ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

Analisi e sviluppo di algoritmi per la valutazione della qualità delle immagini del volto

Tesi di laurea in:
Visione Artificiale

Relatore
Franco Annalisa

Candidato
Senni Mattia

Correlatore
Borghi Guido

II Sessione di Laurea
Anno Accademico 2024-2025

Abstract

La visione artificiale sta venendo sempre più utilizzata nel riconoscimento di un soggetto tramite immagini facciali, ambito in cui si stanno diffondendo sistemi di riconoscimento biometrico automatico. In questi contesti, è fondamentale garantire un'acquisizione controllata e standardizzata delle immagini facciali, al fine di ridurre errori di riconoscimento e vulnerabilità a minacce come il face morphing. Una cattiva acquisizione può compromettere l'intero processo biometrico, esponendo il sistema a minacce quali la falsificazioni e riducendone l'affidabilità. Questo progetto contribuisce allo sviluppo ed alla valutazione di un software che ha l'obbiettivo di implementare un draft ISO, volto a valutare la qualità dell'acquisizione delle immagini facciali secondo diverse metriche.

Contents

Abstract **iii**

1 Introduzione **1**
1.1 Motivazioni 1
1.2 Obbiettivo 3
1.3 Metodo 3

2 Implementazione della metrica per il rilevamento del difetto degli occhi rossi **5**
2.1 Obbiettivo della metrica 5
2.2 Requisiti preliminari 6
2.3 Implementazione della metrica 7
2.3.1 Descrizione dell'algoritmo 7
2.3.2 Dati di Input 8
2.3.3 Dati di Output 8
2.3.4 Algoritmo 8

3 Implementazione della metrica per il rilevamento dello sguardo frontale **11**
3.1 Obbiettivo della metrica 11
3.2 Requisiti preliminari 11
3.3 Implementazione della metrica 12
3.3.1 Descrizione dell'algoritmo 12
3.3.2 Dati di Input 12
3.3.3 Dati di Output 13
3.3.4 Algoritmo 13

4 Rilevamento dello sguardo frontale tramite CNN **15**
4.1 Selta del modello 15
4.2 Utilizzo del modello 15
4.3 Implementazione della metrica 17

4.3.1 Dati di Input 17
4.3.2 Dati di Output 17
4.3.3 Algoritmo 17

5 Rilevamento dello sguardo frontale tramite Machine Learning 19
5.1 Fine-Tuning di modelli CNN 19
5.1.1 Fine-Tuning di MobileNet V2 versione 1 20
5.1.2 Fine-Tuning di MobileNetV2 versione 2 22
5.2 Costruzione di un modello CNN 22
5.3 Costruzione di modelli di machine learning basati sui landmark facciali 24
5.3.1 Estrattore landmark MediePipe Face Mesh con modello XGBoost 26
5.3.2 Estrattore landmark MediePipe Face Mesh con modello Keras 27
5.3.3 Estrattore landmark ADNet con modello XGBoost 27
5.3.4 Estrattore landmark ADNet con modello Keras 27

6 Strumenti utilizzati per lo sviluppo delle metriche 29
6.1 Software OFIQ 29
6.2 Dataset utilizzati per la valutazione 31
6.2.1 ONOT 31
6.2.2 TONO 32
6.3 Onnx e OnnxRuntime 33
6.4 FVC Ongoing 33
6.5 Framework PyTorch 33
6.6 Jupyter Notebook, Numpy e Matplotlib 33

7 Valutazione delle metriche 37
7.1 Introduzione alle valutazioni 37
7.2 Valutazione su FVC-Ongoing 37
7.2.1 Metriche valutate 37
7.2.2 Il protocollo 37
7.2.3 Valutazione metrica per il rilevamento del difetto degli occhi rossi (versione HSV) 40
7.2.4 Valutazione metrica per il rilevamento del difetto degli occhi rossi (versione yCbCr) 41
7.2.5 Valutazione metrica per il rilevamento dello sguardo frontale (metodo algoritmico) 42
7.2.6 Valutazione metrica per il rilevamento dello sguardo frontale (metodo L2CS-Net) 43
7.3 Valutazione su dataset sintetici 44
7.3.1 Dataset utilizzati 44

7.3.2 Metriche valutate 44
7.3.3 Procedimento di valutazione 44
7.3.4 Valutazione metrica per il rilevamento dello sguardo frontale (metodo algoritmico) 45
7.3.5 Valutazione metrica per il rilevamento dello sguardo frontale (metodo L2CS-Net) 45
7.3.6 Valutazioni dei modelli di machine learning sviluppati 47

8 Analisi dei risultati **51**
8.1 Rilevamento del difetto degli occhi rossi 51
8.2 Rilevamento del difetto dello sguardo frontale 52
8.2.1 Versione algoritmica 52
8.2.2 Versione con l'utilizzo del modello L2CS-Net 53

9 Contribution **55**
9.1 Fancy formulas here 55

57

Bibliography **57**

List of Figures

2.1 Immagine dei landmark estratti dalla rete ADNet 6
2.2 Immagine descrittiva del procedimento per il rilevamento del difetto degli occhi rossi 10

3.1 Immagine rappresentativa dei rapporti utilizzato dall'algoritmo di stima dello sguardo frontale manuale 12

5.1 Matrice di confusione del fine tuning del modello mobilenetv2 versione 1 21
5.2 Matrice di confusione del fine tuning del modello mobilenetv2 versione 2 23
5.3 Matrice di confusione del modello CNN from scratch 24
5.4 Struttura della CNN implementata 25
5.5 Matrice di confusione dei modelli basati sui landmark 28

7.1 Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione HSV) 40
7.2 Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione yCbCr) 41
7.3 Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (metodo algoritmico) 42
7.4 Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (versione L2CS-Net) 43
7.5 Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo algoritmico) 46
7.6 Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo L2CS-Net) 47
7.7 Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo Fine Tuning Mobile Net v2, versione 1) su TONO e ONOT 48

7.8 Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo Fine Tuning Mobile Net v2, versione 2) su TONO e ONOT 49
7.9 Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo modello CNN) su TONO e ONOT 49

8.1 Esempi di immagini false positivi alla metrica del rilevamento degli occhi rossi dal dataset Biolab 51
8.2 Esempi di immagini false positivi e fale negativi nella metrica del rilevamento dello sguardo frontale con metodo algoritmico 52
8.3 Esempi di immagini false positivi e fale negativi nella metrica del rilevamento dello sguardo frontale con metodo L2CS-Net 53

List of Listings

listings/onnx_exporter.py 34

Chapter 1

Introduzione

1.1 Motivazioni

Nell'ambito della visione artificiale, e in particolare del riconoscimento facciale, si è ottenuta negli ultimi anni a una significativa riduzione dei tassi di errore grazie all'introduzione di algoritmi basati su tecniche di deep learning. Tuttavia, gli errori di riconoscimento rimangono ancora rilevanti e possono essere influenzati da numerosi fattori, tra cui le modalità di acquisizione dell'immagine, la cooperazione del soggetto durante la cattura biometrica, la specifica implementazione dell'algoritmo di confronto e la logica decisionale associata. In questo contesto, risulta evidente la necessità di un ammodernamento delle procedure di acquisizione, per evitare un peggioramento delle prestazioni dei sistemi biometrici a fronte dell'incremento dei volumi di dati da elaborare. L'urgenza di intervenire su questi aspetti si collega anche a una maggiore consapevolezza dell'importanza dell'usabilità dei sistemi biometrici, sia per gli utenti finali sia per gli operatori umani, i quali possono contribuire in modo significativo alla riduzione degli errori attraverso una migliore qualità nella fase di acquisizione. Spesso, infatti, un sistema progettato esclusivamente sulla base della tecnologia rischia di mostrare i propri limiti se non considera in modo integrato le dinamiche dell'interazione umana. Parallelamente, l'adozione crescente del riconoscimento facciale in contesti di ampia scala, come la gestione dei documenti di identità elettronici, le applicazioni commerciali e le attività di controllo delle forze dell'ordine, ha comportato la raccolta massiva di immagini

del volto, che fungono sia da campioni di riferimento in fase di registrazione sia da elementi di confronto successivo. Programmi di portata nazionale e internazionale, come quelli attivati in Cina, nell'Unione Europea, negli Stati Uniti e in India, testimoniano il crescente impiego di questa tecnologia in settori critici quali i trasporti, l'immigrazione e la verifica dell'identità. Tuttavia, molte delle immagini facciali oggi raccolte provengono da dispositivi di acquisizione non specificamente progettati per l'acquisizione di immagini del volto. Le immagini destinate a documenti d'identità o a database ufficiali vengono per lo più acquisite con telecamere configurate secondo specifiche documentarie, come lo standard ISO/IEC 39794-5, che ne disciplina l'inquadratura e le caratteristiche fotografiche. In assenza di strumenti automatici di valutazione della qualità, la verifica della conformità ai requisiti previsti dagli standard vigenti viene spesso affidata al fotografo. A questo si aggiunge il problema rappresentato da comportamenti involontari dei soggetti durante la cattura, portando ad una variazioni tra l'immagine di riferimento e quella di confronto, compromettendo l'efficacia del riconoscimento. È pertanto essenziale garantire la coerenza con la presentazione canonica prevista dagli standard, ossia un volto centrato e frontale, con espressione neutra, occhi aperti e privo di montature di occhiali che coprano gli occhi o troppo spesse. Considerando la grande eterogeneità degli utenti, per età, costituzione fisica, etnia, lingua, cultura, alfabetizzazione e familiarità con la tecnologia, è evidente l'importanza di un'attenta progettazione dal punto di vista dei fattori umani per migliorare l'acquisizione delle immagini facciali. Un'ulteriore criticità risiede nella separazione tra il processo di acquisizione dell'immagine e quello di valutazione della qualità, che spesso avviene solo in un secondo momento, quando la fotografia viene inviata a un server remoto. Se la qualità dell'immagine risulta inadeguata, si rende necessaria una nuova acquisizione aggravando tempi e costi. In questo scenario, il lavoro descritto da questa tesi si colloca nell'ambito della valutazione della qualità delle immagini del volto (ISO 29794-5), nell'ambito della specifica applicazione legata ai documenti di identità elettronici (ISO 39794-5).

1.2 Obbiettivo

Il progetto consiste in una valutazione del software OFIQ (Open Source Face Image Quality), strumento open source sviluppato per implementare le metriche descritte nel draft dello standard ISO precedentemente illustrato. Il lavoro consiste inoltre nella contribuzione allo sviluppo del software mediante l'implementazione di ulteriori metriche tra cui alcune suggerite dallo standard, ma non ancora integrate nel software. L'obiettivo principale è quello di verificare l'affidabilità di OFIQ per il controllo di qualità delle immagini facciali destinate all'identificazione biometrica, testandolo su esempi conformi alle specifiche e su esempi non conformi. Si vuole valutare se l'impiego del software durante la fase di acquisizione possa rappresentare uno strumento accurato per garantire la qualità e la conformità delle immagini biometriche.

1.3 Metodo

Per la realizzazione di questo progetto si è partiti con la comprensione delle varie metriche descritte nel draft ISO 29794-5:2024. Successivamente si è testata ogni singola metrica comparando diversi dataset divisi tra immagini che soddisfano i requisiti dello standard e immagini che non soddisfano la determinata metrica che si sta valutando, in seguito a queste valutazioni si guarda la distribuzione dei risultati.

