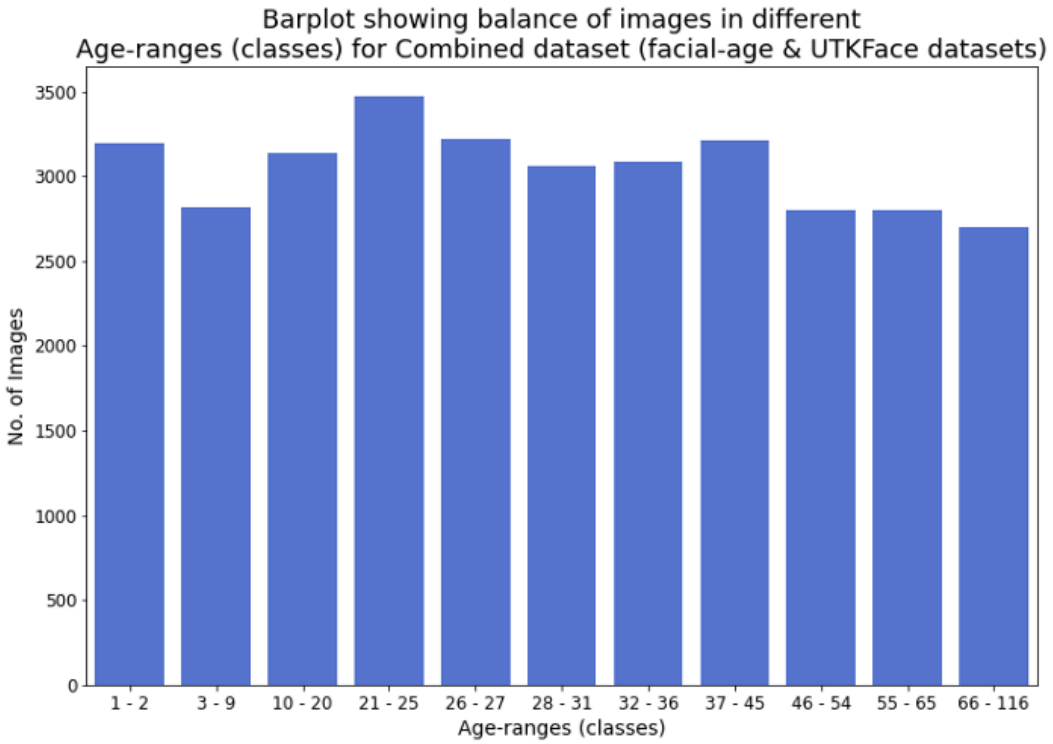


Il passo successivo è quello di dividere le diverse etichette di età in classi di fasce di età, ciò rispettando i seguenti vincoli:

1. Le classi devono essere equilibrate. Ciò garantisce che il modello di classificazione impari a classificare ogni fascia di età allo stesso modo.
2. Il numero di classi deve essere sufficiente. Il numero di classi di fasce d'età deve essere scelto in modo appropriato. Troppe classi si tradurranno in fasce d'età molto ristrette, che potrebbero influire negativamente sulle prestazioni del modello (generalmente è più difficile prevedere l'età di qualcuno fino all'anno esatto). Troppe poche classi risulteranno in fasce d'età molto ampie, che potrebbero non servire allo scopo del modello stesso di classificazione dell'età.
3. Ogni classe deve avere dati sufficienti. Il numero di classi di fasce d'età dipenderà anche dal numero risultante di immagini disponibili per ogni classe nel set di dati. Troppe classi si tradurranno in fasce d'età ristrette, riducendo così il numero di immagini disponibili per addestrare il modello su ogni classe.

La seguente tabella riassume quanto detto:

Class label	Age-ranges (classes)	No. of images	Class balance (%)
0	1 - 2	3192	9.53
1	3 - 9	2816	8.41
2	10 - 20	3136	9.37
3	21 - 25	3474	10.37
4	26 - 27	3217	9.61
5	28 - 31	3063	9.15
6	32 - 36	3086	9.22
7	37 - 45	3207	9.58
8	46 - 54	2802	8.37
9	55 - 65	2796	8.35
10	66 - 116	2697	8.05



2.2 Preparazione dei dati per la modellazione del classificatore

Per suddividere il set di dati in training set e test set si utilizza il metodo offerto dalla libreria scikit learn[7]:

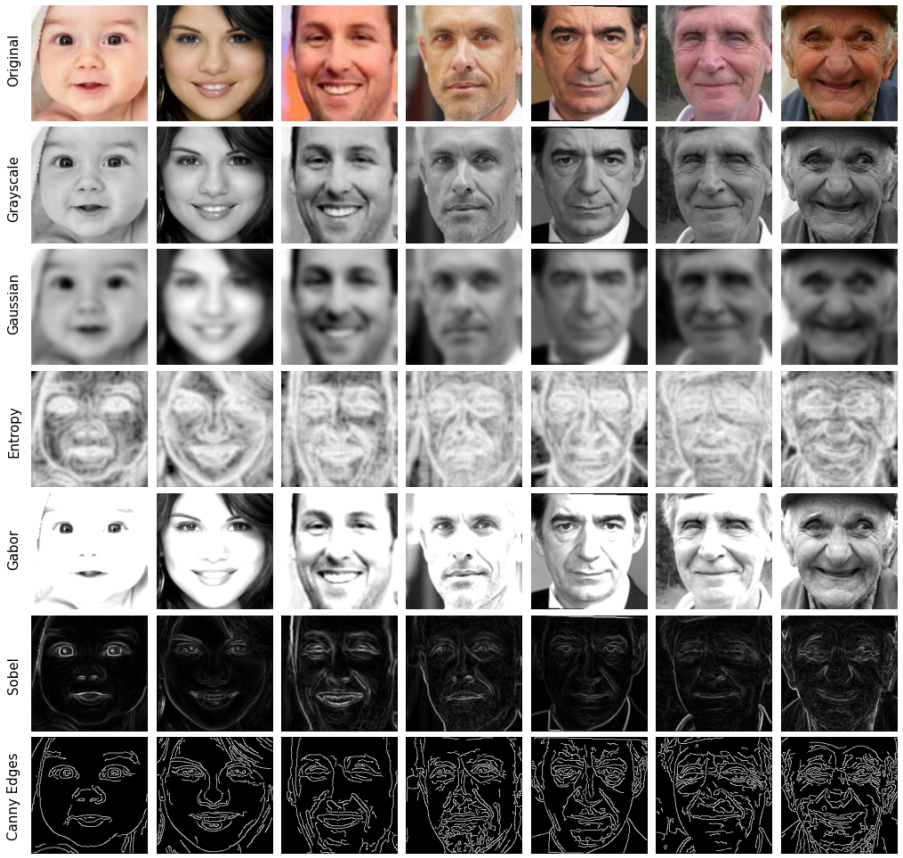
```
sklearn.model_selection.train_test_split
```

In particolare il set di dati è stato scelto di attribuire al training set il 70% del totale e il restante 30% per il test set.

