

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**

---

---

# 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'obiettivo di implementare un draft ISO, volto a valutare la qualità dell'acquisizione delle immagini facciali secondo diverse metriche.

---

---

# Contents

<b>Abstract</b>	<b>iii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Motivazioni . . . . .	1
1.2 Obiettivo . . . . .	3
1.3 Metodo . . . . .	3
1.4 Struttura della tesi . . . . .	3
<b>2 Implementazione della metrica per il rilevamento del difetto degli occhi rossi</b>	<b>5</b>
2.1 Obiettivo 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
<b>3 Implementazione della metrica per il rilevamento dello sguardo frontale</b>	<b>11</b>
3.1 Obiettivo 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
<b>4 Rilevamento dello sguardo frontale tramite CNN</b>	<b>15</b>
4.1 Scelta del modello . . . . .	15
4.2 Utilizzo del modello . . . . .	15

4.3	Implementazione della metrica . . . . .	16
4.3.1	Dati di Input . . . . .	16
4.3.2	Dati di Output . . . . .	17
4.3.3	Algoritmo . . . . .	17
<b>5</b>	<b>Rilevamento dello sguardo frontale tramite Machine Learning</b>	<b>19</b>
5.1	Fine-Tuning di modelli CNN . . . . .	19
5.1.1	Fine-Tuning di MobileNetV2 - versione 1 . . . . .	20
5.1.2	Fine-Tuning di MobileNetV2 - versione 2 . . . . .	21
5.2	Costruzione di un modello CNN gdd CNN . . . . .	22
5.3	Modelli di machine learning basati sui landmark facciali . . . . .	23
5.3.1	Risultati dei modelli basati su landmark . . . . .	25
<b>6</b>	<b>Strumenti utilizzati per lo sviluppo delle metriche</b>	<b>27</b>
6.1	Software OFIQ . . . . .	27
6.2	Dataset utilizzati per la valutazione . . . . .	29
6.2.1	ONOT . . . . .	29
6.2.2	TONO . . . . .	30
6.3	Onnx e OnnxRuntime . . . . .	31
6.4	FVC Ongoing . . . . .	31
6.5	Framework PyTorch . . . . .	31
6.6	Libreria Keras . . . . .	31
6.7	Libreria XGBoost . . . . .	33
6.8	Jupyter Notebook, Numpy e Matplotlib . . . . .	33
<b>7</b>	<b>Valutazione delle metriche</b>	<b>35</b>
7.1	Introduzione alle valutazioni . . . . .	35
7.2	Valutazione su FVC-Ongoing . . . . .	35
7.2.1	Metriche valutate . . . . .	35
7.2.2	Il protocollo . . . . .	35
7.2.3	Valutazione metrica per il rilevamento del difetto degli occhi rossi (versione HSV) . . . . .	38
7.2.4	Valutazione metrica per il rilevamento del difetto degli occhi rossi (versione yCbCr) . . . . .	39
7.2.5	Valutazione metrica per il rilevamento dello sguardo frontale (metodo algoritmico) . . . . .	39
7.2.6	Valutazione metrica per il rilevamento dello sguardo frontale (metodo L2CS-Net) . . . . .	39
7.3	Valutazione su dataset sintetici . . . . .	39
7.3.1	Dataset utilizzati . . . . .	39
7.3.2	Metriche valutate . . . . .	43

## CONTENTS

---

7.3.3	Procedimento di valutazione . . . . .	43
7.3.4	Valutazione metrica per il rilevamento dello sguardo frontale (metodo algoritmico) . . . . .	43
7.3.5	Valutazione metrica per il rilevamento dello sguardo frontale (metodo L2CS-Net) . . . . .	44
7.3.6	Valutazione dei modelli di machine learning sviluppati . . . .	44
<b>8</b>	<b>Analisi dei risultati</b>	<b>49</b>
8.1	Rilevamento del difetto degli occhi rossi . . . . .	49
8.2	Rilevamento del difetto dello sguardo frontale . . . . .	50
8.2.1	Versione algoritmica . . . . .	50
8.2.2	Versione con l'utilizzo del modello L2CS-Net . . . . .	50
8.2.3	Versione con modelli di machine learning . . . . .	51
<b>9</b>	<b>Conclusioni</b>	<b>53</b>
		<b>55</b>
	<b>Bibliography</b>	<b>55</b>

## CONTENTS

---



---

# 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 di MobileNetV2, versione 1 .	21
5.2	Matrice di confusione del Fine-Tuning di MobileNetV2, versione 2 .	22
5.3	Matrice di confusione del modello CNN sviluppato da zero . . . . .	23
5.4	Architettura della CNN implementata . . . . .	24
5.5	Matrici di confusione dei modelli basati su landmark facciali . . . . .	26
7.1	Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione HSV) . . . . .	38
7.2	Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione yCbCr) . . . . .	40
7.3	Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (metodo algoritmico) . . . . .	41
7.4	Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (versione L2CS-Net) . . . . .	42
7.5	Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo algoritmico) . . . . .	45
7.6	Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo L2CS-Net) . . . . .	46
7.7	Distribuzione degli score per MobileNetV2 versione 1 su TONO e ONOT . . . . .	48
7.8	Distribuzione degli score per MobileNetV2 versione 2 su TONO e ONOT . . . . .	48
7.9	Distribuzione degli score per il modello gdd CNN su TONO e ONOT	48

## LIST OF FIGURES

---

8.1	Esempi di falsi positivi nella metrica di rilevamento degli occhi rossi (dataset Biolab). . . . .	50
8.2	Esempi di falsi positivi e falsi negativi nel rilevamento dello sguardo frontale con metodo algoritmico. . . . .	51
8.3	Esempi di falsi positivi e falsi negativi nel rilevamento dello sguardo frontale con L2CS-Net. . . . .	51

---

# List of Listings

listings/onnx_exporter.py . . . . .	32
-------------------------------------	----

## LIST OF LISTINGS

---

---

# 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 Obiettivo

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.

## 1.4 Struttura della tesi

La tesi è organizzata come segue:

- **Capitoli 2 - 5:** presentazione delle metriche sviluppate, degli algoritmi implementati e dei modelli di machine learning proposti.
- **Capitolo 6:** descrizione e spiegazione degli strumenti utilizzati per lo sviluppo delle metriche.
- **Capitolo 7:** valutazione delle metriche sui dataset TONO e ONOT e sulla piattaforma FVC-Ongoing.

- **Capitolo 8:** analisi dei risultati con approfondimento dei casi di fallimento per ciascun metodo.
- **Capitolo 9:** conclusioni con una sintesi dei risultati ottenuti e delle prospettive di miglioramento emerse dall'analisi.



---

## Chapter 2

# Implementazione della metrica per il rilevamento del difetto degli occhi rossi

### 2.1 Obiettivo della metrica

L'obiettivo 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.