Structure of the Thesis

Senni Mattia: At the end, describe the structure of the paper

Chapter 2

Implementazione della metrica per il rilevamento del difetto degli occhi rossi

2.1 Obbiettivo della metrica

L'obbiettivo di questa metrica è quello di rilevare il difetto degli occhi rossi all'interno di un viso. Il difetto degli occhi rossi nelle foto si verifica quando il flash della fotocamera illumina rapidamente la pupilla dell'occhio. La luce intensa penetra nella parte posteriore dell'occhio, raggiungendo la retina, che è ricca di vasi sanguigni. Poiché la luce viene riflessa direttamente verso la fotocamera dai capillari della retina, i quali sono rossi, l'effetto risultante nell'immagine è quello di pupille rosse anziché nere. Questo fenomeno è più comune in condizioni di scarsa illuminazione, quando le pupille sono più dilatate, permettendo a una maggiore quantità di luce di entrare e riflettersi. Dato che questo difetto noto altera il colore originale delle pupille è importante che un software per la valutazione di immagini di volti riesca a rilevarlo con una certa precisione. L'implementazione di questa metrica permette di dare un riscontro sotto forma di valutazione normalizzata da 0 a 100 sulla correttezza della foto rispetto a questo difetto.

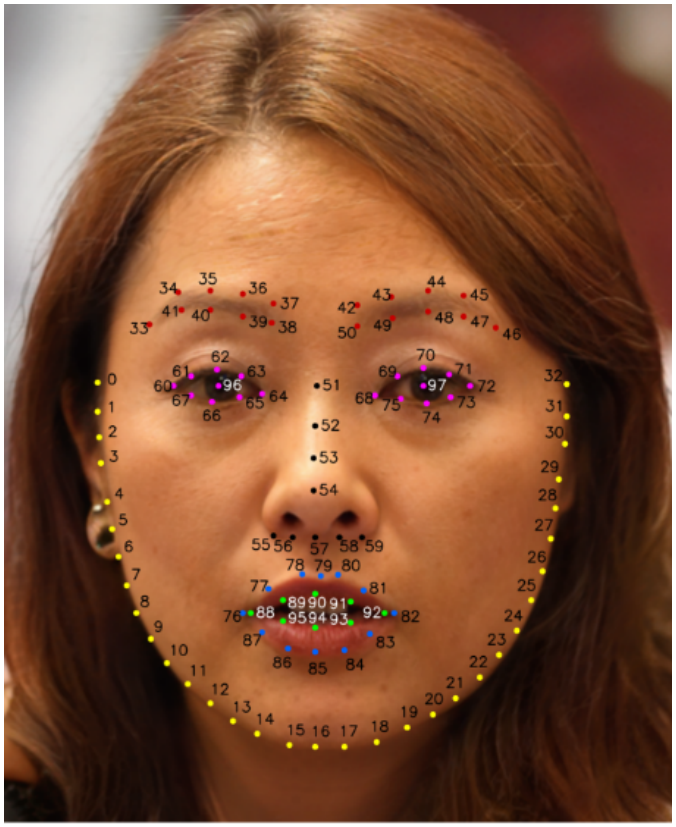


Figure 2.1: Immagine dei landmark estratti dalla rete ADNet

2.2 Requisiti preliminari

I requisiti preliminari di questo task sono i landmark del viso. Il draft della ISO utilizza come riferimento per l'estrazione dei landmark facciali la CNN ADNet allenata sul Wild dataset, prende in input un'immagine RGB e restituisce in output un set di 98 landmark fig. 2.1. I landmark che interessano questa metrica sono i seguenti:

- dal 60 al 67: contorno occhio sinistro
- dal 68 al 75: contorno occhio destro
- 96: pupilla occhio sinistro
- 97: pupilla occhio destro

2.3 Implementazione della metrica

2.3.1 Descrizione dell'algoritmo

Viene inizialmente effettuata la segmentazione dell'occhio, l'algoritmo inizia isolando l'area dell'occhio dall'immagine completa utilizzando i landmarks forniti. Una volta estratta la regione dell'occhio, l'algoritmo crea una maschera specifica per l'iride, escludendo sia la pupilla centrale (che appare normalmente scura) sia le aree esterne all'occhio. Questa segmentazione geometrica si basa su proporzioni anatomiche standard. Lo scopo dell'algoritmo è ora quello di rilevare le zone dell'iride dove sono presenti pixel rosse, per eseguire questo compito è necessario convertire l'immagine dal tradizionale spazio colori RGB ad alcuni più utili all'isolamento di determinati colori all'interno dell'immagine quali :

- HSV: separa il colore dalla luminosità rendendo la misurazione meno sensibilile alle variazioni di luce

- yCbCr: separa luminanza da crominanza garantendo una misurazione più robusta a luce ed combre, inoltre, come illustrato dal paper "Face Detection in Color Images", questo spazio colore è ottimo per segmentare elementi facciali contraddistinti dal colore rosso.

Nel caso dello spazio colori HSV viene scelta una soglia di identificazione del colore rosso su tutti e tre i calanli. Nel caso dello spazio colori yCbCr viene scelta una soglia minima per il canale cr, una massima per il canale cb ed inoltre vengono selezionati solamente i pixel con un valore di cr superiore alla medie per evitare problemi di saturazione. In entrambi i casi le soglie sono state scelte in maniera sperimentale, cercando di rispettare un giosto connubio tra falsi positivi e falsi negativi. Infine l'algoritmo calcola il rapporto tra i pixel rossi rilevati nell'iride e l'area totale dell'iride stessa, fornendo un valore normalizzato tra 0 e 1 che quantifica l'intensità dell'effetto occhi rossi. La soglia di rosso è stata scelta in base ai risultati dei test, cercando di avere un giusto equilibrio tra falsi positivi e falsi negativi, come ad esempio le persone con gli occhi marrone o marrone chiaro.

2.3.2 Dati di Input

- immagine: immagine digitale a colori
- puntiOcchio: vettore di punti che definiscono il contorno dell'occhio
- cornea: punto centrale della cornea

2.3.3 Dati di Output

- rapportoRosso: valore decimale tra 0 e 1 che rappresenta la proporzione di pixel rossi nell'iride

2.3.4 Algoritmo

1. Creazione maschera occhio [Fig. eye mask]
    - Crea una maschera nera delle dimensioni dell'immagine
    - Riempie l'area delimitata dai punti dell'occhio con bianco
    - Calcola il rettangolo che racchiude l'occhio
2. Estrazione regione di interesse (ROI)
    - Estrae la porzione di immagine corrispondente al rettangolo dell'occhio [Fig. eye roi]
    - Estrae anche la corrispondente porzione di maschera dell'occhio [Fig. eye mask roi]
3. Calcolo parametri geometrici
    - Calcola raggio pupilla = max(2, min(larghezza, altezza) / 8)
    - Calcola raggio iride = altezza / 2
    - Calcola centro pupilla rispetto alla ROI
4. Creazione maschera iride [Fig. iris mask]
    - Crea maschera nera delle dimensioni della ROI

- Disegna cerchio bianco con raggio iride centrato sulla pupilla
- Disegna cerchio nero con raggio pupilla per escludere la pupilla centrale

5. Combinazione maschere [Fig. combined mask]
   - Combina maschera occhio e maschera iride con operazione AND
   - Risultato: maschera che isola solo la regione dell'iride
6. Rilevamento colore rosso [Fig. red mask]
   - Versione con HSV
     - Converte l'immagine dell'occhio da BGR a HSV per migliore rilevamento colori
     - Definisce soglie HSV per il colore rosso:
       - Gamma 1: H(0-10), S(100-255), V(50-255)
       - Gamma 2: H(160-180), S(100-255), V(50-255)
     - Crea maschere separate per ciascuna gamma
     - Combina le maschere rosse con operazione OR
   - Versione con yCbCr
     - Converte l'immagine dell'occhio da BGR a yCbCr
     - Separa i canali cb e cr
     - calcola media cr = la media dei valori della matrice cr
     - calcola la matrice valoriAltiCr = maschera sui valori di cr maggiori di 150
     - calcola la matrice valoriBassiCb = maschera sui valori di cb minori di 120
     - calcola la matrice valoriMaggioreMediaCr = maschera sui valori di cr maggiori alla media cr
     - calcola red mask = end logico bit a bit tra valoriAltiCr, valoriBassiCb, valoriMaggioreMediaCr
7. Isolamento pixel rossi nell'iride [Fig. red iris mask]

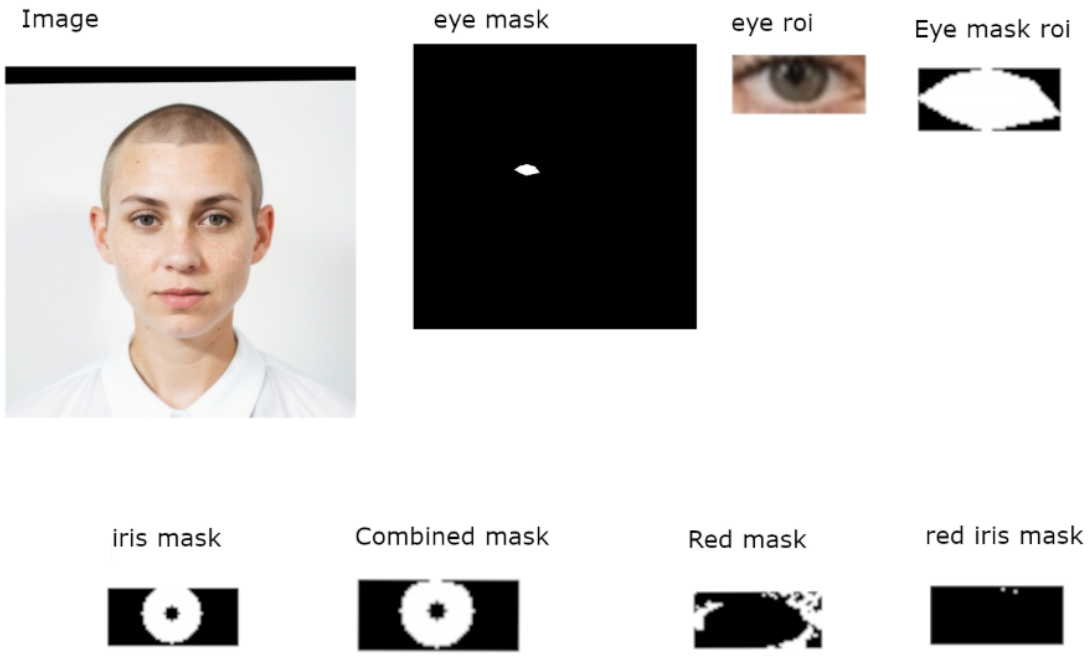


Figure 2.2: Immagine descrittiva del procedimento per il rilevamento del difetto degli occhi rossi

- Applica maschera rossa alla maschera iride combinata
- Risultato: pixel rossi presenti solo nell'area dell'iride

8. Calcolo rapporto finale
   - Conta pixel rossi nell'iride
   - Conta pixel totali nell'area iride
   - Se area iride < 0: rapportoRosso = pixelRossi / pixelTotaliIride
   - Altrimenti: rapportoRosso = 0
9. Ritorna rapportoRosso

Chapter 3

Implementazione della metrica per il rilevamento dello sguardo frontale

3.1 Obbiettivo della metrica

L'obiettivo della metrica è quello di assegnare uno score da 0 a 100 sul grado in cui un soggetto guarda in camera misura necessaria per garantire l'idoneità delle foto nei documenti di identità. Guardare dritto in camera è un requisito fondamentale per il riconoscimento facciale e per la conformità agli standard internazionali.