2.3 Allenamento con un metodi di ML tradizionale

2.3.1 Estrazione delle features

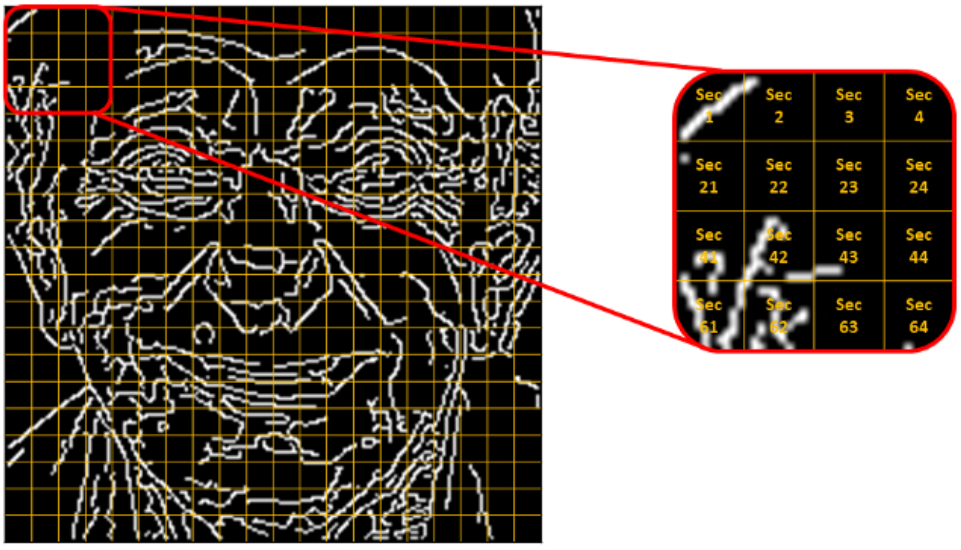
Per estrarre le caratteristiche dalle immagini facciali si applicano alcuni filtri diversi su alcune immagini dal set di dati e cercare visivamente di individuare eventuali differenze significative tra di loro. L'immagine sotto mostra i risultati di questo test:



Dall'immagine si nota che per ogni immagine i filtri evidenziano diverse caratteristiche nei volti di diversa età.

Dalle varie trame è possibile notare che Canny Edges risulta il filtro maggiormente utile per l'estrazione delle caratteristiche, in quanto con l'aumentare dell'età il filtro Canny aumenta di intensità.

Il passo successivo è convertire le immagini filtrate in scalari in modo da poterle inserire in un classificatore di machine learning. Per fare ciò, l'autore ha deciso di suddividere ogni immagine di 200x200 pixel in sezioni di 10x10 pixel.



Per ciascuna delle 400 sezioni risultanti, si calcola la media e la deviazione standard dei valori dei pixel. Ciò ha portato a 800 valori scalari univoci per ogni immagine, successivamente tabulato il tutto in un dataframe da utilizzare come features in un classificatore di machine learning.

2.3.2 Modelli di classificazione

Il processo di estrazione delle features, citato nel paragrafo precedente, è ripetuto per tutte le immagini sia per i dati di training che di test. Queste features sono state utilizzate in due diversi algoritmi di classificazione:

1. sklearn.ensemble.RandomForestClassifier
2. sklearn.svm.SVC

Entrambi i modelli sono stati ottimizzati per il valore di accuracy passando combinazioni multiple di iperparametri, attraverso il metodo *sklearn.model_selection.GridSearchCV.*

La seguente tabella mostra il riassunto dei risultati ottenuti:

	GridSearchCV best score (cv=5)	Train Accuracy	Test Accuracy
RandomForestClassifier	39.3%	66.8%	39.8%
SVC	49.0%	92.9%	53.4%

Da come è possibile vedere dalla tabella i modelli presentano un problema di overfitting e non generalizzano bene sui dati di testing non visti nella fase di training.

Anche se i valori di accuracy sono un pò alti per le fasce di età più giovani (1-2, 3-9, 10-20 e 21-25) e per le fasce di età più vecchie (di 66-116), c'è una presenza di misclassificazione significativa per le fasce di età centrali 26-65.

2.3.3 Limitazioni e ulteriori miglioramenti

Ci sono, naturalmente, una moltitudine di metodi che potrebbero essere utilizzati per migliorare i punteggi di accuracy e ridurre il grado di overfitting nei modelli. Per esempio, si potrebbero estrarre dalle immagini migliori caratteristiche di differenziazione utilizzando altre tecniche più complicate, o si potrebbero utilizzare altri classificatori per vedere se hanno prestazioni migliori.

2.4 Allenamento con un metodo di Deep Learning

2.4.1 Importazione dei set di dati

Per preparare il set di dati di immagini da passare nella rete neurale è necessario convertire le immagini in un dataframe di Pandas, con i valori dei singoli pixel nelle colonne e le immagini nelle righe. Tuttavia, poiché tutti i valori dei pixel dovevano essere di tipo float, ha comportato un dataframe di dimensioni eccessive (33.486 righe X 40.000 colonne). Il caricamento di questo dataframe nella memoria per poter addestrare la rete neurale ha portato a costanti errori di "memoria esaurita", come descrive l'autore Prerak Agarwal nell'articolo[4].

Per evitare questi errori, si adotta un altro approccio che prevede la creazione di pipeline di set di dati utilizzando l'API *.Dataset*[8] integrata di TensorFlow.

Questo approccio ha ridotto significativamente il consumo di RAM, poiché le immagini vengono caricate in memoria solo in batch di dimensioni molto più piccole, e solo quando sono strettamente necessarie alla rete neurale (anziché l'intero set di dati viene tenuto in memoria come nell'approccio precedente).

2.4.2 Modelli di classificazione

Dopo aver importato correttamente i set di dati, il passo successivo è quello di costruire un modello di rete neurale convoluzionale (CNN) di base che funzionasse con ragionevole accuratezza sui dati forniti e con il numero di parametri totali dati. L'idea dell'autore era di ottenere un benchmark iniziale sulle prestazioni del modello, quindi provare in modo incrementale diverse tecniche per vedere se migliorano le prestazioni da quel punto o meno. Dopo alcune esperimenti avanti e indietro con l'architettura della CNN, l'autore ha deciso di correggere la seguente architettura per cominciare:

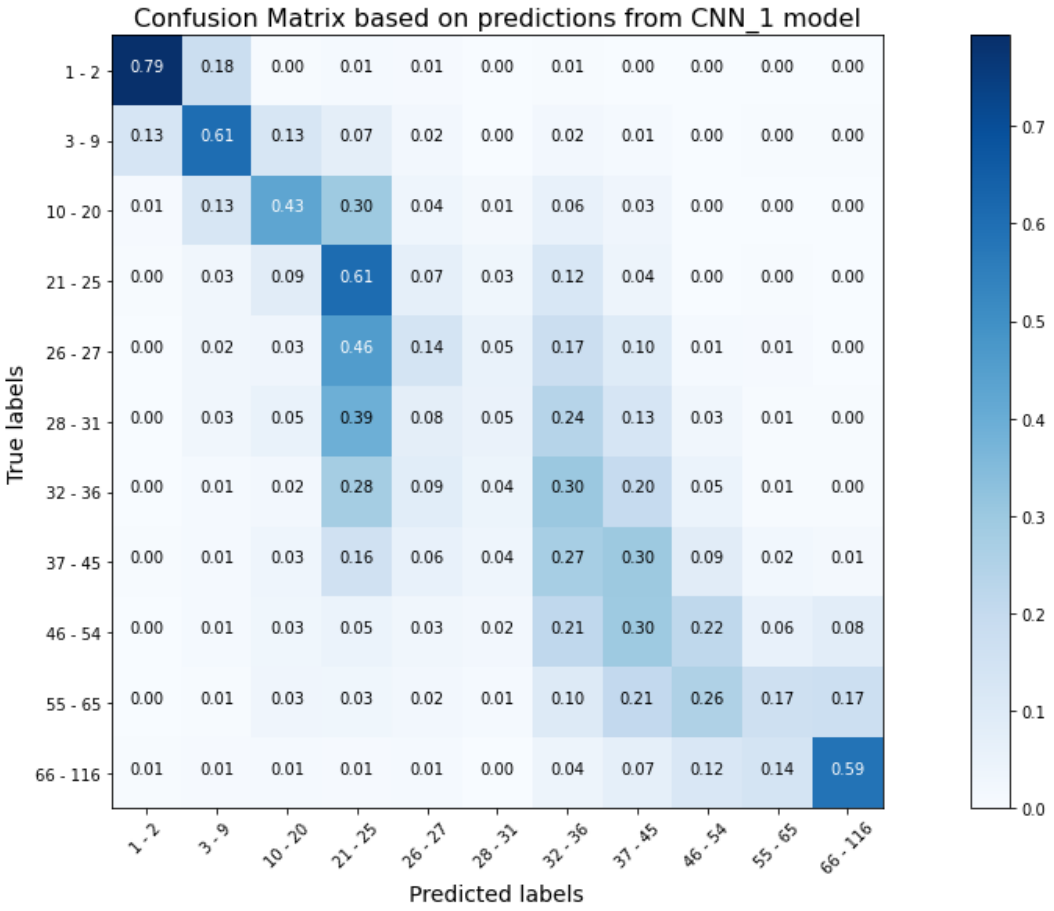
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 198, 8)       80
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 196, 196, 16)      1168
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 194, 194, 32)      4640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 97, 97, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 95, 95, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 47, 47, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
global_average_pooling2d (Gl (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 20)                1300
_________________________________________________________________
dense_1 (Dense)              (None, 11)                231
=================================================================
Total params: 35,163
Trainable params: 35,163
Non-trainable params: 0
_________________________________________________________________
```

Le immagini sono state convertite da colori RGB a scala di grigi prima di sottoporre a questo modello. Dopo aver settato il modello per 30 epoche, sono stati ottenuti i seguenti punteggi di loss e accuracy:

Model Description	Epochs	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
CNN with grayscale images	Early stop at 28 of 30 epochs	1.5355	1.6252	41.40%	38.34%

Come descrive la tabella il modello presenta un leggero overfitting e non sembra spiegare bene i dati. La matrice di confusione normalizzata ha inoltre mostrato che, in modo simile al tradizionale approccio di Machine Learning discusso nel paragrafo precedente, anche se i valori di accuracy sono alquanto alti per le fasce di età più giovani (tra 1-2, 3-9, 10-20 e 21-25) e per le fasce di età più anziane (da 66-116), vi è una presenza di significative classificazioni errate per le fasce di età tra 26-65. Ciò può essere attribuito al fatto che l'aspetto facciale delle persone (in generale) non cambia tanto durante le fasce di mezza età.

Si procede cercando di migliorare le prestazioni del modello utilizzando tecniche di data augmentation.

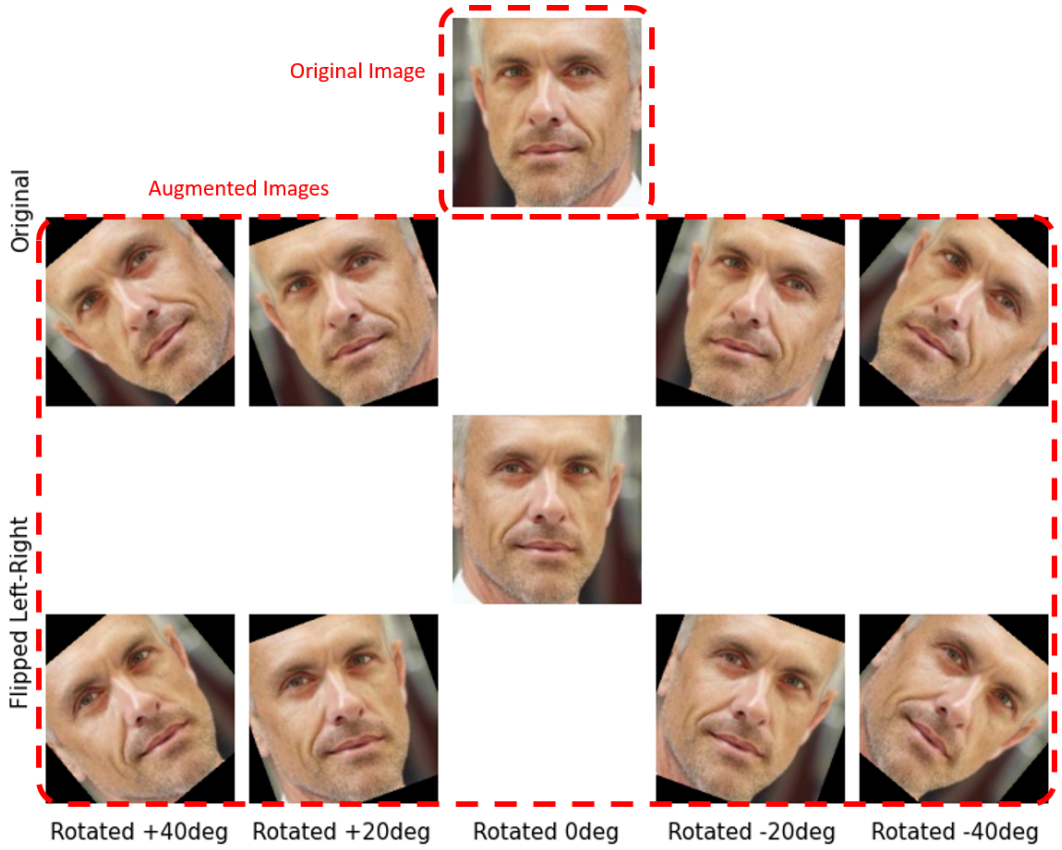


2.4.3 Strategie per migliorare il modello

Successivamente sono state adottate due strategie sono state provate per migliorare le prestazioni del modello:

1. Utilizzo di immagini colorate RGB invece di immagini in scala di grigi: la logica alla base di ciò era che forse l'aggiunta dei dati sui colori nelle immagini potrebbe far emergere alcune features all'interno del modello CNN che potrebbero migliorare le prestazioni complessive del modello.
2. Aumentare le immagini nel set di dati di addestramento: la logica alla base di ciò era che aumentare la quantità di dati per il modello di training contribuirebbe ad aumentare la varianza nel set di dati. Ciò può migliorare l'accuracy del modello riducendo al contempo la possibilità di overfitting. Per ogni immagine originale, sono state create altre 9 immagini, capovolgendo e ruotando le immagini di vari gradi.

Il set di dati di addestramento è conseguentemente aumentato per un totale di 234.400 immagini (rispetto alle 23.440 immagini nel set di dati di addestramento originale).



La stessa architettura del modello CNN, discusso precedentemente, è stata utilizzata per adattare il modello alle immagini RGB e al set di dati di addestramento aumentato, sono stati osservati i seguenti risultati:

Model Description	Epochs	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
CNN with grayscale images	Early stop at 28 of 30 epochs	1.5355	1.6252	41.40%	38.34%
CNN with RGB coloured images	30 epochs	1.4672	1.5971	43.56%	39.46%
CNN with grayscale images & augmented training dataset	30 epochs	1.4710	1.4727	42.51%	42.52%
CNN with grayscale images & augmented training dataset	60 epochs	1.3793	1.4028	45.45%	44.85%

I punteggi di accuracy nella tabella hanno evidenziato due risultati importanti:

1. L'utilizzo di immagini a colori RGB invece di immagini in scala di grigi potrebbe non migliorare le prestazioni del modello e potrebbe portare a un maggiore overfitting.
2. L'aumento delle immagini nel set di dati di addestramento ha contribuito a migliorare l'accuratezza del modello riducendo l'overfitting (anche dopo l'allenamento per 60 epoche).

2.4.4 Ripensare le fasce d'età

I risultati di accuracy dei modelli visti finora non risultano essere soddisfacenti, questo risulta prevedibile in quanto indovinare l'età di qualcuno è basato puramente sull'apparenza, e quindi molto soggettivo. Ci sono molti fattori che incidono nel modo in cui una persona si rapporta con la propria età, come la genetica, le condizioni di vita e le scelte di vita (dieta sana, quantità di esercizio fisico, abitudine nel fumare, ecc.).

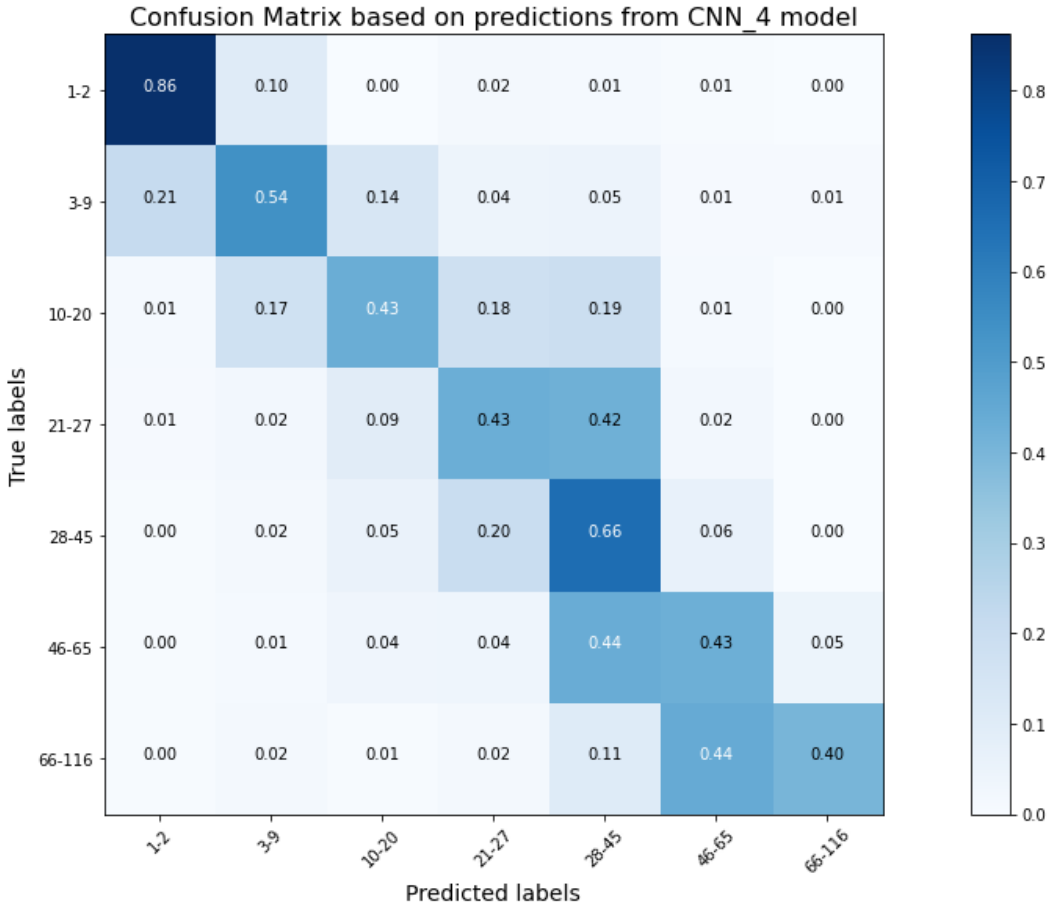
Come detto nel paragrafo precedente, anche con l'aumento dei dati, il punteggio di accuracy non è migliorato in modo significativo. Il problema principale è dovuto all'errata classificazione di persone di mezza età comprese tra 26-65 anni. Quindi, per evitare questo problema, l'autore ha deciso di ridistribuire nuovamente le fasce di età nelle classi. Questa volta, tuttavia, invece di limitarsi a considerare il numero di immagini disponibili per fascia di età, ha deciso di prendere in considerazione anche il fattore di intuizione umana e i punteggi di accuracy per le singole classi mostrati nella matrice di confusione del paragrafo precedente. Le fasce di età sono state ridistribuite nelle seguenti 7 classi:

Class label	Age-ranges (classes)
0	1 - 2
1	3 - 9
2	10 - 20
3	21 - 27
4	28 - 45
5	46 - 65
6	66 - 116

Procede la fase di addestramento del modello con le suddette modifiche, ottenendo:

Model Description	Epochs	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
CNN with grayscale images	Early stop at 28 of 30 epochs	1.5355	1.6252	41.40%	38.34%
CNN with RGB coloured images	30 epochs	1.4672	1.5971	43.56%	39.46%
CNN with grayscale images & augmented training dataset	30 epochs	1.4710	1.4727	42.51%	42.52%
CNN with grayscale images & augmented training dataset	60 epochs	1.3793	1.4028	45.45%	44.85%
CNN with grayscale images & re-distributed age-ranges	30 epochs	1.0265	1.1075	57.58%	54.17%

I punteggi di accuracy hanno mostrato un grado di adattamento simile a quello del primo modello CNN che si adattava alle immagini in scala di grigi. La matrice di confusione normalizzata relativa al quarto modello:



2.4.5 Ottimizzazione dell'architettura

Per ottimizzare l'architettura, l'autore, ha progettato più modelli di diverse architetture e confrontare le loro prestazioni in termini di loss e accuracy. Confrontando 18 modelli differenti l'autore ha scelto il modello seguente:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 198, 32)      320
_________________________________________________________________
average_pooling2d (AveragePo (None, 99, 99, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 97, 97, 64)        18496
_________________________________________________________________
average_pooling2d_1 (Average (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 128)       73856
_________________________________________________________________
average_pooling2d_2 (Average (None, 23, 23, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 256)       295168
_________________________________________________________________
average_pooling2d_3 (Average (None, 10, 10, 256)       0
_________________________________________________________________
global_average_pooling2d (Gl (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 132)               33924
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 931
=================================================================
Total params: 422,695
Trainable params: 422,695
Non-trainable params: 0
_________________________________________________________________
```