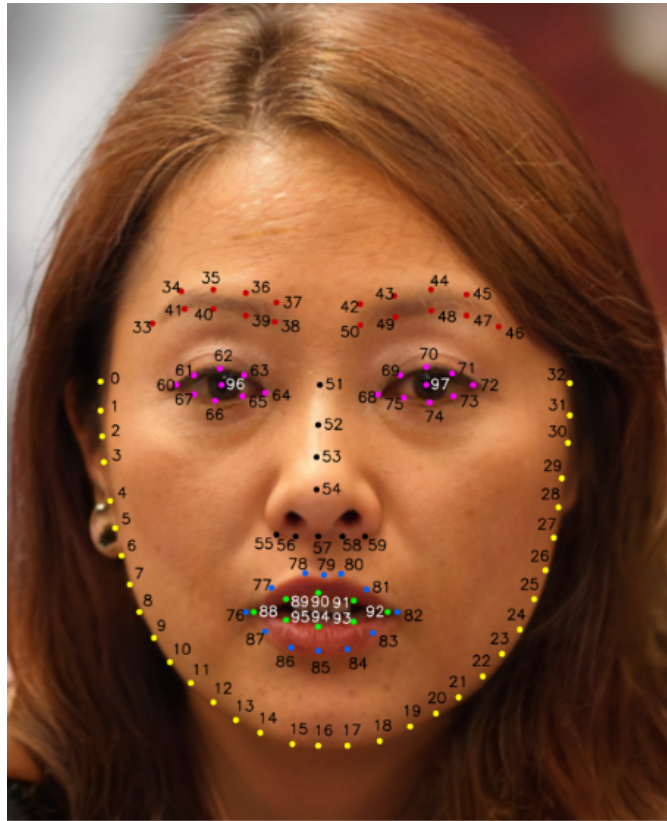


Figure 2.1: Immagine dei landmark estratti dalla rete ADNet

## 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 [WCHG20], 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 sensibile alle variazioni di luce
- yCbCr: separa luminanza da cromaticanza garantendo una misurazione più robusta a luce ed ombre, 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 canali. 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 media per evitare problemi di saturazione. In entrambi i casi le soglie sono state scelte in maniera sperimentale, cercando di rispettare un giusto 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 =  $\max(2, \min(\text{larghezza}, \text{altezza}) / 8)$
  - Calcola raggio iride =  $\text{altezza} / 2$
  - 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 (la versione HSV è stata valutata sia con che senza questo passaggio)

### 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]

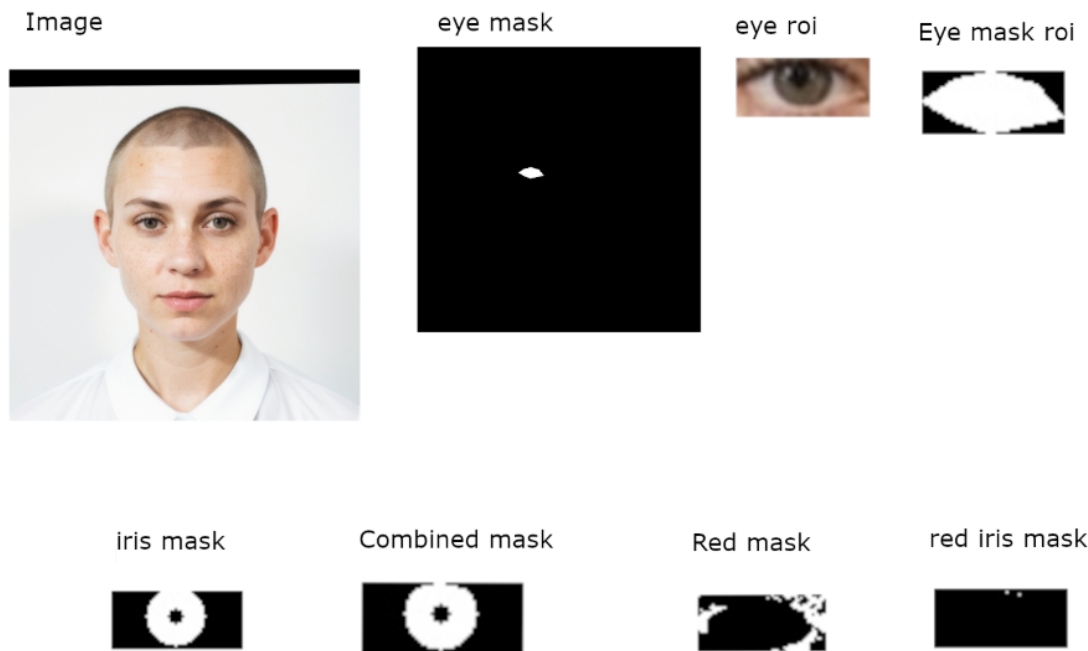


Figure 2.2: Immagine descrittiva del procedimento per il rilevamento del difetto degli occhi rossi

- 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  $\text{area iride} < 0$ :  $\text{rapportoRosso} = \text{pixelRossi} / \text{pixelTotaliIride}$
- Altrimenti:  $\text{rapportoRosso} = 0$

### 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 Scelta del modello

Dopo un'analisi dei modelli di riferimento nello stato dell'arte per il rilevamento dello sguardo, è stata individuata come base l'implementazione della rete neurale descritta nel paper [AHK<sup>+</sup>23, L2CS-NET: Fine-Grained Gaze Estimation in Unconstrained Environments]. Il modello proposto in questo lavoro ha l'obiettivo di stimare lo sguardo in contesti non vincolati, fornendo la previsione degli angoli *Pitch* e *Yaw* del volto, espressi in radianti.

### 4.2 Utilizzo del modello

Il modello L2CS-Net è distribuito sotto forma di pacchetto Python che espone un'API attraverso la quale è possibile processare un'immagine ed ottenere, per ciascun volto rilevato, i valori degli angoli *Pitch* e *Yaw*. Il modello è stato implementato in PyTorch e, per poter essere integrato nel software OFIQ, è stato necessario convertirlo nel formato **ONNX**, uno standard open-source per la rappresentazione di modelli di machine learning. Tale formato consente l'esecuzione in C++ tramite la libreria **ONNX Runtime**, già disponibile all'interno del software.

Il pacchetto Python originale di L2CS-Net include inoltre una pipeline di pre-

processing che segue i seguenti passaggi:

- Utilizzo del modello RetinaFace (fornito dalla libreria `face_detection`) per individuare le bounding box dei volti presenti nell'immagine.
- Estrazione della regione di interesse corrispondente a ciascuna bounding box
- Conversione dello spazio colore da BGR a RGB.
- Ridimensionamento di ciascuna regione di interesse a  $224 \times 224$  pixel.
- Elaborazione delle immagini tramite ResNet-50, seguita da due layer fully-connected che producono in output 90 classi rappresentanti intervalli discreti sia per *Pitch* che per *Yaw*, ognuna associata a una probabilità.
- Applicazione della funzione Softmax per ottenere la distribuzione di probabilità normalizzata sugli intervalli.
- Calcolo del valore atteso dell'angolo come somma pesata delle probabilità per ciascun intervallo.
- Conversione del risultato da classi discrete a gradi, mediante moltiplicazione per quattro e successiva sottrazione di 180.
- Trasformazione finale da gradi a radianti.

Le operazioni di preprocessing basate su OpenCV possono essere facilmente replicate all'interno di OFIQ tramite la libreria `OpenCV` per C++. Per la rilevazione della bounding box facciale, invece, è stato utilizzato il modello *SSD Face Detector CNN*, già integrato in OFIQ in formato `ONNX`. Questo modello fornisce risultati comparabili a quelli di RetinaFace.

## 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* = [0.485*f*, 0.456*f*, 0.406*f*] (dal preprocessing della librerie L2CS-NET)
    - Definisce *std* = [0.229*f*, 0.224*f*, 0.225*f*] (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  $w = 0..inputSize$
    - $tensorInput[c * inputSize * inputSize + h * inputSize + w] = immaginePreprocessata[h][w][c]$
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
    - $angoloInRadianti = \frac{\sum_i (e^{x_i} * i)}{\sum_i (e^x)} * 4 - 180$
  10.  $value = \max(|pitch|, |yaw|)$  (Si usa il valore assoluto in quanto lo sguardo frontale viene classificato come angolo 0)
  11.  $score = \text{round}((1 - (\frac{\min(value, 45)}{45})) * 100)$  (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, insieme a modelli in grado di predire l'angolazione dello sguardo, rappresenta un approccio valido per determinare se un volto stia guardando in fotocamera. Esistono tuttavia ulteriori strategie degne di nota. Questo capitolo è dedicato a tre approcci alternativi sviluppati per stabilire se uno sguardo sia rivolto verso la camera.

### 5.1 Fine-Tuning di modelli CNN

In questa sezione viene approfondito l'utilizzo di reti neurali convoluzionali preaddestrate come classificatori, successivamente adattate tramite la tecnica del *Fine-Tuning*, con l'obiettivo di predire se un volto presenti lo sguardo diretto in camera.