3.2 Requisiti preliminari

I requisiti preliminari di questo task sono i landmark del viso. Il draft della ISO utilizza come riferimento per l'estrazione dei landmark facciali la CNN ADNet allenata sul Wild dataset, prende in input un'immagine RGB e restituisce in output un set di 98 landmark fig. 2.1. I landmark che interessano questa metrica sono i seguenti:

- 60, 64: angoli occhio sinistro
- 68, 72: angoli occhio destro

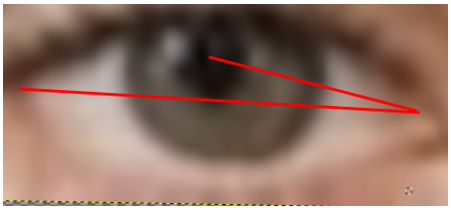


Figure 3.1: Immagine rappresentativa dei rapporti utilizzato dall'algoritmo di stima dello sguardo frontale manuale

- 96: pupilla occhio sinistro
- 97: pupilla occhio destro

3.3 Implementazione della metrica

3.3.1 Descrizione dell'algoritmo

L'algoritmo analizza la direzione dello sguardo misurando la posizione delle pupille all'interno degli occhi. Per ogni occhio, calcola la larghezza totale (distanza tra gli angoli interno ed esterno) e la distanza tra la pupilla e l'angolo interno. Dividendo queste due misure ottiene un rapporto che indica dove si trova la pupilla: un valore di 0.5 significa che la pupilla è perfettamente centrata, mentre valori minori o maggiori indicano sguardo verso sinistra o destra. L'algoritmo confronta entrambi gli occhi e sceglie quello con maggiore deviazione dal centro per determinare la direzione principale dello sguardo. Infine, applica una trasformazione parabolica (con parametri alpha = -400 e beta = 100) che mappa il risultato su una scala da 0 a 100, dove valori più alti indicano maggiore deviazione dalla posizione frontale. Questo approccio permette di quantificare oggettivamente quanto una persona stia guardando lateralmente rispetto alla fotocamera. Una visione grafica dei rapporti sopra elencati è disponibile

3.3.2 Dati di Input

- a: coordinate x ed y dell'angolo esterno dell'occhio sinistro
- b: coordinate x ed y dell'angolo interno dell'occhio sinistro

- c: coordinate x ed y dell'angolo esterno dell'occhio destro
- d: coordinate x ed y dell'angolo interno dell'occhio destro
- e: coordinate x ed y della pupilla dell'occhio sinistro
- f: coordinate x ed y della pupilla dell'occhio destro

3.3.3 Dati di Output

- valoreScalare: valore numerico che rappresenta la deviazione dello sguardo da 0 a 100

3.3.4 Algoritmo

1. Calcolo larghezza degli occhi
   - $larghezzaSinistra = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$
   - $larghezzaDestra = \sqrt{(d_x - c_x)^2 + (d_y - c_y)^2}$
2. Calcolo distanza pupilla-angolo interno
   - $distanzaSinistra = \sqrt{(e_x - b_x)^2 + (e_y - b_y)^2}$
   - $distanzaDestra = \sqrt{(f_x - d_x)^2 + (f_y - d_y)^2}$
3. Calcolo rapporti normalizzati
   - $rapportoSinistro = \frac{distanzaSinistra}{larghezzaSinistra}$
   - $rapportoDestro = \frac{distanzaDestra}{larghezzaDestra}$
4. Calcolo variazioni dal centro (*Nota: 0.5 rappresenta la posizione centrale ideale della pupilla*)
   - $variazioneSinistra = |rapportoSinistro - 0.5|$
   - $variazioneDestra = |rapportoDestro - 0.5|$
5. Selezione rapporto dominante

- Se $variazioneDestra > variazioneDestra$ allora: $punteggioGrezzo = rapportoSinistro$
- Altrimenti: $punteggioGrezzo = rapportoDestro$

6. Applicazione funzione di scaling
   - $alpha = -400$
   - $beta = 100$
   - Calcola $valoreScalare = alpha \cdot (punteggioGrezzo - \frac{1}{2})^2 + beta$

Chapter 4

Rilevamento dello sguardo frontale tramite CNN

4.1 Selta del modello

Dopo un'attenta analisi sui modelli stato dell'arte nel task del rilevamento dello sguardo si è trovato come riferimento l'implementazione della rete neurale descritta all'interno del paper [AHK+23, L2CS-NET: FINE-GRAINED GAZE ESTIMATION IN UNCONSTRAINED ENVIRONMENTS]. Il modello descritto in questo paper si propone di prevedere lo sguardo in ambienti non vincolati, dando una stima degli angoli Pitch e Yaw del volto in radianti.

4.2 Utilizzo del modello

Il modello L2CS-NET viene distribuito sotto forma di pacchetto Python. Il pacchetto espone un API tramite la quale è possibile far processare un'immagine ed ottenere per ogni volto dell'immagine gli angoli Pitch e Yaw. Il modello L2CS-NET è costruito mediante il framework Pytorch ed è quindi un modello Pytorch, per poter utilizzare il modello all'interno del software OFIQ è necessario covnertire il modello al formato Onnx, un formato open utilizzato per rappresentare i modelli di machine learning. Il formato Onnx permettere di eseguire il modello all'interno del software OFIQ in C++ tramite la libreria OnnxRuntime disponi-

bile in C++. Inoltre il pacchetto Python di L2CS-NET esegue il preprocessa dell'immagine tramite una pipeline che effettua i sueguenti step:

- Utilizza il modello RetinaFace (dalla libreria face detection) per trovare il bounding box dei vari volti all'interno dell'immagine
- Per ogni volto viene presa in considerazione solamente la parte dell'immagine compresa all'interno della bounding box (regione di interesse)
- Per ogni regione di interesse viene convertito lo spazio colori da BGR a RGB
- Ogni regione di interesse viene ridimensionato in un'immagine (224 * 224)
- Ogni regione di interesse viene passata al modello ResNet-50 seguito da 2 layer fully-connected che ritornano in output 90 classi rappresentanti intervalli discreti (sia per pitch che per yaw) con all'interno un valore rappresentante la probabilità che lo sguardo per il relativo angolo si trovi in quell'intervallo
- Per ogni valore ritornato da ResNet-50 viene applicata la funzione Softmax di modo da avere la probabilità in percentuale che lo sguardo si trovi all'interno di quell'intervallo
- Vengono moltiplicate le probabilità che lo sguardo si trovi in quell'intervallo per l'indice della probabilità nel vettore dei tensori
- Viene fatta la somma dei valori ottenuti precedentemente
- La somma viene moltiplicata per quattro ed il risultato sottratto di 180, in modo da convertire ogni probabilità ottenuta gradi.
- Viene eseguita la conversione da gradi a radianti

I passaggi che includono l'utilizzo di open cv sono facilmente replicabili all'interno del software OFIQ tramite la libreria open cv per cpp, mentre per trovare il bounding box del volto all'interno dell'immagine viene utilizzato il modello "SSD Face Detector CNN" che riporta risultati molto simili a quelli di RetinaFace, la scelta di utilizzare questo modello è dettata dal fatto di averlo già a disposizione all'interno del software in formato Onnx.

4.3 Implementazione della metrica

4.3.1 Dati di Input

- immagine: L'immagine del volto da valutare

4.3.2 Dati di Output

- valoreScalare: valore numerico che rappresenta la deviazione dello sguardo da 0 a 100

4.3.3 Algoritmo

1. Calcola $detectedFaces$ tramite il modello SSD Face Detector CNN
2. Se $detectedFaces.length < 1$ allora: $valoreScalare = 0$. Fine.
3. $faceBoundingBox = detectedFaces[0]$, solamente il primo volto trovato verrà valutato
4. Carica il modello in formato onnx tramite onnxRuntime
5. Calcola $immagineVolto$ = la sezione di immagine all'interno di $faceBoundingBox$
6. Preprocessing per L2CS-NET
   - Definisce $inputSize = 443$
   - Calcola $immagineRidimensionata$ ridimensionando con open cv $immagineVolto$ alla grandezza ($inputSize$ x $inputSize$)
   - Converte lo schema colore di $immagineRidimensionata$ da BGR a RGB
   - Scala i valori dell'immagine dall'intervallo [0,255] a [0,1] convertendoli a float
   - Normalizza l'immagine
     - Definisce $mean = [0.485f, 0.456f, 0.406f]$ (dal preprocessing della librerie L2CS-NET)

  - Definisce $std = [0.229f, 0.224f, 0.225f]$ (dal preprocessing della librerie L2CS-NET)
  - Per ogni canale RGB dell'immagine: $channels[i] = \frac{(channels[i] - mean[i])}{std[i]}$ con $i = 0..3$
  - Definisce $immaginePreprocessata$ la matrice formata dai canali normalizzati nei punti precedenti
- Definisce $input_shape = [1, 3, inputSize, inputSize]$ (aggiunge la dimensione della batch (in questo caso 1 dato che viene processata solamente un'immagine))
- Converte l'input in una matrice CHW:
  - Per ogni canale $c = 0..3$:
  - Per ogni riga dell'immagine $h = 0...inputSize$
  - Per ogni colonna dell'immagine $w = 0...inputSize$
  - $tensoreInput[c * inputSize * inputSize + h * inputSize + w] = immaginePreprocessata[h][w][c]$

7. Vengono usate le API offerte da *onnxRuntime* per eseguire il modello L2CS in formato Onnx.

8. Vengono calcolati *pitch* e *yaw* sulla base dell'output del modello (il modello restituisce in output i valori per 90 classi sia per pitch che per yaw, i seguenti step vengono eseguiti per entrambi gli angoli)

9. - Sia x il vettore con i 90 valori
   - $angoloInRadianti = \frac{\sum_i^x (e^{x_i} * i)}{\sum_i^x (e^x)}$ * 4 - 180

10. $value = max(|pitch|, |yaw|)$ (Si usa il valore assoluto in quanto lo sguardo frontale viene classificato come angolo 0)

11. $score = round((1 - (\frac{min(value, 45)}{45})) * 100)$ (Uso 45 come valore limite, tutti valore da 45 a 180 vengono classificati come punteggio 0)

12. value è il valore assoluto dell'angolo più lontano da 0 Gradi, mentre score è la valutazione dell'immagine da 0 a 100

Chapter 5

Rilevamento dello sguardo frontale tramite Machine Learning

L'utilizzo di metodi algoritmici basati sulle coordinate dei landmark e l'utilizzo di modelli in grado di predire l'angolazione dello sguardo sono due ottimi metodi per cercare di comprendere se il volto analizzato sta guardando in fotocamera, ma ci sono ulteriori approcci che sono degni di nota, in particolare questo capitolo è dedicato a due ulteriori tentativi effettuati per comprendere se un volto ha lo sguardo diretto verso la camera.

5.1 Fine-Tuning di modelli CNN

In questo capitolo viene approfondito il metodo che sceglie di fare utilizzo di Reti neurali convoluzionali già preaddestrate come classificatori per poi eseguire la tecnica del Fine Tuning in modo da riaddestrare i modelli a predire se un volto ha o no lo sguardo diretto verso la camera. Per eseguire i seguenti test è stato necessario l'utilizzo di dateset di grandi dimensioni e con dati reali, per cui la scelta è ricaduta sul dataset Gaze Direction Detection disponibile sulla piattaforma Kaggle. Il dataset "Gaze Direction Detection" contiene 53000 coppie di occhi ed è rilasciato sotto forma di array binario numpy (.npz) contenente all'interno 3 array:

- array `targets`: contiene valori 0 e 1, dove il valore *i-esimo* indica che l'immagine *i-esima* guarda in camera (se = 1) o altrove (se = 0)
- array `right_eyes`: Immagini raffiguranti gli occhi destri
- array `left_eyes`: Immagini raffiguranti gli occhi sinistri

Ogni immagine del dataset è un'immagine di dimensioni $64 \cdot 64$ raffigurante l'occhio ed una parte del volto che in alcuni casi arrivava fino a raffigurare una piccola parte del naso. Il dataset è stato successivamente diviso tra dataset di training e dataset di test (40000 immagini di training set e 4000 di test set). Training set e Test set contengono inoltre lo stesso numero di immagini tra sguardo in camera e sguardo non in camera. Il Fine-Tuning è stato eseguito sul modello MobileNetV2, la cui architettura è descritta nel Paper [SHZ+18, MobileNetV2: Inverted Residuals and Linear Bottlenecks]. Al modello MobileNetV2 è stato sostituito il layer di output con un layer Dense di 2 nodi con la funzione di attivazione "softmax". Il modello è stato successivamente addestrato utilizzando "Adam" come funzione di ottimizzazione e come funzione di loss la funzione "categorical_crossentropy". Sono inoltre stati seguiti i seguenti passagi per eseguire il training della rete:

- Tecnica dell'early stop: tecnica che consiste nel monitorare la funzione di loss e di interrompere l'addestramento quando per n epoche di fila non si riscontrano miglioramenti per poi utilizzare gli ultimi pesi in cui si è riscontrato un miglioramento
- Eseguire un primo ciclo di addestramento dei soli ultimi strati congelando gli altri
- Eseguire l'addestramento della rete sul dataset

5.1.1 Fine-Tuning di MobileNet V2 versione 1

Nella versione 1 del modello di Fine-Tuning di MobileNet V2, il modello è stato addestrato utilizzando le immagini del dataset "Gaze Direction Detection".
Il modello sul test set riporta un'accuracy del 74.0%.
La matrice di confusione riportata è la seguente: Figure 5.1.