2.4.6 Costruzione del modello finale

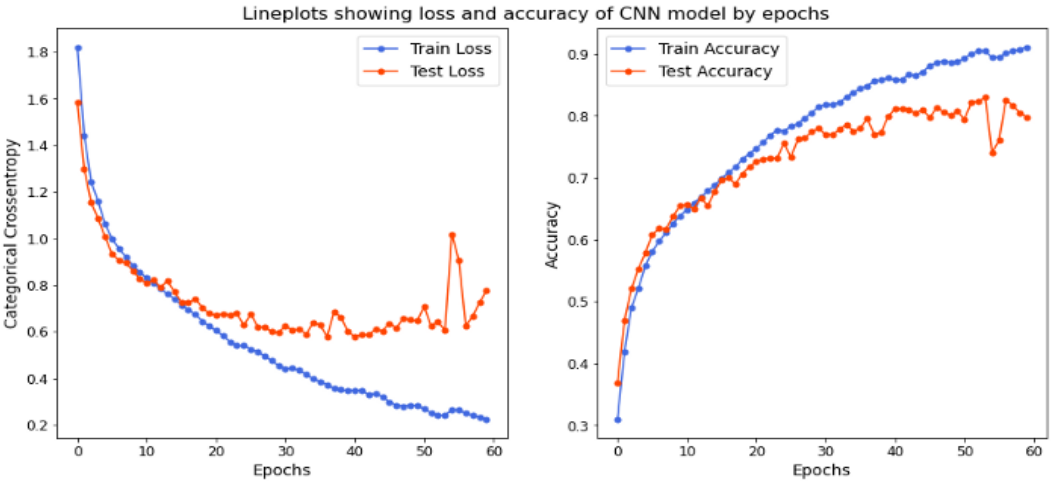
Il modello finale di training è composto da:

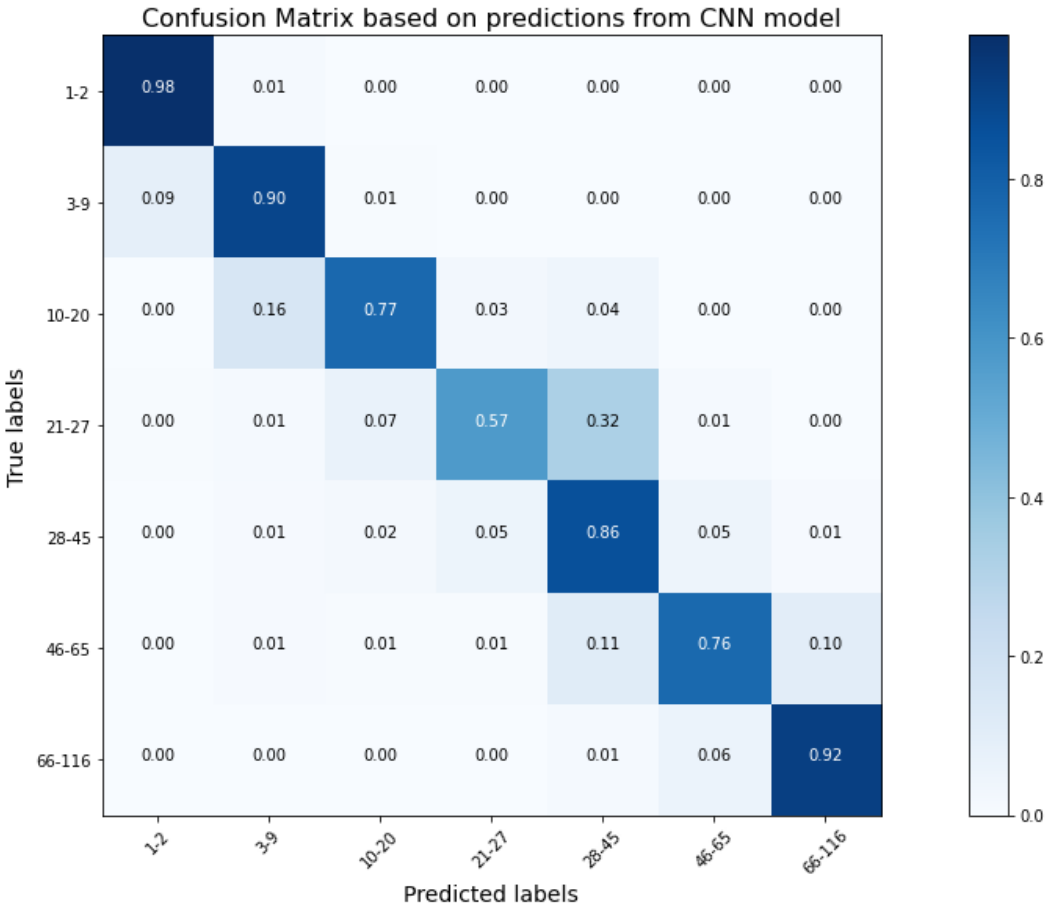
1. immagini in scala di grigi piuttosto di immagini a colori RGB;
2. set di dati di addestramento aumentato (234.400 immagini) invece del set di dati di addestramento originale (23.440 immagini);
3. allenamento per 60 epoche;
4. classi di età ridistribuite;
5. un'architettura ottimizzata.