Per eseguire i test è stato necessario disporre di un dataset di grandi dimensioni con dati reali. La scelta è ricaduta sul dataset Gaze Direction Detection, disponibile sulla piattaforma Kaggle.

Il dataset contiene circa 53000 coppie di occhi ed è rilasciato sotto forma di array binario `numpy` (.npz), al cui interno sono presenti tre array principali:

- **targets:** contiene valori binari (0 e 1), dove l'elemento  $i$ -esimo indica se

lo sguardo corrispondente negli altri array è rivolto in camera (1) oppure altrove (0).

- **right\_eyes**: immagini raffiguranti gli occhi destri.
- **left\_eyes**: immagini raffiguranti gli occhi sinistri.

Ogni immagine ha dimensioni  $64 \times 64$  e raffigura un occhio con una piccola porzione del volto, comprendendo, a volte, anche parte del naso. Il dataset è stato suddiviso in training set (40000 immagini) e test set (4000 immagini), bilanciati in termini di immagini con sguardo in camera e non.

Il Fine-Tuning è stato eseguito sul modello MobileNetV2, la cui architettura è descritta in [SHZ<sup>+</sup>18]. Il layer di output è stato sostituito con un layer denso (Dense) di 2 nodi con funzione di attivazione **softmax**. L'addestramento è stato condotto con l'ottimizzatore **Adam** e come funzione di loss **categorical\_crossentropy**.

Sono inoltre stati adottati i seguenti accorgimenti:

- **Early stopping**: interruzione dell'addestramento se, per  $n$  epoche consecutive, non si osservano miglioramenti della loss, mantenendo i pesi relativi al miglior risultato.
- Addestramento iniziale dei soli strati finali, mantenendo congelati gli altri.
- Successivo addestramento dell'intera rete.

### 5.1.1 Fine-Tuning di MobileNetV2 - versione 1

Nella prima versione il modello è stato addestrato direttamente sulle immagini originali del dataset "Gaze Direction Detection".

**Accuracy sul test set:** 74.0%.

La matrice di confusione è riportata in Figure 5.1.

Per integrare il modello in OFIQ, è stato esportato in formato ONNX e affiancato da una pipeline di preprocessing delle immagini, così strutturata:

- Costruzione di un bounding box attorno ai landmark dell'occhio rilevati da ADNet.



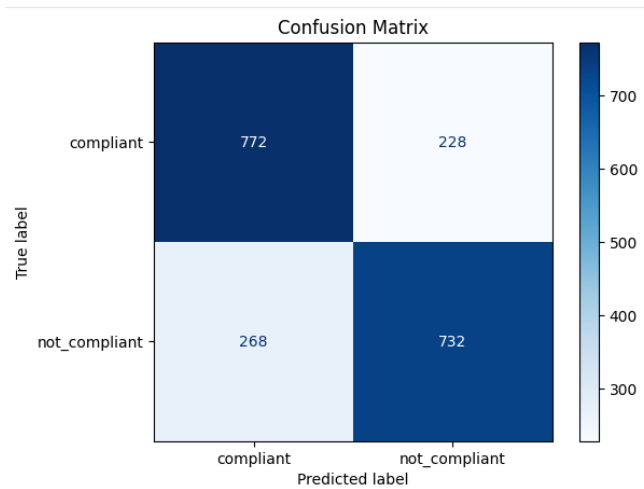


Figure 5.1: Matrice di confusione del Fine-Tuning di MobileNetV2, versione 1

- Individuazione del lato maggiore del bounding box e moltiplicazione per un fattore di scala (2.5).
- Ricostruzione del bounding box mantenendo l'occhio al centro.
- Ridimensionamento dell'immagine a  $224 \times 224$ , dimensione richiesta da MobileNetV2.
- Inferenza tramite il modello, che restituisce due valori: probabilità di sguardo frontale (tensore 0) e probabilità contraria (tensore 1).

### 5.1.2 Fine-Tuning di MobileNetV2 - versione 2

Poiché la versione 1 mostrava buoni risultati sul test set ma scarsa capacità di generalizzazione su dataset differenti, nella versione 2 si è intervenuti sul preprocessing, facendo in modo che il modello valutasse esclusivamente l'occhio.

Il preprocessing adottato è il seguente:

- Rilevamento degli occhi tramite classificatore HaarCascade di OpenCV.
- Ritaglio del bounding box dell'occhio.
- Ridimensionamento a  $96 \times 96$ .

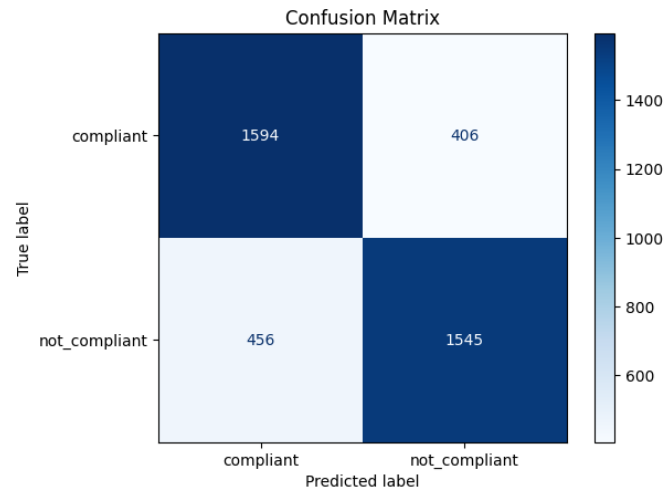


Figure 5.2: Matrice di confusione del Fine-Tuning di MobileNetV2, versione 2

- Aggiunta di un bordo nero fino a raggiungere  $224 \times 224$ , così da preservare le proporzioni senza introdurre rumore eccessivo.

**Accuracy sul test set:** 78.4%.

La matrice di confusione è riportata in Figure 5.2.

## 5.2 Costruzione di un modello CNN gdd CNN

Per ridurre l'impatto del rumore dovuto al ridimensionamento e a particolari tecniche di preprocessing, è stato implementato un modello CNN da zero, accettando in input immagini di dimensione  $96 \times 96$ .

La struttura del modello, illustrata in Figure 5.4, comprende 4 blocchi alternati di convoluzione e max pooling, fino a ottenere una dimensione di  $6 \times 6 \times 256$ . Segue una rete fully connected con 512 nodi, un layer di dropout e infine il layer di output con 2 classi e funzione di attivazione `softmax`.

Addestramento:

- Ottimizzatore: `Adam`.
- Funzione di loss: `sparse_categorical_crossentropy`.
- Early stopping.

### 5.3. MODELLI DI MACHINE LEARNING BASATI SUI LANDMARK FACCIALI

---

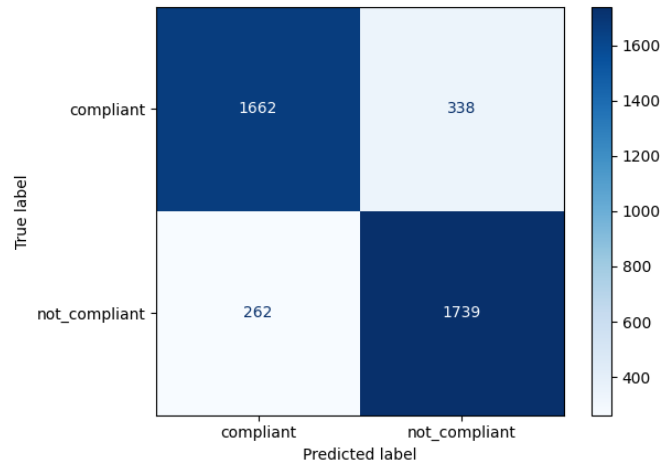


Figure 5.3: Matrice di confusione del modello CNN sviluppato da zero

Il modello è stato addestrato sul dataset “Gaze Direction Detection” e successivamente convertito in formato ONNX per l’integrazione in OFIQ.