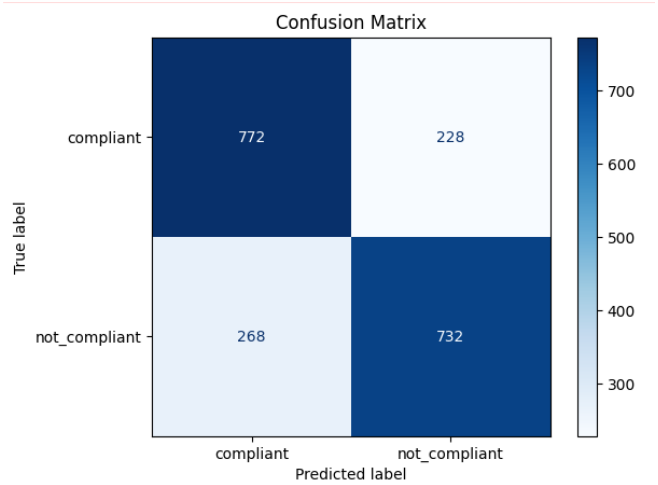


Figure 5.1: Matrice di confusione del fine tuning del modello mobilenetv2 versione 1

Per riuscire integrare il modello all'interno di OFIQ è stato necessario esportare il modello in formato Onnx e definire una pipeline di preprocessing delle immagini all'interno del software OFIQ.La pipeline presenta i seguenti step:

- Viene costruito un bounding box rettangolare che racchiuda i landmark della rete adnet attorno all'occhio
- Viene trovato il lato maggiore del bounding box e moltiplicato per un "fattore di scala".
- Viene ridisegnato il bounding box rettangolare attorno alla nuova dimensione del lato maggiore mantenendo l'occhio al centro.
- Viene ridimensionata l'immagine alle dimensioni originali richieste da MobileNetV2: $224 \cdot 224$
- Viene data l'immagine in pasto al modello.
- L'output del modello sono 2 tensori dove il tensore 0 indica la percentuale di probabilità che ha l'immagine di avere sguardo frontale, mentre il tensore 1 la percentuale contraria.

5.1.2 Fine-Tuning di MobileNetV2 versione 2

Dato che il modello precedentemente introdotto 5.1.1 portava buoni score sul test set ma mostrava problemi in fase di generalizzazione una volta sottoposti al modello immagini relative ad ulteriori dataset, nella versione 2 del modello si è intervenuti sulla capacità di generalizzazione del dataset in particolare cercando di far valutare al modello il solo occhio dalle immagini del dataset iniziali. E' stata quindi definita il seguente preprocessing per tutte le immagini del dataset:

- utilizzo del classificatore HaarCascade di OpenCV per identificare la posizione degli occhi nelle immagini del dataset
- Il classificatore restituisce il bounding box dell'occhio che viene ritagliato dall'immagine originale
- L'immagine ottenuta viene ridimensionata alla dimensione $96 \cdot 96$
- Viene aggiunto un bordo nero fino a far arrivare l'immagine alla dimensione richiesta da MobileNetV2 ($224 \cdot 224$), questo viene fatto per non far ridimensionare troppo l'immagine e non aggiungere eccessivo rumore che potrebbe portare il modello a non generalizzare bene su immagini con risoluzioni diverse

Il modello sul test set riporta un'accuracy del 78.4%.
La matrice di confusione riportata è la seguente: Figure 5.2.
Per riuscire ad utilizzare il modello su OFIQ è stata definita una pipeline di preprocessing analoga a quella utilizzata sul dataset per la versione 2 del modello.

5.2 Costruzione di un modello CNN

Per riuscire a minimizzare l'impatto del rumore presente durante il ridimensionamento delle immagini e di particolari tecniche di preprocessing è stato implementato un modello CNN from scratch con la differenza rispetto a MobileNetV2 di accettare immagini di dimensione ridotta: $96 \cdot 96$. La struttura del modello è descritta nell'immagine Figure 5.4. Contiene 4 strati in cui si alternano Layer di convoluzioni e layer di max pooling che dimezzano la dimensione dei batch fino ad

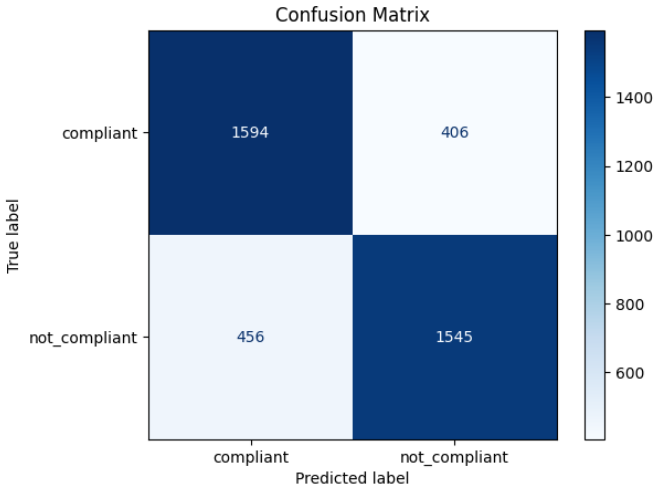


Figure 5.2: Matrice di confusione del fine tuning del modello mobilenetv2 versione 2

arrivare al layer finale con una dimensione $6 \cdot 6 \cdot 256$. La parte finale è una rete fully connected con 512 nodi ed un layer di Dropout per favorire l'addestramento, fino all'ultimo layer con le 2 classi finali e la "softmax" come funzione di attivazione. Per l'addestramento del modello sono state utilizzate le seguente tecniche:

- "Adam" come funzione di ottimizzazione.
- "sparse_categorical_crossentropy" come funzione di loss.
- la tecnica dell'Early stop.

Il modello è stato addestrato e testato sullo stesso dataset utilizzato per il Fine Tuning dei precedenti modelli "Gaze Direction Detection" ed è stato successivamente convertito in formato Onnx per permetterne l'esecuzione all'interno del software OFIQ. La pipeline utilizzata sulle immagini prima di essere date in pasto al modello è la seguente:

- utilizzo del classificatore HaarCascade di OpenCV per identificare la posizione degli occhi nelle immagini del dataset
- Il classificatore restituisce il bounding box dell'occhio che viene ritagliato dall'immagine originale

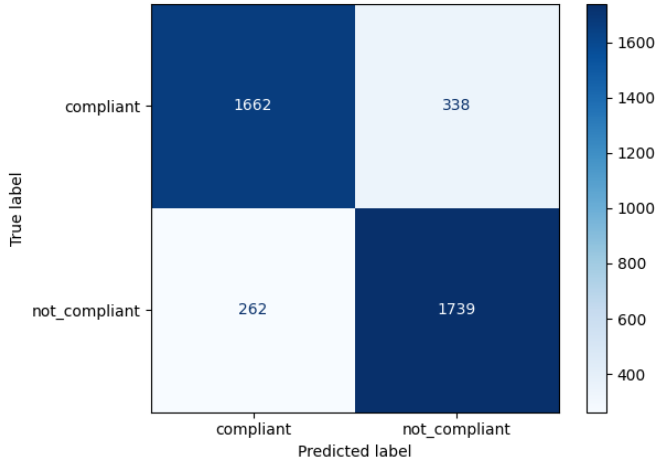


Figure 5.3: Matrice di confusione del modello CNN from scratch

- L'immagine ottenuta viene ridimensionata alla dimensione $96 \cdot 96$

Il modello sul test set riporta un'accuracy dell'85.0%. La matrice di confusione riportata è la seguente: Figure 5.3

5.3 Costruzione di modelli di machine learning basati sui landmark facciali

Dati i problemi di generalizzazione riscontrati nei precedenti modelli basati su CNN basati sui risultati ottenuti su dataset differenti, viene costruita una soluzione con il solo utilizzo di CNN capaci di estrarre i Landmark facciali, per poi addestrare modelli che possano predire lo sguardo in camera basati sulle posizioni dei landmark. I modelli sviluppati mantengono tutti la stessa struttura cambiando però i componenti di riferimento, la struttura presenta i seguenti componenti:

- estrattore di landmark: Un modello in grado di estrarre i landmark degli occhi partendo dall'immagine di un volto, i componenti testati sono i seguenti:
  - MediaPipe Face Mesh: MediaPipe è una suite di librerie e strumenti open source sviluppati da Google, Face Mesh è l'estrattore di landmark del volto di MediaPipe. MediaPipe Face Mesh è in grado di estrarre 20

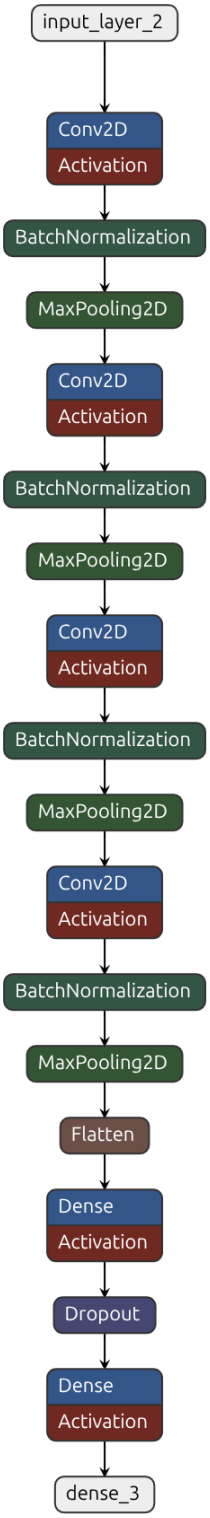


Figure 5.4: Struttura della CNN implementata

landmark per occhio che riescono a descrivere la posizione dell'occhio e dell'iride

  - ADNet: modello per estrarre landmark da un volto, è in grado di estrarre 9 landmark per occhio.

- pandas dataframe: I landmark estratti vengono inseriti in un pandas dataframe.
- modello predittivo:
  - XGBoost (eXtreme Gradient Boosting): libreria open source di apprendimento automatico basata sull'algoritmo gradient boosting
  - Rete neurale fully connected in Keras: rete neurale formata da un layer di input con un neurone per ogni colonne del dataset, un hidden layer composto da 128 neuroni ed un layer finale con singolo neurone e con funzione di attivazione "sigmoid".