Inoltre, si utilizza *tf.keras.callbacks.ModelCheckpoint*[9] come callback durante l'addestramento del modello finale in modo da poter salvare il modello mentre continua l'addestramento e il miglioramento delle prestazioni per oltre 60 epoche. Il modello ha raggiunto il picco di prestazioni all'epoca 54 su 60, con valori di loss e accuracy:

Model Description	Epochs	Train Loss	Validation Loss	Train Accuracy	Validation Accuracy
CNN with grayscale images	Early stop at 28 of 30 epochs	1.5355	1.6252	41.40%	38.34%
CNN with RGB coloured images	30 epochs	1.4672	1.5971	43.56%	39.46%
CNN with grayscale images & augmented training dataset	30 epochs	1.4710	1.4727	42.51%	42.52%
CNN with grayscale images & augmented training dataset	60 epochs	1.3793	1.4028	45.45%	44.85%
CNN with grayscale images & re-distributed age-ranges	30 epochs	1.0265	1.1075	57.58%	54.17%
CNN with grayscale images, augmented training dataset, re-distributed age-ranges & optimized architecture	Peak at 54 of 60 epochs	0.2430	0.6052	90.44%	82.97%

I punteggi di accuracy sono migliorati in modo significativo, anche se con un leggero grado di overfitting, che potrebbe essere considerato accettabile. Il grafico seguente mostra i cambiamenti nei punteggi di loss e accuracy durante l'allenamento del modello per 60 epoche. La matrice di confusione normalizzata mostra anche una significativa riduzione dell'errata classificazione tra le fasce di età compresa tra 26-65.


Lineplots showing loss and accuracy of CNN model by epochs
Train Loss
Test Loss
Categorical Crossentropy
Epochs
Train Accuracy
Test Accuracy
Accuracy
Epochs


Confusion Matrix based on predictions from CNN model
True labels
1-2
3-9
10-20
21-27
28-45
46-65
66-116
0.98 0.01 0.00 0.00 0.00 0.00 0.00
0.09 0.90 0.01 0.00 0.00 0.00 0.00
0.00 0.16 0.77 0.03 0.04 0.00 0.00
0.00 0.01 0.07 0.57 0.32 0.01 0.00
0.00 0.01 0.02 0.05 0.86 0.05 0.01
0.00 0.01 0.01 0.01 0.11 0.76 0.10
0.00 0.00 0.00 0.00 0.01 0.06 0.92
Predicted labels

2.4.7 Limiti e ulteriori miglioramenti

Anche l'approccio di deep learning presentato nel paragrafo precedente comporta i suoi limiti. Ad esempio, i set di dati utilizzati in questo progetto contenevano solo 33.000 immagini circa. Il modello CNN avrebbe potuto essere addestrato con un set di dati molto più grande con più variazioni nelle immagini in modo da ottenere risultati ancora migliori. Un altro approccio a questo progetto potrebbe essere quello di utilizzare il transfer learning (utilizzando i pesi di una rete neurale pre-addestrata) invece di creare e addestrare una rete neurale da zero.

2.4.8 Machine learning e deep learning a confronto

	Model	Train Accuracy	Validation Accuracy
Traditional Machine Learning	RandomForestClassifier	66.8%	39.8%
	SVC	92.9%	53.4%
Deep Learning	Convolutional Neural Network (CNN)	90.4%	83.0%

L'utilizzo del tradizionale approccio di ML in questo scenario, sebbene possibile, non ha prodotto modelli con punteggi di accuracy molto elevati. Inoltre, questo approccio richiede una notevole quantità di conoscenza del dominio ed esperienza nell'elaborazione dei dati per poter estrarre features significative dalle immagini e utilizzarle nei modelli di classificazione.

L'approccio del DL e delle reti neurali, d'altra parte, non richiede alcuna conoscenza significativa del dominio e competenza nell'elaborazione delle immagini, poiché non richiede l'estrazione manuale di features dell'immagine. Questo approccio ha anche prodotto modelli con prestazioni molto migliori con punteggi di accuracy più elevati. Pertanto, questo confronto tra i due approcci in questo progetto ha evidenziato il vero potere e le possibilità del deep learning con le reti neurali.

Capitolo 3: Implementazione

3.1 Setup

Il progetto è stato sviluppato interamente in un foglio Jupyter. Si inizia importando le librerie utilizzate tramite:

```
import numpy as np
import cv2
import os
from keras.models import load_model
import time
from datetime import datetime
```

Successivamente si carica il modello di age-detection pre-allenato necessario per il riconoscimento dell'età dei volti:

```
model = load_model("age_detect_cnn_model.h5")
age_ranges = ['1-2', '3-9', '10-20', '21-27', '28-45',
'46-65', '66-116']
```

Il rilevamento dei volti in Python può essere facilmente implementato utilizzando la libreria OpenCV. In questo notebook, per rilevare i volti, e' stato utilizzato il classificatore Haar Cascades. Il file del modello pre-addestrato in formato XML ed e' scaricabile[10] dal sito di OpenCV:

```
face_cascade =
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
```

Successivamente si definisce una funzione per restringere la regione del viso per migliorare la previsione nel modello:

```
def shrink_face_roi(x, y, w, h, scale=0.9):
    wh_multiplier = (1-scale)/2
    x_new = int(x + (w * wh_multiplier))
    y_new = int(y + (h * wh_multiplier))
    w_new = int(w * scale)
    h_new = int(h * scale)
    return (x_new, y_new, w_new, h_new)
```

In seguito, si definisce una funzione che crea la scritta con l'età prevista da sovrapporre all'immagine in input:

```
def create_age_text(img, text, pct_text, x, y, w, h):

    # Definizione del font, scale e spessore.
    fontFace = cv2.FONT_HERSHEY_SIMPLEX
    text_scale = 1.2
    yrsold_scale = 0.7
    pct_text_scale = 0.65

    (text_width, text_height), text_bsln =
cv2.getTextSize(text, fontFace=fontFace,
fontScale=text_scale, thickness=2)
    (yrsold_width, yrsold_height), yrsold_bsln =
cv2.getTextSize("years old", fontFace=fontFace,
fontScale=yrsold_scale, thickness=1)
    (pct_text_width, pct_text_height), pct_text_bsln =
cv2.getTextSize(pct_text, fontFace=fontFace,
fontScale=pct_text_scale, thickness=1)

    # Calcolo delle coordinate del punto centrale del
rettangolo contenete il testo.
    x_center = x + (w/2)
    y_text_center = y + h + 20
    y_yrsold_center = y + h + 48
    y_pct_text_center = y + h + 75

    # Calcolo delle coordinate dell'angolo in basso a
sinistra del testo in base alla dimensione del testo e al
punto centrale del rettangolo calcolato precedentemente.
    x_text_org = int(round(x_center - (text_width / 2)))
    y_text_org = int(round(y_text_center + (text_height /
2)))
    x_yrsold_org = int(round(x_center - (yrsold_width /
2)))
    y_yrsold_org = int(round(y_yrsold_center +
(yrsold_height / 2)))
    x_pct_text_org = int(round(x_center - (pct_text_width
/ 2)))
    y_pct_text_org = int(round(y_pct_text_center +
```

```
(pct_text_height / 2)))

    face_age_background = cv2.rectangle(img, (x-1, y+h),
(x+w+1, y+h+94), (0, 100, 0), cv2.FILLED)
    face_age_text = cv2.putText(img, text,
org=(x_text_org, y_text_org), fontFace=fontFace,
fontScale=text_scale, thickness=2, color=(255, 255, 255),
lineType=cv2.LINE_AA)
    yrsold_text = cv2.putText(img, "years old",
org=(x_yrsold_org, y_yrsold_org), fontFace=fontFace,
fontScale=yrsold_scale, thickness=1, color=(255, 255,
255), lineType=cv2.LINE_AA)
    pct_age_text = cv2.putText(img, pct_text,
org=(x_pct_text_org, y_pct_text_org), fontFace=fontFace,
fontScale=pct_text_scale, thickness=1, color=(255, 255,
255), lineType=cv2.LINE_AA)

    return (face_age_background, face_age_text,
yrsold_text)
```