Pipeline di preprocessing:

- Rilevamento dell’occhio con HaarCascade di OpenCV.
- Ritaglio del bounding box dell’occhio;
- Ridimensionamento a  $96 \times 96$ .

**Accuracy sul test set:** 85.0%.

La matrice di confusione è riportata in Figure 5.3.

Il modello è stato denominato **gdd CNN** (Gaze Direction Detector CNN).

## 5.3 Modelli di machine learning basati sui landmark facciali

Per affrontare i problemi di generalizzazione emersi nei modelli CNN, è stata sviluppata una soluzione basata sull’estrazione dei landmark facciali e sul loro utilizzo come input per modelli di machine learning.

Struttura generale:

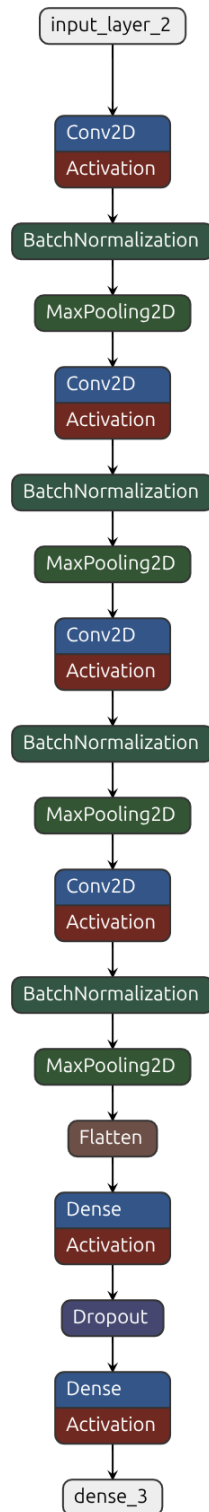


Figure 5.4: Architettura della CNN implementata

- **Estrattore di landmark:** due approcci testati
  - **MediaPipe Face Mesh** (20 landmark per occhio, comprensivi dell'iride, Paper [LTN<sup>+</sup>19]).
  - **ADNet** (9 landmark per occhio).
- **DataFrame pandas:** per la gestione dei landmark.
- **Modello predittivo:**
  - **XGBoost**.
  - Rete neurale fully connected in Keras (input layer con un neurone per colonna, hidden layer da 128 neuroni, output layer con un neurone e attivazione `sigmoid`).

L'addestramento ha utilizzato la tecnica della *K-Fold Cross Validation*, con landmark normalizzati per favorire la generalizzazione.

Dataset:

- **ONOT:** immagini con sguardo in camera;
- **TONO:** immagini con sguardo non in camera;

per un totale di circa 1.000 immagini.

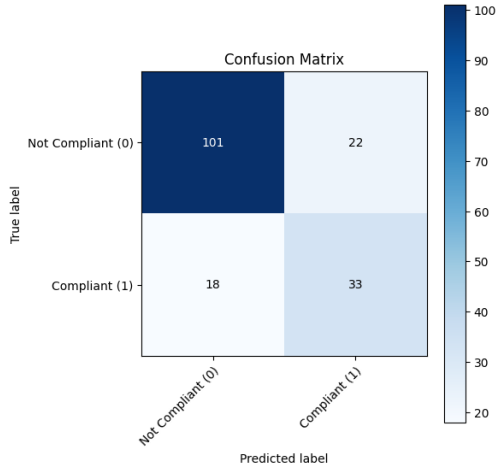
Il dataset "Gaze Direction Detection" non è stato utilizzato poiché non includeva il volto completo, necessario agli estrattori di landmark. I modelli sono stati inoltre testati su un sotto-dataset di immagini reali fornite da Biolab, tutte con sguardo frontale.

#### 5.3.1 Risultati dei modelli basati su landmark

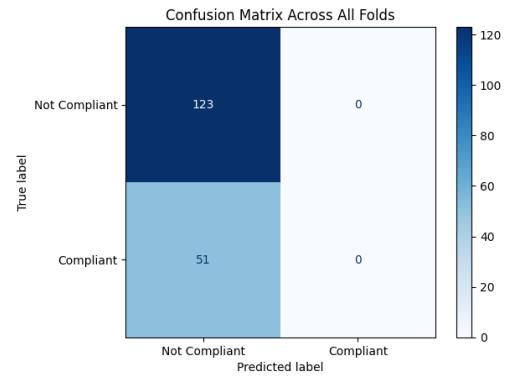
- **MediaPipe Face Mesh + XGBoost** Accuracy: 77.0%, Equal Error Rate: 26.5%, Accuracy Biolab: 63.2%. Matrice di confusione: Figure 5.5a.
- **MediaPipe Face Mesh + Keras** Accuracy: 64.0%, Equal Error Rate: 50.0%, Accuracy Biolab: 0.0%. Matrice di confusione: Figure 5.5b.

### 5.3. MODELLI DI MACHINE LEARNING BASATI SUI LANDMARK FACCIALI

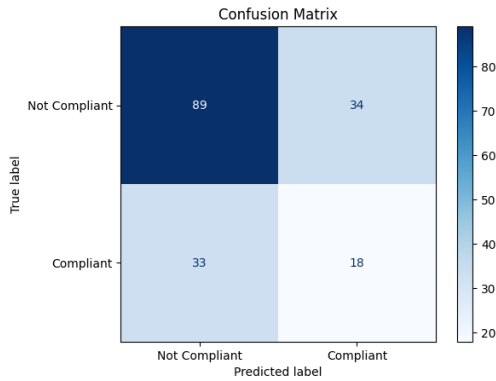
- **ADNet + XGBoost** Accuracy: 64.3%, Equal Error Rate: 43.1%, Accuracy Biolab: 12.1%. Matrice di confusione: Figure 5.5c.
- **ADNet + Keras** Accuracy: 69.5%, Equal Error Rate: 50.0%, Accuracy Biolab: 0.0%. Matrice di confusione: Figure 5.5d.



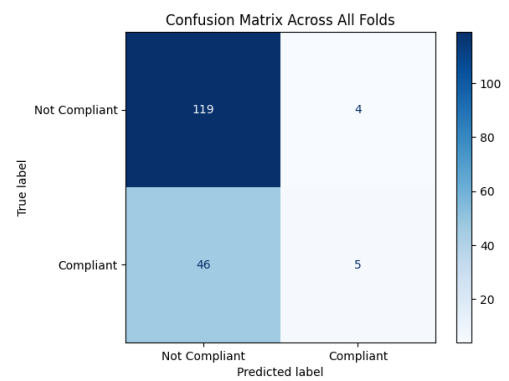
(a) MediaPipe + XGBoost



(b) MediaPipe + Keras



(c) ADNet + XGBoost



(d) ADNet + Keras

Figure 5.5: Matrici di confusione dei modelli basati su landmark facciali

---

## 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  $\pm 5^\circ$  dal frontale.
- Pose angle pitch frontal alignment: verifica che l'angolo *pitch* del volto sia inferiore a  $\pm 5^\circ$  dal frontale.
- Pose angle roll frontal alignment: verifica che l'angolo *roll* del volto sia inferiore a  $\pm 8^\circ$  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<sup>+</sup>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<sup>+</sup>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<sup>+</sup>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 Libreria Keras

Keras è una libreria open-source di alto livello per il *deep learning*, scritta in Python e basata sul motore di calcolo TensorFlow. Nel presente lavoro, Keras è stata utilizzata per:

- Eseguire il Fine-Tuning di MobileNetV2 (Sezione 5.1).

```
1 import torch
2 from l2cs import Pipeline
3 from pathlib import Path
4
5 def export_l2cs_to_onnx():
6
7     gaze_pipeline = Pipeline(
8         weights=Path("./L2CSNet_gaze360.pkl"), # pretrained model weight
9         arch='ResNet50',
10        device=torch.device('cpu')
11    )
12
13    model = gaze_pipeline.model
14
15    dummy_input = torch.randn(1, 3, 448, 448)
16
17    torch.onnx.export(
18        model,
19        dummy_input,
20        "l2cs_gaze.onnx",
21        export_params=True,
22        opset_version=11,
23        do_constant_folding=True,
24        input_names=['input'],
25        output_names=['pitch', 'yaw']
26    )
27    print("Model exported to l2cs_gaze.onnx")
28
29 if __name__ == "__main__":
30     export_l2cs_to_onnx()
```

- Sviluppare la rete CNN descritta nella Sezione 5.2.
- Implementare la rete neurale fully connected basata sui landmark (Sezione 5.3).