Per l'addestramento di entrambi i modelli è stata utilizzata la tecnica del KFold cross validation. Inoltre i landmark sono stati presi per entrambi gli estrattori in forma normalizzata per favorire la capacità di generalizzazione del modello. Di seguito verranno riportati i risultati ottenutti per ogni combinazione di componenti. Il training set ed il test set dei modelli sono composti da immagini appartenenti ai dataset ONOT per le immagini con sguardo in camera e TONO per le immagini non in camera, i dataset contengono un totale di circa 1000 immagini. Non è stato possibile utilizzare il dataset "Gaze Direction Detection" in quanto le immagini contenute al suo interno non erano supportate dagli estrattori di landmark perchè non contenevano l'immagine dell'intero volto. I modelli sono stati ulteriormente testati anche su un sotto dataset estratto da Biolab di immagini reali tutte con lo sguardo frontale.

5.3.1 Estrattore landmark MediePipe Face Mesh con modello XGBoost

Accuracy su test set: 77.0%
Equal Error Rate su test set: 26.5%

Accuracy su sotto dataset Biolab: 63.2%
Matrice di confusione su test set: fig. 5.5a

5.3.2 Estrattore landmark MediePipe Face Mesh con modello Keras

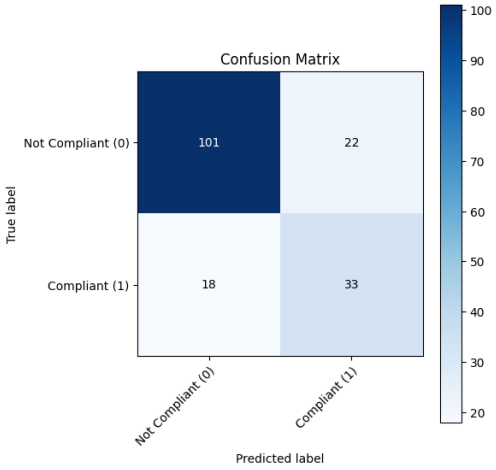
Accuracy su test set: 64.0%
Equal Error Rate su test set: 50.0%
Accuracy su sotto dataset Biolab: 100%
Matrice di confusione su test set: fig. 5.5b

5.3.3 Estrattore landmark ADNet con modello XGBoost

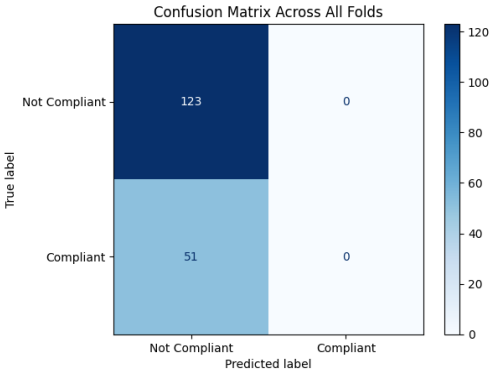
Accuracy su test set: 64.3%
Equal Error Rate su test set: 43.1%
Accuracy su sotto dataset Biolab: 12.1%
Matrice di confusione su test set: fig. 5.5c

5.3.4 Estrattore landmark ADNet con modello Keras

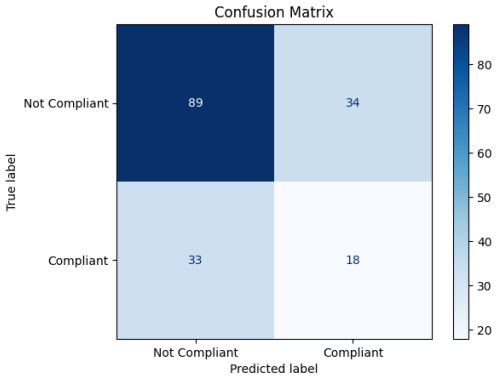
Accuracy su test set: 69.5%
Equal Error Rate su test set: 50.0%
Accuracy su sotto dataset Biolab: 0.0%
Matrice di confusione su test set: fig. 5.5b



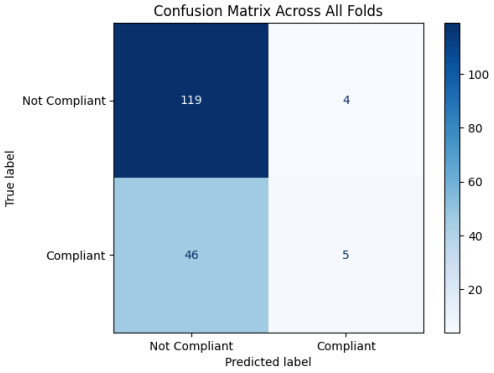
(a) Matrice di confusione del modello con MediePipe e XGBoost



(b) Matrice di confusione del modello con MediePipe e Rete Keras



(c) Matrice di confusione del modello con ADNet e XGBoost



(d) Matrice di confusione del modello con ADNet e Keras

Figure 5.5: Matrice di confusione dei modelli basati sui landmark

Chapter 6

Strumenti utilizzati per lo sviluppo delle metriche

6.1 Software OFIQ

Il software OFIQ, acronimo di *Open Source Face Image Quality*, è un progetto open source sviluppato per implementare le metriche descritte nel draft dello standard ISO 29794-5. È scritto in linguaggio C++ e dispone di file CMake per la compilazione su sistemi Linux, Windows e macOS. Al suo interno utilizza librerie come OpenCV per la gestione delle immagini e OnnxRuntime per l'esecuzione dei modelli Onnx. Inizialmente presentava un'implementazione fedele di tutte le metriche riportate nello standard ISO 29794-5, restituendo in output un valore grezzo ed un valore normalizzato tra 0 e 100 (utile come *score*) per ciascuna metrica. OFIQ include inoltre delle utility che permettono l'utilizzo dei seguenti modelli (in formato Onnx):

- SSD Face Detector CNN: modello per il rilevamento dei volti in un'immagine.
- ADNet: modello per l'estrazione dei landmark facciali.

Il software implementa le seguenti metriche:

- Quality Score: uno score generale dell'immagine ottenuto tramite il modello MagFace.

- Background uniformity: misura l'uniformità dello sfondo.
- Illumination uniformity: misura l'uniformità dell'illuminazione confrontando la parte destra e sinistra del volto.
- Luminance mean: verifica che l'immagine presenti un'illuminazione adeguata e uniforme.
- Luminance variance: misura i contrasti nell'immagine.
- Under-exposure prevention: verifica che l'immagine non contenga troppi pixel con bassa luminosità.
- Over-exposure prevention: verifica che l'immagine non contenga troppi pixel con alta luminosità.
- Dynamic range: misura la variazione di intensità luminosa nella regione del volto, assicurando che non sia completamente scura o chiara.
- Sharpness: valuta la nitidezza dell'immagine (corretta messa a fuoco).
- No compression artifacts: valuta la presenza di artefatti di compressione, assicurando che l'immagine non sia stata eccessivamente compressa.
- Natural colour: valuta la naturalezza del colore della pelle.
- Single face present: verifica la presenza di un singolo volto.
- Eyes open: controlla che entrambi gli occhi siano aperti in modo naturale.
- Mouth closed: verifica che la bocca sia chiusa.
- Eyes visible: verifica che siano visibili pupilla e iride in entrambi gli occhi.
- Mouth occlusion prevention: controlla che la bocca non sia occlusa.
- Face occlusion prevention: verifica che la regione del volto (dalla sommità del capo al mento e da orecchio a orecchio) sia chiaramente visibile.
- Inter-eye distance: misura in pixel la distanza tra gli occhi.

- Head size: valuta la dimensione del volto per evitare immagini troppo ravvicinate.
- Leftward crop of face: verifica che il volto non sia decentrato a sinistra.
- Rightward crop of face: verifica che il volto non sia decentrato a destra.
- Downward crop of face: verifica che il volto non sia decentrato in basso.
- Upward crop of face: verifica che il volto non sia decentrato in alto.
- Pose angle yaw frontal alignment: verifica che l'angolo *yaw* del volto sia inferiore a $\pm 5°$ dal frontale.
- Pose angle pitch frontal alignment: verifica che l'angolo *pitch* del volto sia inferiore a $\pm 5°$ dal frontale.
- Pose angle roll frontal alignment: verifica che l'angolo *roll* del volto sia inferiore a $\pm 8°$ dal frontale.
- Expression neutrality: verifica che il volto presenti un'espressione neutrale.
- No head covering: verifica che il soggetto non indossi copricapi (ad esempio un cappello).

6.2 Dataset utilizzati per la valutazione

I seguenti dataset sono stati utilizzati per la valutazione delle metriche implementate.

6.2.1 ONOT

ONOT è un dataset introdotto nel paper [DDBF+24, ONOT: a High-Quality ICAO-compliant Synthetic Mugshot Dataset]. È composto da immagini sintetiche di alta qualità conformi ai requisiti dello standard ISO/IEC 39794-5. Quest'ultimo definisce un formato per lo scambio di immagini facciali negli *electronic Machine-Readable Travel Documents* (eMRTD), seguendo le linee guida dell'International

Civil Aviation Organization (ICAO). Le immagini di ONOT includono volti di diverse etnie, età, generi e caratteristiche facciali, rendendo il dataset adatto alla valutazione delle metriche implementate. Per questa analisi sono state considerate le sole immagini del *Subset 1*, ICAO compliant.

6.2.2 TONO

TONO è un dataset introdotto nel paper [BFDDM24, TONO: a synthetic dataset for face image compliance to ISO/ICAO standard]. È costituito da immagini sintetiche di volti ad alta qualità, create per sviluppare e valutare sistemi di verifica della conformità delle immagini facciali allo standard ISO/ICAO. Le immagini di TONO derivano dal dataset ONOT ([DDBF$^+$24]), con l'aggiunta di uno o più elementi non conformi agli standard. I difetti presenti in TONO sono i seguenti:

- Head and Shoulder Pose: assenza di posa frontale sia del volto che delle spalle.
- Gaze Direction: assenza di sguardo frontale.
- Expression: mancanza di espressione neutrale o presenza di denti visibili.
- Face Illumination: non uniformità dell'illuminazione del volto.
- Background: sfondo non uniforme.
- Head Coverings: presenza di copricapi.
- Eye Visibility: occhi chiusi, presenza di occhiali (da vista o da sole), make-up eccessivo.
- Photographic: difetti quali pixelazione, posterizzazione, sfocatura, sovraesposizione, sovrasaturazione.

Per questa casistica è stata utilizzata la versione di TONO in cui ogni immagine contiene un solo elemento in contrasto con i requisiti ISO/ICAO.

6.3 Onnx e OnnxRuntime

ONNX (*Open Neural Network Exchange*) è un formato open source per la rappresentazione di modelli di machine learning, progettato per garantire compatibilità tra diversi framework (PyTorch, TensorFlow, ecc.). Permette di salvare un modello in un file `.onnx` indipendente dall'ambiente di training. La libreria OnnxRuntime fornisce API utilizzabili da diversi linguaggi di programmazione e su diverse piattaforme (inclusi i web browser). Il progetto, sviluppato da Microsoft, è disponibile al seguente repository GitHub: https://github.com/microsoft/onnxruntime.

6.4 FVC Ongoing

FVC Ongoing ([DCF+09]) è una piattaforma web che consente la valutazione di algoritmi di vario tipo, tra cui il task *Face Image ISO Compliance Verification*. I test vengono condotti su diversi dataset e metriche note.

6.5 Framework PyTorch

Il linguaggio di programmazione Python ed il framework PyTorch sono comunemente utilizzati per l'addestramento e la distribuzione di modelli di Machine Learning. Nel progetto il loro impiego riguarda principalmente il testing e la conversione in formato Onnx del modello L2CS-Net, originariamente distribuito in PyTorch. In particolare, è stato utilizzato il modulo `ONNX exporter API`, che consente di esportare un modello PyTorch in formato Onnx, impiegato per la conversione di L2CS-Net tramite il seguente codice: section 6.5

6.6 Jupyter Notebook, Numpy e Matplotlib

Jupyter Notebook, Numpy e Matplotlib sono stati utilizzati per creare documenti interattivi e rappresentare graficamente (tramite boxplot e istogrammi) le performance delle metriche di valutazione sui dataset ONOT, TONO e su un sottoinsieme di Biolab. Il linguaggio Python è stato inoltre impiegato per:

```python
import torch
from l2cs import Pipeline
from pathlib import Path

def export_l2cs_to_onnx():

    gaze_pipeline = Pipeline(
        weights=Path("./L2CSNet_gaze360.pkl"), # pretrained model weight
        arch='ResNet50',
        device=torch.device('cpu')
    )

    model = gaze_pipeline.model

    dummy_input = torch.randn(1, 3, 448, 448)

    torch.onnx.export(
        model,
        dummy_input,
        "l2cs_gaze.onnx",
        export_params=True,
        opset_version=11,
        do_constant_folding=True,
        input_names=['input'],
        output_names=['pitch', 'yaw']
    )
    print("Model exported to l2cs_gaze.onnx")

if __name__ == "__main__":
    export_l2cs_to_onnx()
```

- suddividere i dataset in base ai file di riferimento in formato `.txt`;
- elaborare le metriche per valutare il software OFIQ e gli algoritmi proposti.

Chapter 7

Valutazione delle metriche

7.1 Introduzione alle valutazioni

Per quanto riguarda le valutazioni del software OFIQ, verranno riportati ed analizzati esclusivamente i risultati delle metriche implementate nelle loro diverse versioni. In particolare, per ogni metrica saranno presi in esame l'`Equal Error Rate` e la distribuzione dei risultati.

7.2 Valutazione su FVC-Ongoing

7.2.1 Metriche valutate

La piattaforma FVC-Ongoing valuta 24 diverse metriche table 7.1. Nel caso di questo progetto, le metriche selezionate sono:

- Looking Away: valuta se un soggetto sta guardando in camera.
- Red Eyes: valuta la presenza del difetto degli occhi rossi.

7.2.2 Il protocollo

Per la sottomissione degli algoritmi sulla piattaforma FVC-Ongoing è necessario rispettare il seguente protocollo:

N°	Description of the test
Feature extraction accuracy tests	
1	Eye center location accuracy
Photographic and pose-specific tests	
2	Blurred
3	Looking away
4	Ink marked/creased
5	Unnatural skin tone
6	Too dark/light
7	Washed out
8	Pixelation
9	Hair across eyes
10	Eyes closed
11	Varied background
12	Roll/pitch/yaw > predefined thresholds
13	Flash reflection on skin
14	Red eyes
15	Shadows behind head
16	Shadows across face
17	Dark tinted lenses
18	Flash reflection on lenses
19	Frames too heavy
20	Frame covering eyes
21	Hat/cap
22	Veil over face
23	Mouth open
24	Other faces/toys too close

Table 7.1: Metriche valutate da FVC-Ongoing

- inviare una cartella compressa in formato ZIP;
- all'interno deve essere presente un file *Check.exe*, eseguibile per Win32 in formato *console application*;
- la sintassi da riga di comando deve essere: `./Check.exe <faceimagefile> <outputfile>`, dove:
  - `faceimagefile`: percorso dell'immagine del volto da valutare (formati supportati: BMP, JPG, PNG);

  - `outputfile`: percorso del file TXT di output, su cui scrivere (in modalità *append*) l'esito dei test.

- Ogni riga del file di output deve contenere i seguenti campi separati da spazi:
  - `ImageName`: nome del file immagine;
  - `RetVal`: intero che indica se l'immagine può essere processata:
    - 1: immagine processabile;
    - 0: immagine non processabile;
    - -1: dimensione dell'immagine non supportata;
    - -2: formato immagine non supportato;
    - -3: contenuto non processabile.
  - `LE_x`, `LE_y`: coordinate X e Y del centro dell'occhio sinistro (in pixel);
  - `RE_x`, `RE_y`: coordinate X e Y del centro dell'occhio destro (in pixel);
  - `Test_2`: intero compreso tra 0 e 100 indicante il grado di conformità rispetto al test 2 (0 = non conforme, 100 = massimo livello di conformità). In alternativa, può assumere i valori:
    - `-`: se la metrica non è supportata dal programma;
    - `?`: se la metrica è supportata ma non valutabile nell'immagine corrente per un motivo specifico;
    - `!`: se la metrica è supportata ma non valutabile nell'immagine corrente per motivo indefinito.
  - ...
  - `Test_24`: analogo al campo `Test_2`, relativo al test 24.

- L'eseguibile deve avere permessi di scrittura solo sul file di output, mentre i file di configurazione possono essere caricati esclusivamente in lettura.

Dopo un'attenta analisi del protocollo richiesto dalla piattaforma FVC-Ongoing, è stato necessario rendere il software OFIQ conforme a tali specifiche ed adattare le metriche implementate.

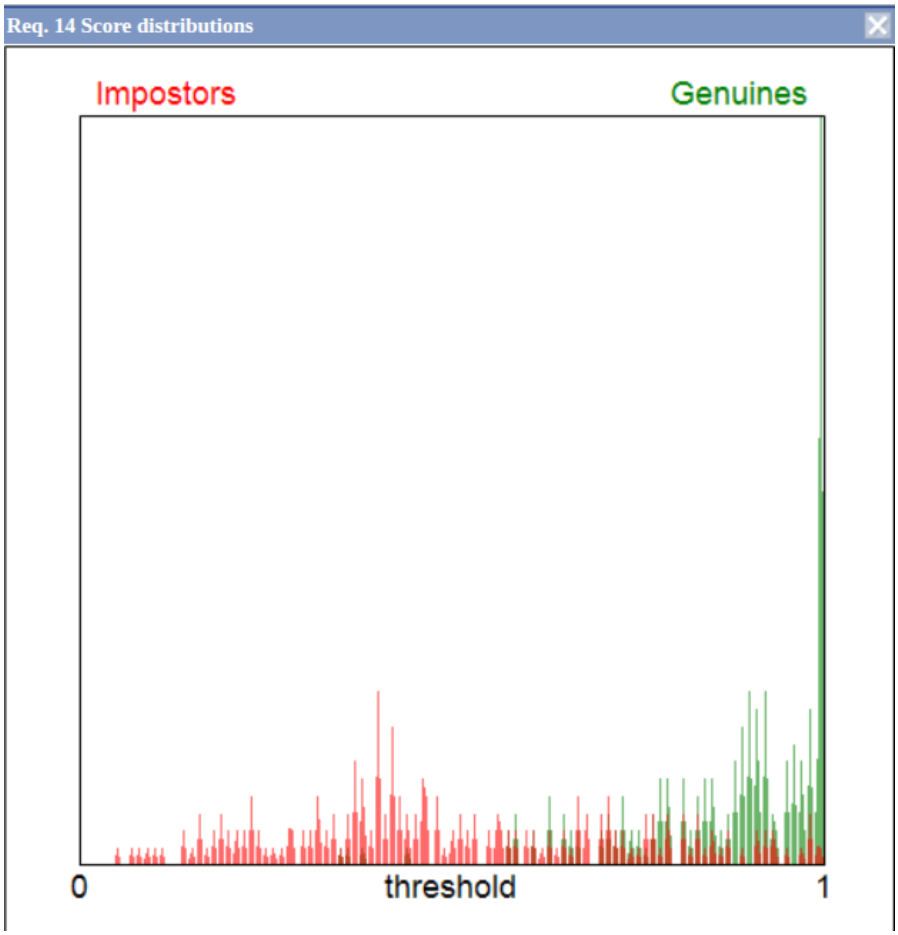


Figure 7.1: Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione HSV)

7.2.3 Valutazione metrica per il rilevamento del difetto degli occhi rossi (versione HSV)

La valutazione della metrica per il difetto degli occhi rossi è divisa in due parti. Nella prima parte vengono analizzati i risultati ottenuti tramite l'algoritmo basato sullo spazio colore HSV, mentre nella seconda parte si utilizza lo spazio colore yCbCr.

`Equal Error Rate`: 18.1%.
La distribuzione degli score è riportata nel grafico Figure 7.1.

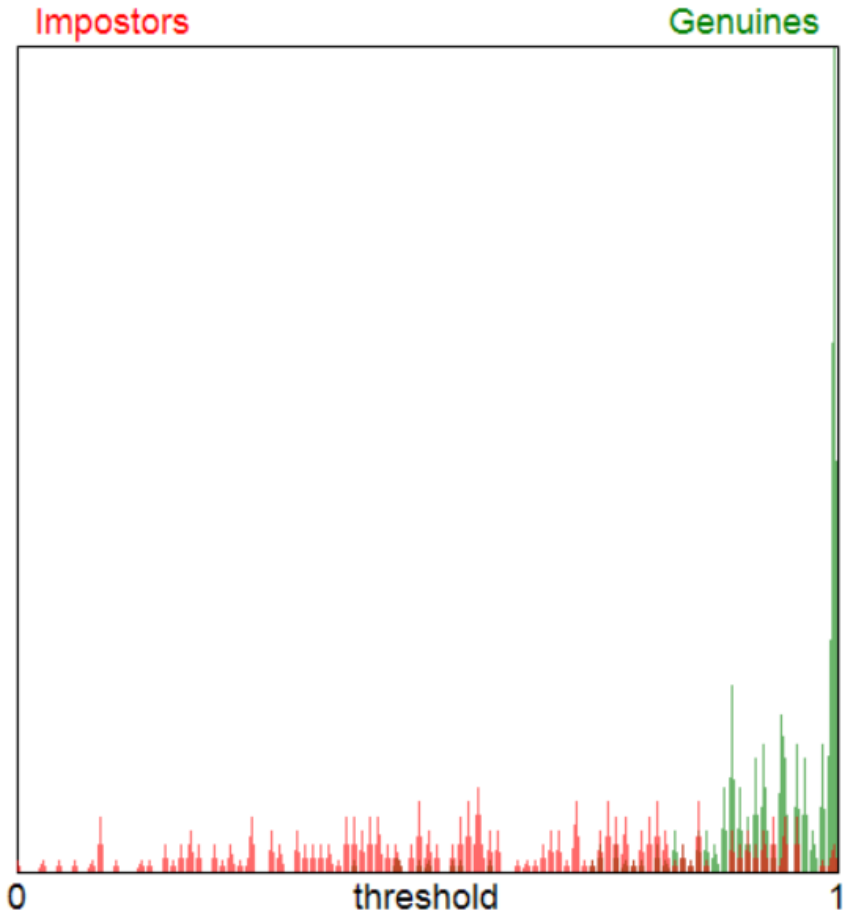


Figure 7.2: Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione yCbCr)

7.2.4 Valutazione metrica per il rilevamento del difetto degli occhi rossi (versione yCbCr)

La versione dell'algoritmo che utilizza lo spazio colore yCbCr ottiene risultati migliori rispetto alla versione HSV.

`Equal Error Rate`: 14.6%.
La distribuzione degli score è riportata nel grafico Figure 7.2.