In seguito la funzione adibita al blurring del volto:

```
def anonymize_face_simple(image, factor=3.0):
    # automatically determine the size of the blurring
kernel based
    # on the spatial dimensions of the input image
    (h, w) = image.shape[:2]
    kW = int(w / factor)
    kH = int(h / factor)
    # ensure the width of the kernel is odd
    if kW % 2 == 0:
        kW -= 1
    # ensure the height of the kernel is odd
    if kH % 2 == 0:
        kH -= 1
    # apply a Gaussian blur to the input image using our
computed
    # kernel size
    return cv2.GaussianBlur(image, (kW, kH), 0)
```

Successivamente di definisce una funzione per trovare i volti in un'immagine per poi classificare ogni volto in una delle fasce d'età definite in precedenza.
In particolare le prime 2 righe definiscono due copie dell'immagine di input, una per la sovrapposizione dell'età sul volto e l'altra, in scala di grigi, da passare al modello pre-allenato per dare la predizione sull'età. La terza riga si occupa di rilevare le facce nell'immagine usando la face_cascade, caricata precedentemente, e memorizzare le loro coordinate in una lista.
Il for, per ogni volto presente nell'immagine, provvederà a disegnare attorno ciascun volto un rettangolo con l'età predetta, inoltre in questo momento viene fatto il controllo sull'età predetta: se l'età è minore di 20 al volto viene applicato un filtro di blurring.

```
def classify_age(img):
    img_copy = np.copy(img)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(img_copy,
scaleFactor=1.2, minNeighbors=6, minSize=(100, 100))
    for i, (x, y, w, h) in enumerate(faces):
        face_rect = cv2.rectangle(img_copy, (x, y), (x+w,
y+h), (0, 100, 0), thickness=2)

        x2, y2, w2, h2 = shrink_face_roi(x, y, w, h)
        prova=img[y2:y2+h2, x2:x2+w2]
        face_roi = img_gray[y2:y2+h2, x2:x2+w2]
        face_roi = cv2.resize(face_roi, (200, 200))
        face_roi = face_roi.reshape(-1, 200, 200, 1)
        face_age =
age_ranges[np.argmax(model.predict(face_roi))]
        face_age_pct =
f"({round(np.max(model.predict(face_roi))*100, 2)}%)"

        face_age_background, face_age_text, yrsold_text =
create_age_text(img_copy, face_age, face_age_pct, x, y, w,
h)
        if(face_age in age_ranges[0:3]):
            face=anonymize_face_simple(prova)
            img_copy[y2:y2+h2, x2:x2+w2] = face
    return img_copy
```

Infine, vengono definite due funzioni adibite al salvataggio in locale delle immagini o video dati in input:

```
def new_img_name(org_img_path):
    img_path, img_name_ext = os.path.split(org_img_path)
    img_name, img_ext = os.path.splitext(img_name_ext)
    new_img_name_ext = img_name+"_WITH_AGE"+img_ext
    new_img_path = os.path.join(img_path,
new_img_name_ext)
    return new_img_path

def new_vid_name(org_vid_path):
    vid_path, vid_name_ext = os.path.split(org_vid_path)
    vid_name, vid_ext = os.path.splitext(vid_name_ext)
    new_vid_name_ext = vid_name+"_WITH_AGE"+".mp4"
    new_vid_path = os.path.join(vid_path,
new_vid_name_ext)
    return new_vid_path
```

3.2 Inferenze

Dopo aver definito tutte le funzioni necessarie per il riconoscimento del volto, age detection e face blurring, si procede sviluppando 3 modalità di inferenze differenti:

1) Basato su immagini: viene data in input un'immagine.
2) Basato su video:
    a) viene dato in input un video.
    b) viene utilizzata la webcam del pc, inferenza in modalità real-time.

3.2.1 Inferenze su immagini

Per poter fare inferenza su un'immagine è necessario definire un path, ovvero il percorso della directory dove è salvata l'immagine. Successivamente il file path viene passato al metodo di classificazione definito nel capitolo precedente. L'immagine di output viene quindi salvata nella stessa directory dell'immagine di input.

```
my_image = "./img_test/test.jpg"
img = cv2.imread(my_image)
age_img = classify_age(img)

try:
    new_my_image = new_img_name(my_image)
    cv2.imwrite(new_my_image, age_img)
    print(f"Saved to {new_my_image}")
except:
    print("Error: Could not save image!")

cv2.imshow("Age Detection on Live Video", age_img)
cv2.waitKey(0);
```

3.2.2 Inferenze su video

Per poter fare inferenza su video è necessario definire un path, ovvero il percorso della directory dove è salvato il video. Successivamente il file path viene passato al metodo di classificazione definito nel capitolo precedente.
Il video si aprirà in una nuova finestra. Per interrompere bisogna che il codice finisca di analizzare l'intero video, o premere il tasto "Q" della tastiera per uscire in maniera forzata.
Il video di output viene quindi salvato nella stessa directory del video di input.

```
my_video = "./video_test/test.mp4"
cap = cv2.VideoCapture(my_video)

# Checking if video can be accessed successfully.
if (cap.isOpened() == False):
    print("Unable to read video!")

# Getting the video frame width and height.
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# Defining the codec and creating a VideoWriter object to
save the output video at the same location.
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
new_my_video = new_vid_name(my_video)
out = cv2.VideoWriter(new_my_video, fourcc, 18,
(frame_width, frame_height))

while(cap.isOpened()):

    # Grabbing each individual frame, frame-by-frame.
    ret, frame = cap.read()

    if ret==True:

        # Running age detection on the grabbed frame.
        age_img = classify_age(frame)

        # Saving frame to output video using the
VideoWriter object defined above.
        out.write(age_img)
```

```
            # Displaying the frame with age detected.
            cv2.imshow("video",age_img)

            # Exiting if "Q" key is pressed on the keyboard.
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        else:
            break

# Releasing the VideoCapture and VideoWriter objects, and
closing the displayed frame.
cap.release()
out.release()
cv2.destroyAllWindows()
print(f"Saved to {new_my_video}")
```