## 6.7 Libreria XGBoost

La libreria **XGBoost** (*eXtreme Gradient Boosting*) [CG16] mette a disposizione il modulo **XGBoostClassifier**, che consente di utilizzare modelli basati su alberi decisionali ottimizzati tramite l'algoritmo del Gradient Boosting per la risoluzione di problemi di classificazione. In questo lavoro, XGBoost è stato impiegato per costruire due dei modelli basati sui landmark (Sezione 5.3).

## 6.8 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 (un dataset con immagini di volti reali etichettati). Il linguaggio Python è stato inoltre impiegato per:

- 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
<b>Feature extraction accuracy tests</b>	
1	Eye center location accuracy
<b>Photographic and pose-specific tests</b>	
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.

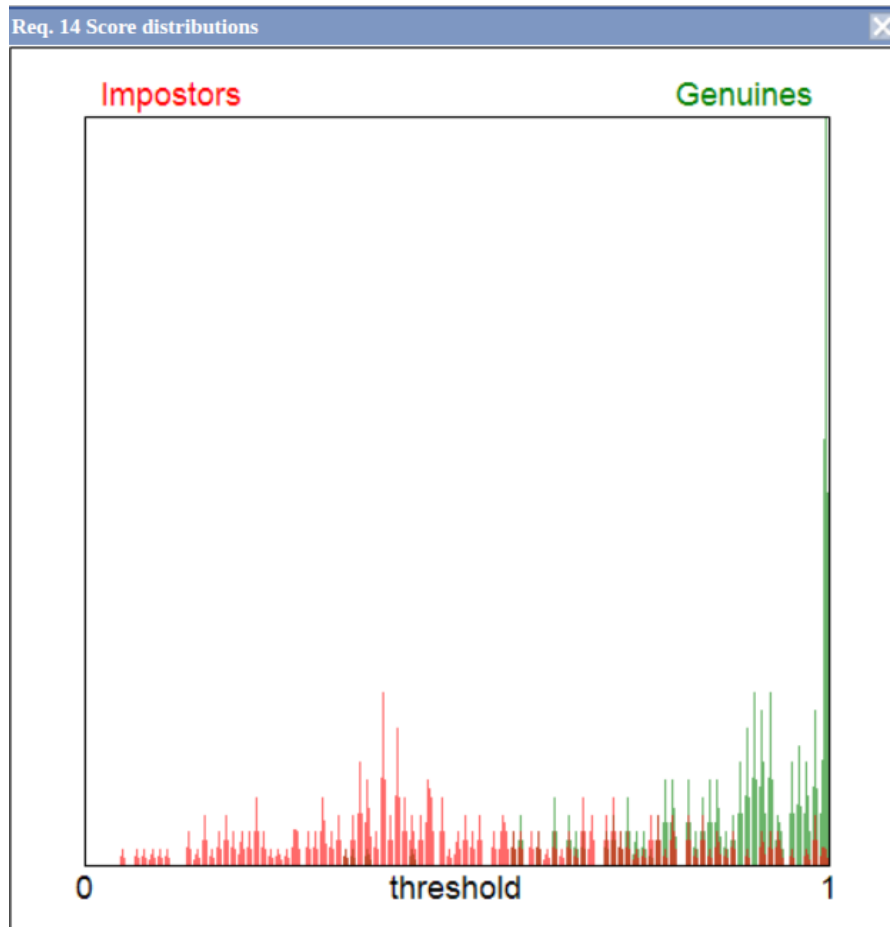


Figure 7.1: Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione HSV)

### 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.

### 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 (versione con esclusione della pupilla dal calcolo dei pixel dell'iride): 17.1%.

Equal Error Rate (versione senza esclusione della pupilla dal calcolo dei pixel dell'iride): 14.6%.

La distribuzione degli score è riportata nel grafico Figure 7.2 (questa distribuzione è relativa alla versione senza l'esclusione della pupilla dal calcolo dei pixel dell'iride).

### 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.

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

Equal Error Rate: 12.9%.

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.

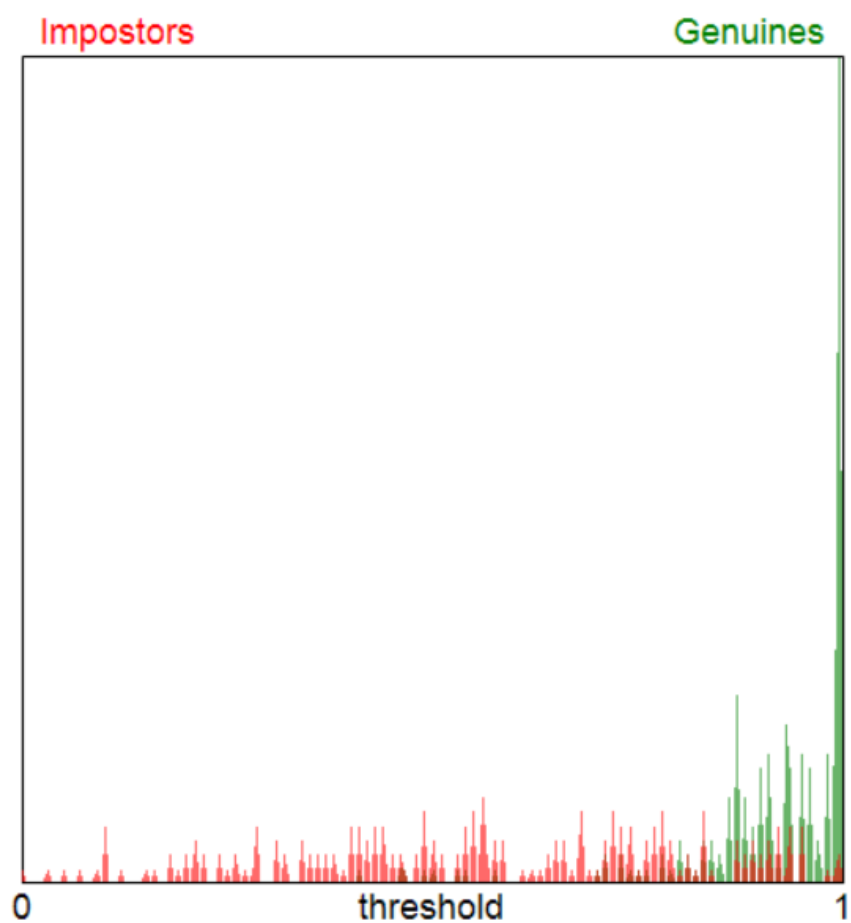


Figure 7.2: Distribuzione degli score per la metrica di rilevamento del difetto degli occhi rossi (versione yCbCr)

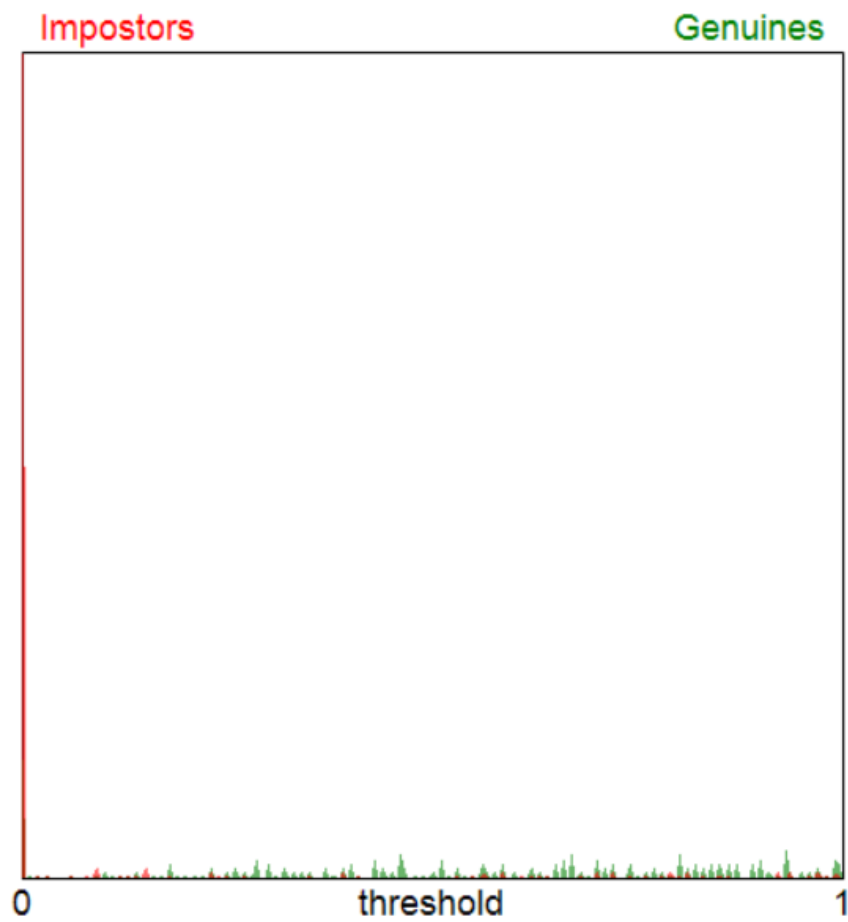


Figure 7.3: Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (metodo algoritmico)

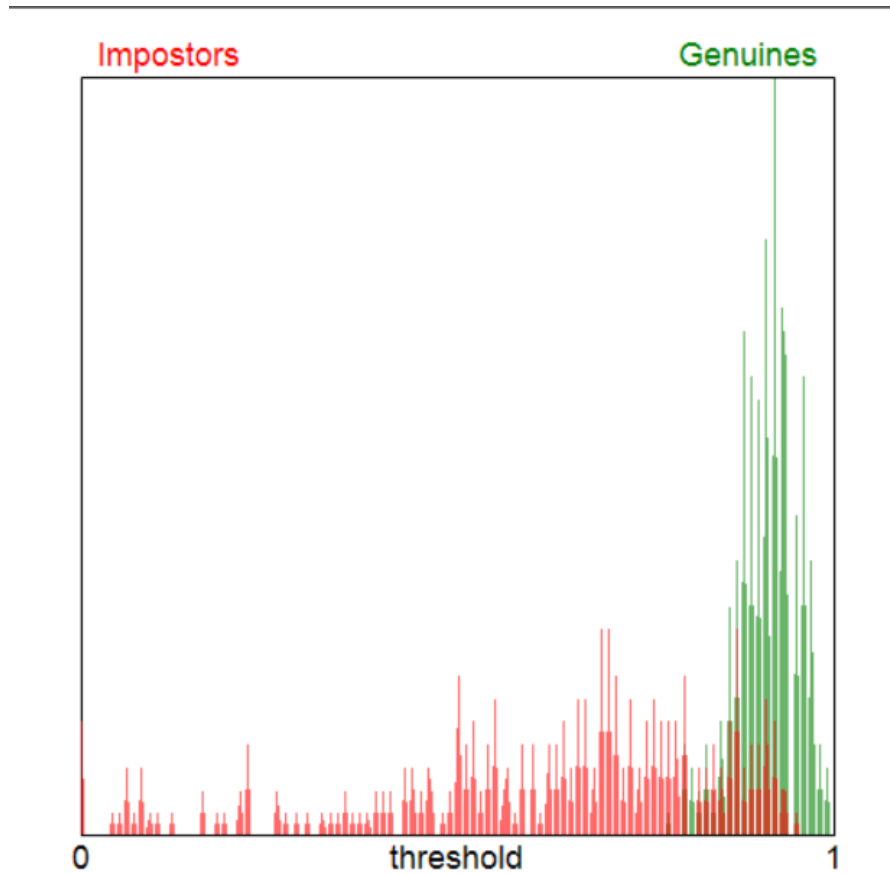


Figure 7.4: Distribuzione degli score per la metrica di rilevamento dello sguardo frontale (versione L2CS-Net)

### 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%.

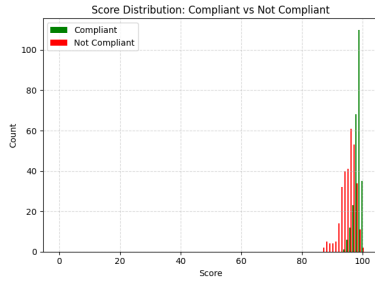
Distribuzione degli score: Figure 7.6c.

BoxPlot comparativo: Figure 7.6d.

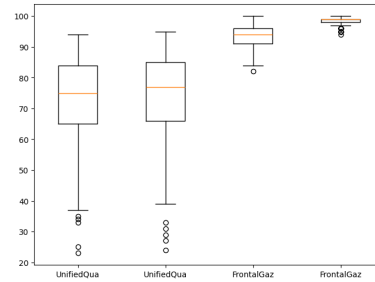
#### 7.3.6 Valutazione dei modelli di machine learning sviluppati

Di seguito sono riportati i valori di Equal Error Rate (EER) e la distribuzione degli score sui dataset TONO e ONOT per i modelli ottenuti tramite *Fine-Tuning*

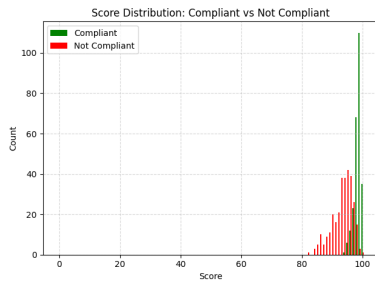




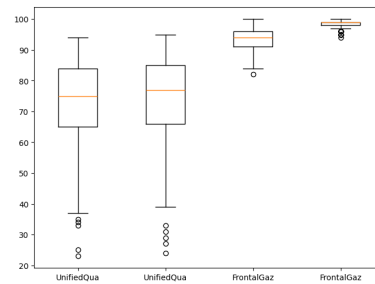
(a) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la\_1) e ONOT



(b) BoxPlot comparativo per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la\_1) e ONOT

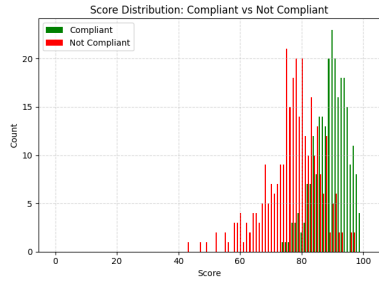


(c) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la\_2) e ONOT

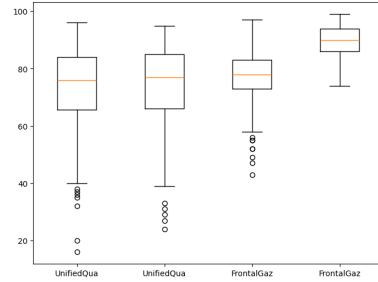


(d) BoxPlot comparativo per il rilevamento dello sguardo frontale (metodo algoritmico), dataset TONO (cartella la\_2) e ONOT

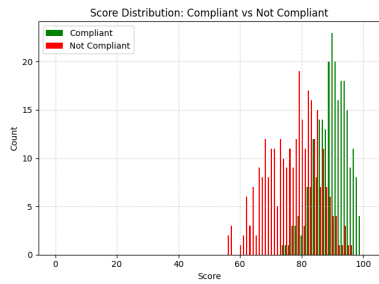
Figure 7.5: Grafici sulla valutazione della metrica del rilevamento dello sguardo frontale (metodo algoritmico)



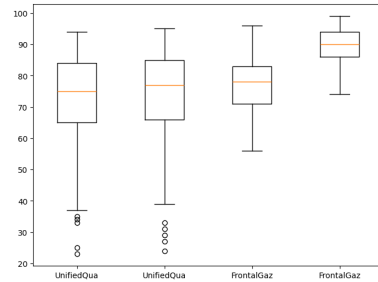
(a) Distribuzione degli score per il rilevamento dello sguardo frontale (metodo L2CS-Net), dataset TONO (cartella la\_1) e ONOT



(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)

di MobileNetV2 e per la CNN sviluppata da zero.

#### **Versione 1, Fine-Tuning di MobileNetV2 sul dataset “Gaze Direction Detection”** (Sezione 5.1.1)

- Dataset 1a\_1 di TONO: EER = 47.3%, Score distribution: Figure 7.7a
- Dataset 1a\_2 di TONO: EER = 58.5% , Score distribution: Figure 7.7b

#### **Versione 2, Fine-Tuning di MobileNetV2 sul dataset “Gaze Direction Detection”** (Sezione 5.1.2)

- Dataset 1a\_1 di TONO: EER = 46.9%, Score distribution: Figure 7.8a
- Dataset 1a\_2 di TONO: EER = 58.0%, Score distribution: Figure 7.8b

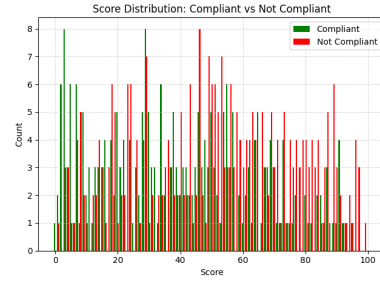
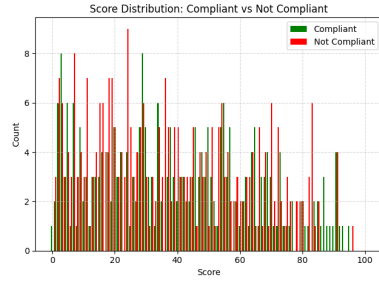
#### **gdd CNN, Modello CNN sviluppato sul dataset “Gaze Direction Detection”** (Sezione 5.2)

- Dataset 1a\_1 di TONO: EER = 38.4%, Score distribution: Figure 7.9a
- Dataset 1a\_2 di TONO: EER = 21.8%, Score distribution: Figure 7.9b

Non sono state riportate le metriche su TONO e ONOT per i modelli basati sui landmark (Sezione 5.3), in quanto tali modelli sono stati addestrati direttamente su questi dataset. Per essi sono pertanto sufficienti le metriche calcolate sul test set.

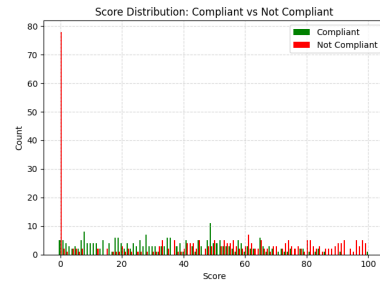
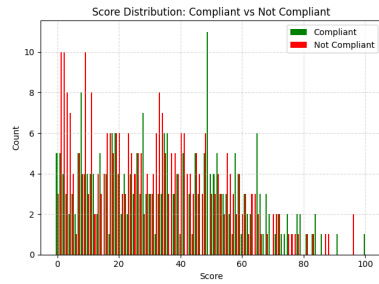
### 7.3. VALUTAZIONE SU DATASET SINTETICI

---



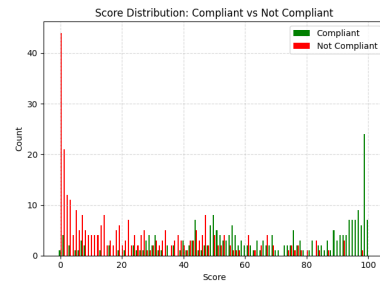
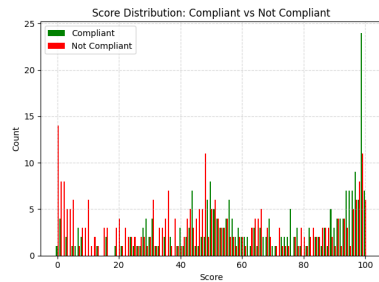
(a) Confronto con cartella 1a.1 di TONO    (b) Confronto con cartella 1a.2 di TONO

Figure 7.7: Distribuzione degli score per MobileNetV2 versione 1 su TONO e ONOT



(a) Confronto con cartella 1a.1 di TONO    (b) Confronto con cartella 1a.2 di TONO

Figure 7.8: Distribuzione degli score per MobileNetV2 versione 2 su TONO e ONOT



(a) Confronto con cartella 1a.1 di TONO    (b) Confronto con cartella 1a.2 di TONO

Figure 7.9: Distribuzione degli score per il modello gdd CNN su TONO e ONOT

---

# Chapter 8

## Analisi dei risultati

### 8.1 Rilevamento del difetto degli occhi rossi

Nella valutazione della metrica per il rilevamento degli occhi rossi si è osservato un miglioramento delle performance utilizzando lo spazio colore YCbCr rispetto a quello HSV. Ciò è dovuto principalmente a due fattori:

- Il canale **Cr** assume valori elevati in presenza del colore rosso, favorendone l'isolamento.
- L'informazione di luminanza è separata dagli altri componenti, rendendo il filtro più robusto alle variazioni di luce e ombra.

Un ulteriore incremento delle performance è stato ottenuto considerando anche i pixel della pupilla nel calcolo dei pixel totali dell'iride. Infatti, in alcuni casi il landmark non risultava correttamente centrato nella pupilla, con il rischio di escludere pixel significativi dal conteggio.

Un fattore che potrebbe aver influito sull'*Equal Error Rate* (EER), pari al 14%, è la presenza nel dataset di immagini con difetti diversi dagli occhi rossi, ma classificate come prive del difetto in questione. In questi casi, l'algoritmo ha attribuito valori di score bassi (ad esempio 51 e 66), come mostrato in Figure 8.1. Tali errori derivano da un'errata saturazione dell'immagine, difetto rilevabile tramite altre metriche implementate in OFIQ.



(a) Score: 51



(b) Score: 66

Figure 8.1: Esempi di falsi positivi nella metrica di rilevamento degli occhi rossi (dataset Biolab).

## 8.2 Rilevamento del difetto dello sguardo frontale

### 8.2.1 Versione algoritmica

Le migliori prestazioni del metodo algoritmico sono state ottenute su immagini generate artificialmente e prive di altri difetti (dataset TONO e ONOT). Al contrario, su immagini reali (es. FVC-Ongoing) le performance risultano inferiori.

Le principali criticità del metodo includono:

- proporzioni non corrette nei rapporti tra le distanze di specifici landmark oculari (esempi in Figure 8.2);
- dipendenza dalla precisione del modello di rilevamento dei landmark: in presenza di riflessi vicino agli occhi, i landmark risultano talvolta disallineati di alcuni pixel, con conseguente calo degli score;
- presenza di difetti non direttamente correlati allo sguardo, come pose non frontali del viso, con conseguente calo degli score.

### 8.2.2 Versione con l'utilizzo del modello L2CS-Net

Il metodo basato su L2CS-Net ha ottenuto i migliori risultati tra i modelli valutati, classificandosi al secondo posto tra i risultati pubblicati sulla piattaforma FVC-Ongoing. Nonostante ciò, gli score ottenuti sono ulteriormente migliorabili.

Un aspetto rilevante riguarda la pipeline di preprocessing, il modello L2CS-Net si aspetta come input un volto rilevato dal modello *RetinaFace* (libreria python `face_detection`), mentre per le valutazioni in OFIQ è stato utilizzato un diverso

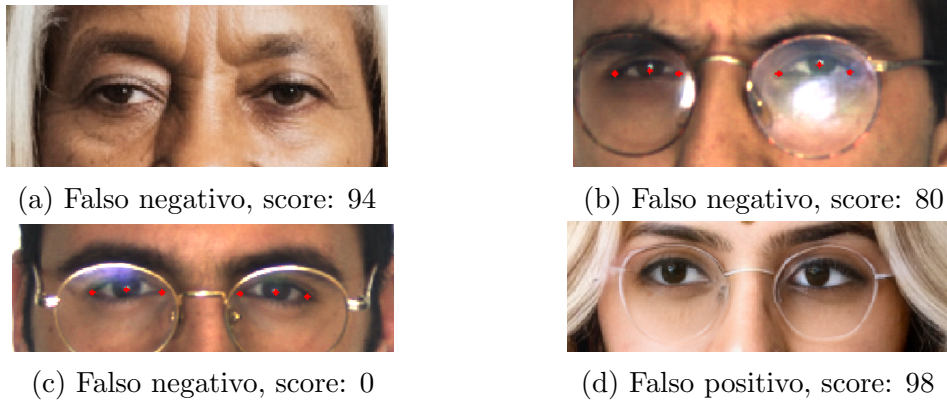


Figure 8.2: Esempi di falsi positivi e falsi negativi nel rilevamento dello sguardo frontale con metodo algoritmico.

face detector in formato ONNX già integrato. In un caso specifico (Figure 8.3b), la stima dell'angolo yaw differisce in modo significativo (0.03 radianti con *RetinaFace* contro 0.27 radianti con il detector interno). Tuttavia, nella media delle immagini la differenza supera raramente 0.03 radianti.

Inoltre, come mostrato in Figure 8.3, esistono casi in cui il modello assegna score migliori a un'immagine generata da AI con sguardo laterale (Figure 8.3a) rispetto a una reale con sguardo frontale (Figure 8.3b).



Figure 8.3: Esempi di falsi positivi e falsi negativi nel rilevamento dello sguardo frontale con L2CS-Net.

### 8.2.3 Versione con modelli di machine learning

I modelli descritti nel Chapter 5 mostrano significative difficoltà di generalizzazione: sebbene ottengano punteggi elevati sul test-set, come nel caso della CNN gdd che raggiunge un'accuratezza dell'85%, le prestazioni calano drasticamente

quando vengono valutati su dataset differenti. In particolare, i modelli ottenuti tramite Fine Tuning di MobileNetV2, testati sui dataset Tono e Onot, presentano equal error rate compresi tra il 46% e il 58%, rendendoli poco affidabili. Anche sul sottodataset Biolab, composto esclusivamente da immagini reali e compliance, le performance non migliorano, con un'accuratezza che si attesta intorno al 40%.

La CNN `gdd` ottiene performance migliori con EER compresi tra il 21% e il 38% su TONO e ONOT, e un'accuracy di circa il 66% su Biolab. Tuttavia, anche questo modello risulta inferiore ai metodi algoritmico (7.3.4) e L2CS-Net (7.3.5).

Per quanto riguarda i modelli basati sui landmark facciali, le performance non sono attualmente affidabili. Va però evidenziato che il training è stato effettuato interamente su dati sintetici, con dataset ridotto ( $\sim 1000$  immagini totali, divise in 80% training e 20% test) e sbilanciato (2 immagini non frontali ogni immagine frontale). Dai test condotti si può notare che:

- i modelli che utilizzano Mediapipe come estrattore di landmark mostrano score e capacità di generalizzazione migliori rispetto ad altri approcci;
- i modelli basati su XGBoost hanno ottenuto EER più bassi e una generalizzazione migliore rispetto alle reti neurali fully connected sviluppate con Keras.



---

## Chapter 9

# Conclusioni

Le valutazioni condotte sul software OFIQ e la relativa analisi dei risultati hanno confermato la solidità delle metriche già implementate nel software e nel draft ISO 29794-5. Sebbene i parametri considerati in OFIQ non coincidano pienamente con quelli previsti dallo standard ICAO, nelle metriche comuni il software ha dimostrato buone prestazioni sia su TONO e ONOT che su FVC Ongoing, mantenendo valori di Equal Error Rate sempre inferiori al 10%.

L’obiettivo di questa tesi è stato l’implementazione di due metriche non ancora presenti in OFIQ: il rilevamento del difetto degli occhi rossi e quello dello sguardo frontale. Nel primo caso, oltre a proporre un metodo algoritmico efficace, è stato dimostrato come l’utilizzo dello spazio colore YCbCr permetta di isolare la componente rossa con maggiore precisione rispetto allo spazio HSV, grazie alla separazione tra luminanza e crominanza che rende il filtro meno sensibile alle variazioni di illuminazione.

Per quanto riguarda lo sguardo frontale, la metrica era menzionata nel draft ISO ma non era stata implementata né resa operativa, rimanendo una proposta per edizioni future dello standard. Il lavoro è iniziato sviluppando l’approccio algoritmico suggerito, valutandolo sia su FVC Ongoing che sui dataset TONO e ONOT. Successivamente la ricerca si è concentrata sull’esplorazione di soluzioni alternative in grado di migliorare le prestazioni, obiettivo raggiunto attraverso l’impiego del modello L2CS-Net, che ha ottenuto risultati competitivi su FVC Ongoing. Nonostante questi progressi, il metodo risulta ulteriormente migliorabile

---

sfruttando **RetinaFace** per il rilevamento della bounding box facciale da utilizzare come input del modello.

Parallelamente, sono stati progettati e sperimentati ulteriori approcci basati sullo sviluppo di modelli di machine learning. Tra questi, la rete neurale **gdd CNN** ha mostrato le migliori prestazioni, evidenziando come l'adattamento del modello al contesto specifico possa incidere in maniera significativa sui risultati. L'elaborazione di immagini a dimensioni ridotte rispetto a MobileNetV2 ha ridotto il rumore nei dati, contribuendo al miglioramento delle predizioni. Tuttavia, i test cross-dataset hanno messo in luce problemi di generalizzazione, che potranno essere affrontati con l'impiego di dataset più ampi ed eterogenei. Un ulteriore contributo della ricerca è stato lo sviluppo di modelli basati sui landmark facciali. L'integrazione di MediaPipe Face Mesh con XGBoost ha prodotto buoni risultati, considerando che l'addestramento è stato condotto su un dataset ridotto, non bilanciato e composto totalmente da immagini generate artificialmente. L'ampliamento del dataset potrebbe contribuire nel miglioramento delle prestazioni e della capacità di generalizzazione. Inoltre, l'accuratezza dei landmark estratti da MediaPipe suggerisce possibili applicazioni per il miglioramento del metodo algoritmico proposto nello standard ISO 29794-5.

In conclusione, il lavoro svolto ha proposto un approccio più efficace rispetto a quello suggerito nel draft ISO 29794-5 per il rilevamento dello sguardo frontale, individuando anche diversi spunti per migliorare ulteriormente le metriche sviluppate.

---

# Bibliography

- [AHK<sup>+</sup>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<sup>+</sup>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<sup>+</sup>24] Nicolò 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.
- [LTN<sup>+</sup>19] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and

- Matthias Grundmann. Mediapipe: A framework for perceiving and processing reality. In *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019*, 2019.
- [SHZ<sup>+</sup>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.
- [WCHG20] Mingjie Wang, Hao Cai, Xin Huang, and Minglun Gong. Adnet: Adaptively dense convolutional neural networks. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 990–999, 2020.