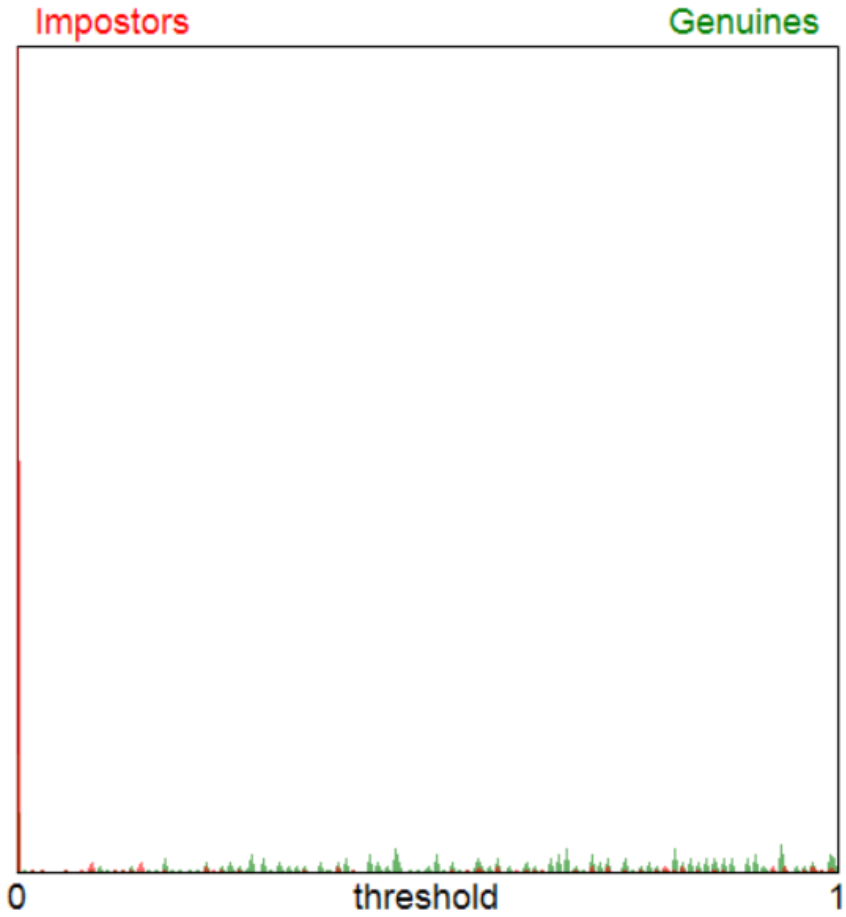


Figure 7.3: Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (metodo algoritmico)

7.2.5 Valutazione metrica per il rilevamento dello sguardo frontale (metodo algoritmico)

I risultati ottenuti tramite il metodo algoritmico rientrano nella media degli algoritmi pubblicati sulla piattaforma.

`Equal Error Rate`: 17.1%.
La distribuzione degli score è riportata nel grafico Figure 7.3.

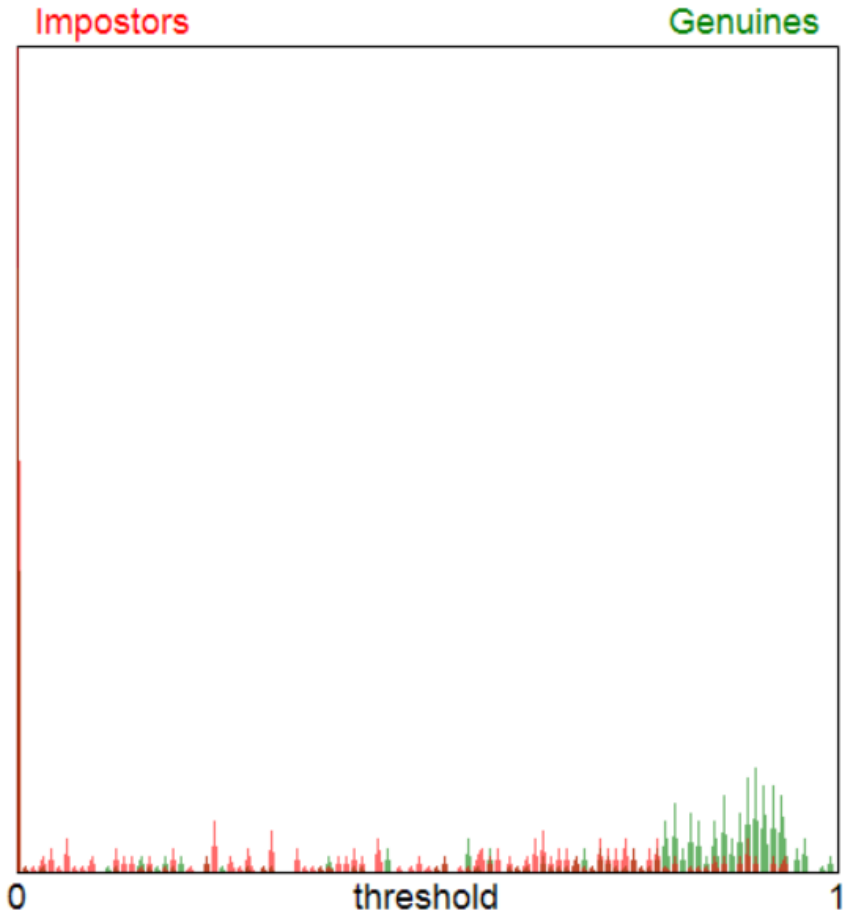


Figure 7.4: Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (versione L2CS-Net)

7.2.6 Valutazione metrica per il rilevamento dello sguardo frontale (metodo L2CS-Net)

L'utilizzo della CNN L2CS-Net, in grado di predire l'angolatura dello sguardo, si è rivelato meno efficace del metodo algoritmico.

`Equal Error Rate`: 36.0%.
La distribuzione degli score è riportata nel grafico Figure 7.4.

7.3 Valutazione su dataset sintetici

7.3.1 Dataset utilizzati

I dataset utilizzati per questa valutazione sono ONOT e TONO (già introdotti in precedenza). TONO fornisce immagini non conformi per i singoli difetti analizzati, mentre ONOT fornisce immagini conformi.

7.3.2 Metriche valutate

Nei dataset TONO e ONOT non è possibile valutare il difetto degli occhi rossi in quanto assente. La valutazione verrà effettuata sul solo difetto dello sguardo frontale.

7.3.3 Procedimento di valutazione

La valutazione delle metriche sui dataset ONOT e TONO è stata condotta secondo le seguenti fasi:

1. **Filtraggio delle immagini in ONOT**: tramite i file TXT forniti con il dataset, sono state mantenute solo le immagini conformi allo standard ICAO.
2. **Valutazione delle metriche in TONO**: ogni cartella di TONO e le rimanenti immagini di ONOT sono state valutate dal software OFIQ, generando file CSV contenenti, per ciascuna immagine, gli score assegnati alle metriche.
3. **Analisi dei risultati**: le valutazioni delle immagini di TONO sono state confrontate con quelle di ONOT. Per ciascuna sottocartella di TONO, i risultati nella metrica relativa al difetto rappresentato sono stati messi in relazione con i corrispondenti risultati di ONOT.
4. **Visualizzazione dei risultati**: per ogni metrica valutata, i risultati sono stati rappresentati con:
    - `BoxPlot`: un confronto diretto tra distribuzioni di TONO e ONOT;
    - `Distribuzione degli score`: un istogramma che confronta la distribuzione degli score (rosso = TONO, verde = ONOT);

- `Equal Error Rate`: misura dell'equilibrio tra falsi positivi e falsi negativi.

7.3.4 Valutazione metrica per il rilevamento dello sguardo frontale (metodo algoritmico)

Il dataset TONO contiene due cartelle con immagini relative al difetto dello sguardo frontale:

- *la_1*: immagini di volti con lo sguardo rivolto verso destra;
- *la_2*: immagini di volti con lo sguardo rivolto verso sinistra.

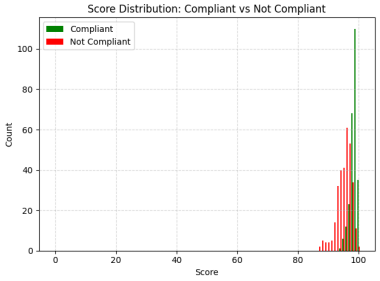
Risultati per la cartella *la_1*:
`Equal Error Rate`: 15.8%.
Distribuzione degli score: Figure 7.5a.
BoxPlot comparativo: Figure 7.5b.

Risultati per la cartella *la_2*:
`Equal Error Rate`: 11.1%.
Distribuzione degli score: Figure 7.5c.
BoxPlot comparativo: Figure 7.5d.

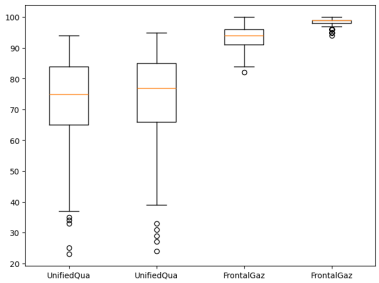
7.3.5 Valutazione metrica per il rilevamento dello sguardo frontale (metodo L2CS-Net)

Come introdotto in precedenza, i risultati sono riportati separatamente per le cartelle *la_1* e *la_2*.
Risultati per la cartella *la_1*:
`Equal Error Rate`: 18.8%.
Distribuzione degli score: Figure 7.6a.
BoxPlot comparativo: Figure 7.6b.

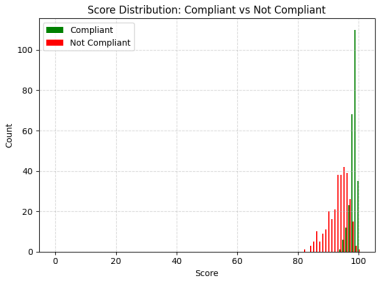
Risultati per la cartella *la_2*:
`Equal Error Rate`: 18.6%.



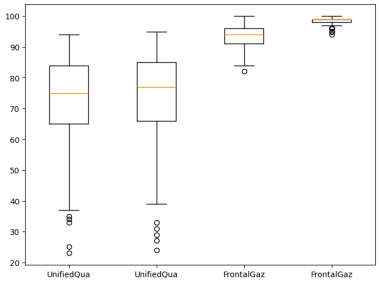
(a) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la_1) e ONOT



(b) BoxPlot comparativo per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la_1) e ONOT



(c) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la_2) e ONOT

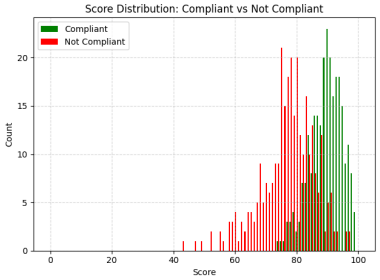


(d) BoxPlot comparativo per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la_2) e ONOT

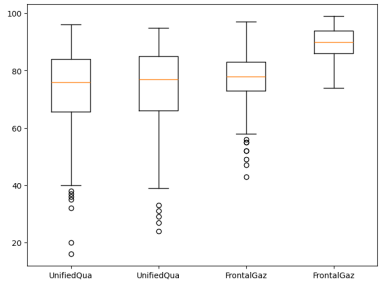
Figure 7.5: Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo algoritmico)

Distribuzione degli score: Figure 7.6c.

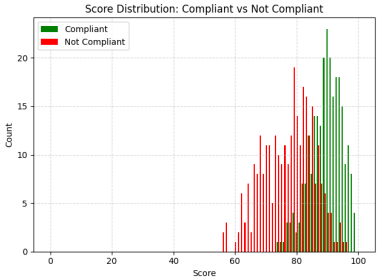
BoxPlot comparativo: Figure 7.6d.



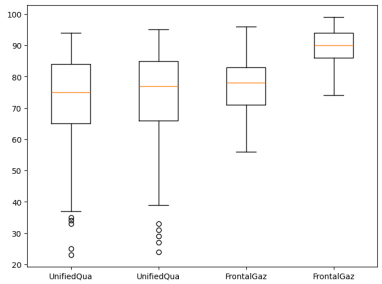
(a) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo L2CS-Net), dataset TONO (cartella la_1) e ONOT



(b) BoxPlot comparativo per il rilevamento dello sguardo frontale (metodo L2CS-Net), dataset TONO (cartella la_1) e ONOT



(c) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo L2CS-Net), dataset TONO (cartella la_2) e ONOT



(d) BoxPlot comparativo per il rilevamento dello sguardo frontale (metodo L2CS-Net), dataset TONO (cartella la_2) e ONOT

Figure 7.6: Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo L2CS-Net)

7.3.6 Valutazioni dei modelli di machine learning sviluppati

Sono riportati di seguito Equal Error Rate e distribuzione degli score su TONO e ONOT dei modelli ottenuti tramite Fine Tuning di Mobile Net V2.

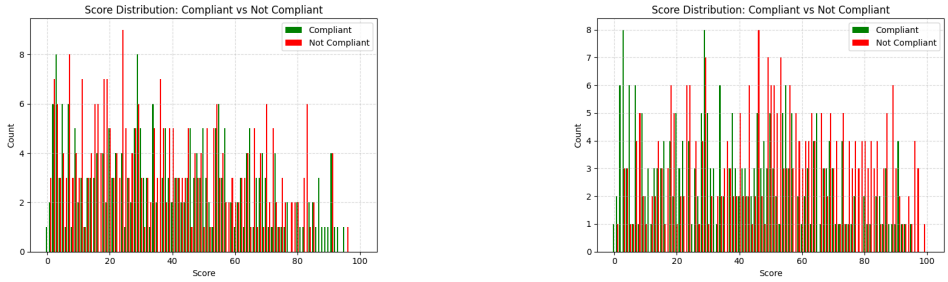
Versione 1: Fine tune di Mobile Net V2 sul dataset “Gaze Direction Detection” versione 1 (5.1.1)

Dataset `la_1` di Tono:

Equal Error Rate: 47.3%
Score Distribution: Figure 7.7a Dataset `la_1` di Tono:
Equal Error Rate: 58.5%
Score Distribution: Figure 7.7b

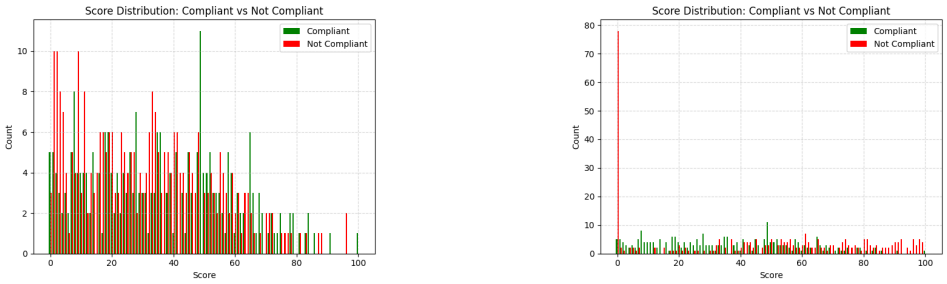
Versione 2: Fine tune di MObile Net V2 sul dataset "Gaze Direction Detection" versione 2 (5.1.2)
Dataset `la_1` di Tono:
Equal Error Rate: 46.9%
Score Distribution: Figure 7.8a Dataset `la_1` di Tono:
Equal Error Rate: 58.0%
Score Distribution: Figure 7.8b

Versione 3: Modello CNN allenato sul dataset "Gaze Direction Detection" (5.2)
Dataset `la_1` di Tono:
Equal Error Rate: 38.4%
Score Distribution: Figure 7.9a Dataset `la_1` di Tono:
Equal Error Rate: 21.8%
Score Distribution: Figure 7.9b



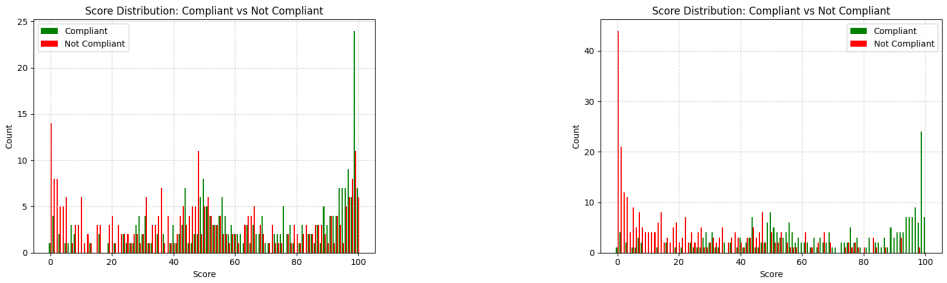
(a) Comparazione con cartella la_1 di Tono (b) Comparazione con cartella la_2 di Tono

Figure 7.7: Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo Fine Tuning Mobile Net v2, versione 1) su TONO e ONOT



(a) Comparazione con cartella la_1 di Tono (b) Comparazione con cartella la_2 di Tono

Figure 7.8: Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo Fine Tuning Mobile Net v2, versione 2) su TONO e ONOT



(a) Comparazione con cartella la_1 di Tono (b) Comparazione con cartella la_2 di Tono

Figure 7.9: Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo modello CNN) su TONO e ONOT

Chapter 8

Analisi dei risultati

8.1 Rilevamento del difetto degli occhi rossi

Nella metrica del rilevamento degli occhi rossi si è riscontrato un miglioramento delle performance nella versione che prevede l'utilizzo dello spazio colore YCBCR rispetto a quella con l'utilizzo dello spazio colore HSV, questo è dato dal fatto che il canale CR assume valori molto alti in presenza del colore rosso favorendone l'isolamento, inoltre l'informazione di luminanza è separata rispetto agli altri componenti rendendo il filtro più robusto a variazioni di luce/ombra. Un fattore che può aver inciso sull'equal error rate al 14% può essere la sottomissiona all'algoritmo di immagini riportanti diversi difetti oltre a quello degli occhi rossi, come dimostrato dalle immagini Section 8.1 che riportano score al 51 e 66 per colpa dell'errata saturazione dell'immagine, difetto riscontrabile da altre metiche implementate nel software OFIQ.

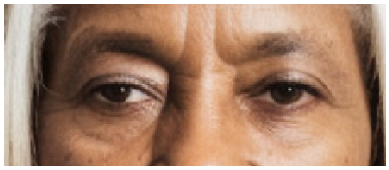
(a) Score: 51 (b) Score: 66

Figure 8.1: Esempi di immagini false positivi alla metrica del rilevamento degli occhi rossi dal dataset Biolab

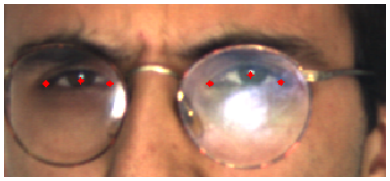
8.2 Rilevamento del difetto dello sguardo frontale

8.2.1 Versione algoritmica

Nella metrica del rilevamento dello sguardo frontale il metodo algoritmico basato sulla posizione dei landmark è quello che tra tutti i metodi in questa metrica ha rilevato gli score migliori dimostrando la difficolta per una modello di machine basato su CNN o su reti neurali nel riuscire a predire la direzione dello sguardo partendo dal volto dell'utente. Nonstante gli ottimi score realizzati si possono notare prestazioni migliori di questo algoritmi su immagini generate artificialmente presentanti nessun altro difetto specifico, i casi di TONO e ONOT, e prestazioni inferiori su immagini reali con potenzialmente altri difetti, come nel caso di FVC-Ongoing. Le debolezze di questo metodo includono immagini di volti nei quali i rapporti tra le distanze di punti specifi dell'occhio non sempre sono proporzionate (es. immagine classificata falsa negativi Figure 8.2c, immagine classificata falsa negativa Figure 8.2a, immagine classificata falsa positiva Figure 8.2d), la dipendenza dell'algoritmo rispetto all'errore del modello che genera i landmark il quale in casi in cui sono presenti riflessi vicino o sopra gli occhi tende a trovare landmark distanti di qualche pixel rispetto a dove dovrebbero essere portando ad eventuali score minori e le immagini le quali riportano altri difetti quali una posa non frontale del viso (es. falso negativo Figure 8.2b)

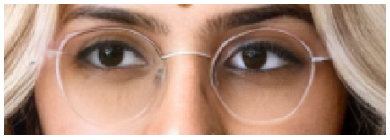


(a) Falso negativa, score: 94



(b) Falso negativa, score: 80

(c) Falso negativa, score: 0



(d) Falso positiva, score: 98

Figure 8.2: Esempi di immagini false positivi e fale negativi nella metrica del rilevamento dello sguardo frontale con metodo algoritmico

8.2.2 Versione con l'utilizzo del modello L2CS-Net

Il modello L2CS Net non riesce sempre a distinguere uno sguardo rivolto verso la camera ed uno leggermente laterale, le motivazione che possono portare ad un equal error rate sempre superiore al 18% potrebbero essere diverse. Una possibile causa è data dalla pipeline di preprocessing per la quale il modello l2cs si aspetta in input l'immagine di un volto rilevato dal modello *RetinaFace* disponibile all'interno della libreria python "face_detection", mentre per poter valutare il modello in OFIQ è stato utilizzato un altro modello di face detecting già presente in formato onnx all'interno di OFIQ. In particolare nell'immagine Figure 8.3b l2cs-net, analizzando l'angolo yaw, usando *RetinaFace* viene calcolato `0.03` radianti, mentre usando il modello all'interno di OFIQ viene calcolato `0.27` radianti, nonostante ciò, questo è un caso isolato, in quanto generalmente la differenza di predizione degli angoli nella media delle immagini è raramente superiore a `0.03` radianti. Un ulteriore causa potrebbe essere data dal dataset di addestramento, come possibile visualizzare sulla repository git il modello non è stato addestrato solamente su volti frontali, bensi su volti posti in diverse angolazioni e distanze, il che potrebbe aver inciso sulle prestazioni del modello in volti con il viso rivolto verso la camera, inoltre anche la qualità e il rumore presente all'interno delle immagini con cui è stato addestrato potrebbe aver inciso. Come possibile visualizzare nelle immagini Section 8.2.2, nonostante ad occhio nudo sia chiaro che l'immagine Figure 8.3a stia guardando lateralmente e che l'immagine Figure 8.3b non lo stia facendo, L2CS-Net prevede un angolo pitch maggiore nella seconda immagine rispetto alla prima, questo risultato è verificabile sia usando *RetinaFace* come face detector sia il face detector utilizzato nell'implementazione in OFIQ.

(a) Falso negativa, angolo pitch calcolato: 0.02

(b) Falso negativa, angolo pitch calcolato: 0.05

Figure 8.3: Esempi di immagini false positivi e fale negativi nella metrica del rilevamento dello sguardo frontale con metodo L2CS-Net

Chapter 9

Contribution

9.1 Fancy formulas here

Bibliography

[AHK+23] Ahmed A. Abdelrahman, Thorsten Hempel, Aly Khalifa, Ayoub Al-Hamadi, and Laslo Dinges. L2cs-net : Fine-grained gaze estimation in unconstrained environments. In *2023 8th International Conference on Frontiers of Signal Processing (ICFSP)*, pages 98–102, 2023.

[BFDDM24] Guido Borghi, Annalisa Franco, Nicolò Di Domenico, and Davide Maltoni. Tono: a synthetic dataset for face image compliance to iso/icao standard. In *The 18th European Conference on Computer Vision Workshops 2024*, 2024.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. pages 785–794, 08 2016.

[DCF+09] B. Dorizzi, R. Cappelli, M. Ferrara, D. Maio, D. Maltoni, N. Houmani, S. Garcia-Salicetti, and A. Mayoue. Fingerprint and on-line signature verification competitions at icb 2009. In *Proceedings of the International Conference on Biometrics (ICB)*, pages 725–732, Alghero, Italy, 2009.

[DDBF+24] Nicol'o Di Domenico, Guido Borghi, Annalisa Franco, Davide Maltoni, et al. Onot: a high-quality icao-compliant synthetic mugshot dataset. In *The 18th IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–6, 2024.

[SHZ+18] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and

linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

Acknowledgements

Optional. Max 1 page.