E' possibile fare inferenze utilizzando la webcam per proprio pc:

```
# Creating a VideoCapture object.
cap = cv2.VideoCapture(0)

# Checking if camera opened successfully.
if (cap.isOpened() == False):
    print("Unable to read camera feed!")

# Getting the video frame width and height.
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

# Defining the codec and creating a VideoWriter object to save
the output video in the current working directory.
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
cur_time =
datetime.fromtimestamp(time.time()).strftime('%Y%m%d%H%M%S')
out = cv2.VideoWriter(f"Webcam_{cur_time}_WITH_AGE.mp4",
fourcc, 18, (frame_width, frame_height))

while(cap.isOpened()):

    # Grabbing each individual frame, frame-by-frame.
    ret, frame = cap.read()
```

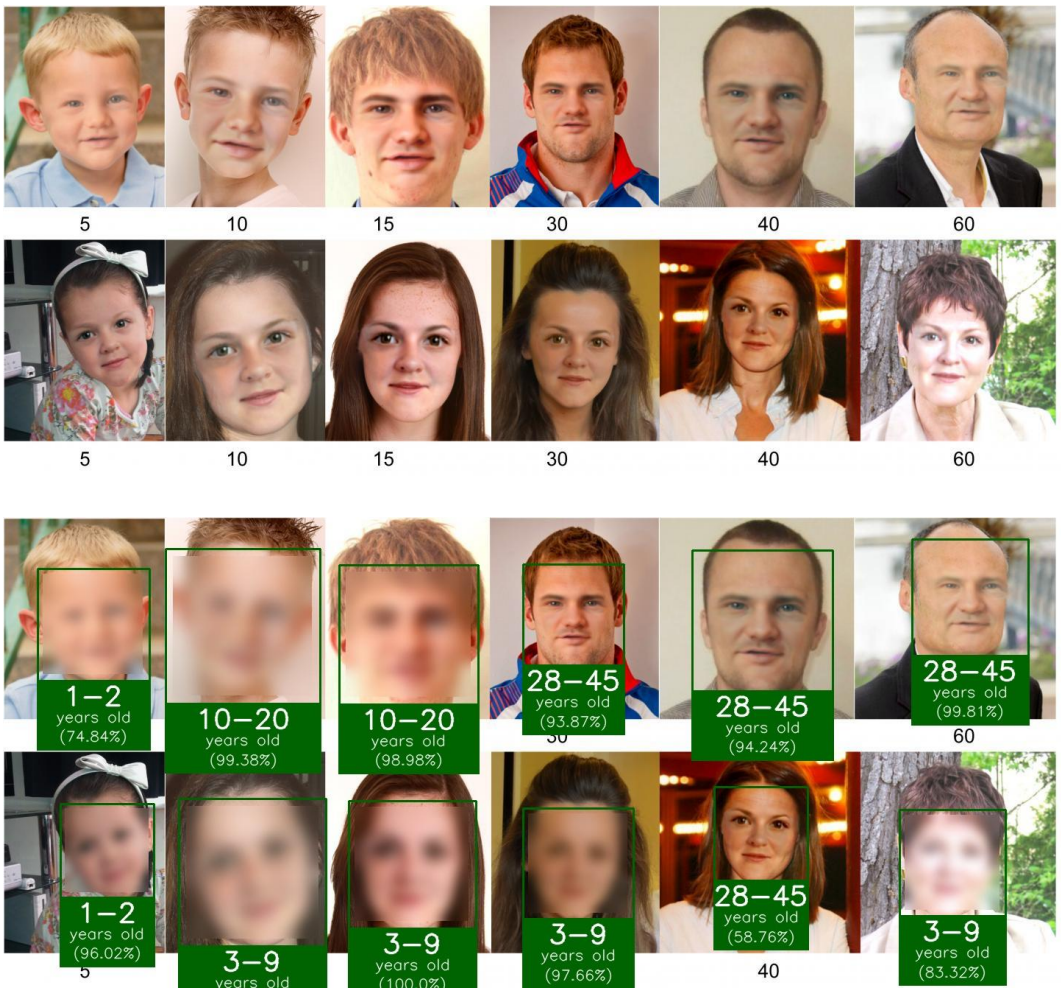
```
    if ret==True:

        # Running age detection on the grabbed frame.
        age_img = classify_age(frame)

        # Saving frame to output video using the VideoWriter
object defined above.
        out.write(age_img)

        # Displaying the frame with age detected.
        cv2.imshow("Age Detection on Live Video", age_img)

        # Exiting if "Q" key is pressed on the keyboard.
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    else:
        break

# Releasing the VideoCapture and VideoWriter objects, and
closing the displayed frame.
cap.release()
out.release()
cv2.destroyAllWindows()
print(f"Saved to Webcam_{cur_time}_WITH_AGE.mp4")
```

Capitolo 4: Limitazioni e conclusioni

La rilevazione automatica dell'età umana può essere impiegata in molte utili applicazioni della vita reale, come sistemi di servizio al cliente, distributori automatici, intrattenimento, sistemi di valutazione dell'efficacia della pubblicità ai clienti, sistemi che aiutano a prevenire che i minori comprino alcol, tabacco o accedono a siti web per adulti, anonimizzazione, ecc. Al fine di ottenere informazioni sull'età, sono stati sviluppati sistemi di stima dell'età basati sulle immagini utilizzando informazioni presenti nel volto umano.

Tuttavia, esistono delle limitazioni per gli attuali sistemi di stima dell'età, tra i quali i fattori relativo al movimento della telecamera, sfocatura ottica, espressioni facciali, sesso, ecc..

Un esempio è dato dalla seguente immagine:



Da come è possibile notare il classificatore fa alcuni errori specialmente nelle persone più adulte. Nell'ultima immagine, per esempio, il classificatore ha fatto confusione sull'età, questo probabilmente a causa dell'illuminazione eccessiva che ha agito coprendo i tratti distintivi di una persona adulta, come le rughe.

Concludo affermando che le tecniche di age detection possono essere molto utili per sviluppare sistemi di anonymization. Sviluppando algoritmi sempre più sofisticati e precisi si potrebbe riuscire a ridurre gli errori generati dal classificatore rendendo molto accurata la predizione.

Riferimenti

[1] About OpenCV
https://opencv.org/about/

[2] Cascade Classifier
https://docs.opencv.org/4.5.2/db/d28/tutorial_cascade_classifier.html

[3] Paper di Viola e Jones
(Rapid Object Detection using a Boosted Cascade of Simple Features)
https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[4] "Age Detection using Facial Images: traditional Machine Learning vs. Deep Learning" di Prerak Agarwal.
https://towardsdatascience.com/age-detection-using-facial-images-traditional-machine-learning-vs-deep-learning-2437b2feeab2

[5] https://susanqq.github.io/UTKFace/

[6] https://www.kaggle.com/frabbisw/facial-age

[7] Scikit learn library.
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[8] https://www.tensorflow.org/api_docs/python/tf/data/Dataset

[9] tf.keras.callbacks.ModelCheckpoint
https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint

[10] https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml