

**Università degli Studi di Salerno**

Dipartimento di Informatica



**Tesi di Laurea in**  
**INFORMATICA**

Iris detection and segmentation in scenari non  
controllati

**Relatore:**

Prof. Andrea F. Abate

**Candidato:**

Alessio Romano

Matr. 0512104527

Anno Accademico 2018/2019

*Alla mia famiglia*

*Computer Science is no more about computers than astronomy is about  
telescopes.*

***Edsger W. Dijkstra***

# Abstract

Un framework biometrico offre una prova di identità automatica in base a caratteristiche uniche dell'individuo. Inoltre, la comunità scientifica è prevalentemente d'accordo sul fatto che il riconoscimento dell'iride sia il framework di identificazione biometrica più preciso e affidabile disponibile. Con la crescita delle richieste riguardanti l'identificazione sicura e dato che l'iride umana offre un pattern robusto per l'identificazione, l'utilizzo di strumenti poco costosi potrebbe rendere il riconoscimento dell'iride un nuovo standard nei security framework. È stato sviluppato un sistema di riconoscimento e segmentazione (identificazione dei confini al fine di estrarre solo le informazioni rilevanti) dell'iride sulla base di tecniche biometriche e sulla base di algoritmi, quali Viola-Jones, con lo scopo di mostrare il potenziale offerto dalla segmentazione dell'iride umana a partire da una riproduzione live, utilizzando un dispositivo comune quale la videocamera. In particolare, il sistema localizza gli occhi per poi estrarne le regioni circolari (pupilla e iride) dalle quali viene ottenuta la segmentazione tramite polarizzazione dell'immagine.

# Indice

<b>Abstract</b>	<b>iv</b>
<b>Lista degli acronimi</b>	<b>vii</b>
<b>Introduzione</b>	<b>viii</b>
<b>1 Concetti di Biometria e Obiettivi</b>	<b>1</b>
1.1 Concetti di Biometria . . . . .	2
1.2 Obiettivi . . . . .	4
<b>2 Artificial Intelligence e Machine Learning</b>	<b>5</b>
2.1 L'Artificial Intelligence (AI) . . . . .	6
2.2 Machine Learning . . . . .	8
2.3 Data, Datasets e Preprocessing . . . . .	11
2.3.1 Data . . . . .	11
2.3.2 Dataset . . . . .	12
2.3.3 Preprocessing . . . . .	13
2.4 Problematiche legate alla fase di Training . . . . .	15
2.4.1 Underfitting . . . . .	15
2.4.2 Overfitting . . . . .	16
<b>3 Neural Network e Convolutional Neural Network</b>	<b>17</b>
3.1 Neural Network (NN) . . . . .	18
3.1.1 Cosa sono le Neural Network? . . . . .	18
3.1.2 Come funzionano le Neural Network? . . . . .	18

3.1.3	Come modelliamo i neuroni artificiali? . . . . .	20
3.1.4	Cosa avviene nel Perceptron? . . . . .	22
3.2	Convolutional Neural Network (CNN) . . . . .	23
3.2.1	Convolution e funzionamento . . . . .	24
3.2.2	Perchè non è stato utilizzato CNN? . . . . .	26
<b>4</b>	<b>Viola Jones e Haar Cascades</b>	<b>27</b>
4.1	Viola Jones . . . . .	28
4.1.1	Features e Integral Image . . . . .	29
4.1.2	Training e Weak Classifiers . . . . .	31
4.1.3	Strong Classifier . . . . .	34
4.2	Considerazioni riguardanti Viola-Jones . . . . .	34
4.3	Haar Cascades . . . . .	35
4.3.1	Haar Feature e Adaboosting . . . . .	35
4.3.2	Cascade Classifier . . . . .	37
4.4	Considerazioni riguardanti Haar Cascades . . . . .	39
<b>5</b>	<b>Realizzazione e sviluppo del software</b>	<b>40</b>
5.1	L'approccio utilizzato . . . . .	41
5.2	Codice . . . . .	43
5.2.1	Lo script Iris Segmentation . . . . .	43
5.2.2	Lo script Extract Sector . . . . .	48
<b>6</b>	<b>Conclusioni e sviluppi futuri</b>	<b>50</b>
6.1	Conclusioni . . . . .	51
6.2	Sviluppi futuri . . . . .	51
	<b>Ringraziamenti</b>	<b>52</b>
	<b>Bibliografia</b>	<b>53</b>

# Lista degli acronimi

**AI** Artificial Intelligence

**ML** Machine Learning

**CSV** comma-separated values

**HTML** HyperText Markup Language

**XML** eXtensible Markup Language

**JSON** JavaScript Object Notation

**TSV** tab-separated values

**SQL** Structured Query Language

**NN** Neural Network

**CNN** Convolutional Neural Network

**GUI** Graphic User Interface

# Introduzione

Il lavoro di tesi, svolto durante l'attività di tirocinio presso l'Università degli studi di Salerno nella sede di Fisciano (SA), si colloca nell'ambito della ricerca e sviluppo riguardante l'integrazione di servizi di segmentazione dell'iride umana. In particolare, utilizzando *Haar Cascades* per il riconoscimento del viso e degli occhi e basandosi su algoritmi di Machine Learning (ML) è stato sviluppato un servizio di segmentazione dell'iride utilizzando un dispositivo comune quale la webcam. Utilizzando tale pattern si potranno sviluppare sistemi di autenticazione sicura. I sistemi biometrici sono in grado di fornire un livello di sicurezza più elevato rispetto ad altri sistemi di autenticazione basati su password, ma esistono alcuni problemi legati alle caratteristiche della biometria stessa (alcune cambiano nel tempo in modo significativo) o ai dispositivi utilizzati per catturarle (alcuni possono essere indotti in errore o possono avere difficoltà ad acquisire il tratto biometrico) che scoraggiano la loro diffusione. L'obiettivo del servizio è quello di mostrare il potenziale del riconoscimento e della segmentazione dell'iride.

Nel capitolo 1 vengono introdotti i concetti necessari per la comprensione degli approcci descritti in seguito.

Nel capitolo 2 viene illustrata la letteratura sulle tecniche e sui modelli di Machine Learning e i fondamenti di Artificial Intelligence (AI) cui si basa parte della progettazione del servizio prodotto. Vengono inoltre illustrati accenni sullo stato dell'arte del Machine Learning.

Nel capitolo 3 viene illustrato un possibile approccio per lo sviluppo del software prodotto e le motivazioni dietro la possibile scelta di un tale ap-



proccio

Nel capitolo 4 viene illustrato l'approccio utilizzato per lo sviluppo del software, confrontandolo all'approccio descritto nel precedente capitolo e motivandone l'utilizzo.

Nel capitolo 6 vengono presentate le conclusioni e i possibili sviluppi futuri.

# **Capitolo 1**

## **Concetti di Biometria e Obiettivi**

## 1.1 Concetti di Biometria

Le fasi di riconoscimento dell'iride disturbata sono gli stessi utilizzati in condizioni controllate, e quindi in sistemi "tradizionali", anche se richiedono approcci diversi a causa di caratteristiche dell'immagine quali la distanza dell'individuo dal dispositivo di cattura, la qualità dell'immagine e così via. Tali fasi sono, in sequenza: acquisizione, segmentazione, normalizzazione, codifica, corrispondenza.

- **Acquisizione:** rispetto ai sistemi tradizionali l'acquisizione non è necessariamente eseguita con dispositivi dedicati o videocamere di alta qualità. Immagini dell'iride possono essere ottenute da fotocamere semplici o attrezzature di acquisizione standard incorporate nel computer o dispositivi mobili. Le condizioni di acquisizione (illuminazione, distanza, posa, ecc) non sono strettamente controllate, contrariamente ai sistemi tradizionali.
- **Segmentazione:** è il processo di identificazione dei confini dell'iride al fine di estrarre solo le informazioni dell'iride dalle immagini dell'occhio. Nei sistemi tradizionali, si tratta di un'operazione relativamente semplice che consiste nel trovare due cerchi che corrispondono con i bordi pupilla-iride e pupilla-sclera. Nell'iride disturbata il processo di segmentazione è molto più complicato. Si deve tener conto dell'eventuale presenza di occlusioni o riflessi, che devono essere scartati, nel senso che la superficie corrispondente non deve essere considerata per la codifica e il confronto. L'individuazione dei confini è ulteriormente ostacolata dalla bassa risoluzione o presenza di rumore, che rendono i confini meno chiari. Per questo motivo i metodi di segmentazione dell'iride disturbata di solito implementano una fase di pre-elaborazione in cui vengono applicati filtri di smoothing (per ridurre il rumore) e/o filtri di miglioramento (per migliorare le caratteristiche, come i confini dell'iride)

- **Normalizzazione:** nei sistemi tradizionali, a causa della condizione di acquisizione controllata, è necessario solo normalizzare la forma segmentata dell'iride. La normalizzazione tipica implica la trasformazione di coordinate cartesiane in quelle polari. Se si tiene conto delle informazioni sul colore, correzione del colore, istogramma di normalizzazione o operazioni simili possono essere risultare utili.
- **Codifica:** questa fase produce un vettore modello di caratteristiche, cioè una rappresentazione compatta di un'immagine dell'iride. Le differenze negli algoritmi di estrazione di caratteristiche quando le iridi disturbate vengono processate dipendono dal fatto che in immagini di alta qualità anche piccoli dettagli strutturali dell'iride sono facilmente visibili. Al contrario, le immagini disturbate possono presentare caratteristiche alterate o meno caratteristiche da osservare. Approcci per l'estrazione di caratteristiche di immagini disturbate analizzano principalmente la texture dell'iride, quali la distribuzione del colore, la presenza di regioni più chiare o più scure e possono combinare un certo numero di operatori, ognuno applicato a una particolare caratteristica
- **Corrispondenza:** la fase di confronto dipende solo dal tipo di modelli utilizzati.[1]

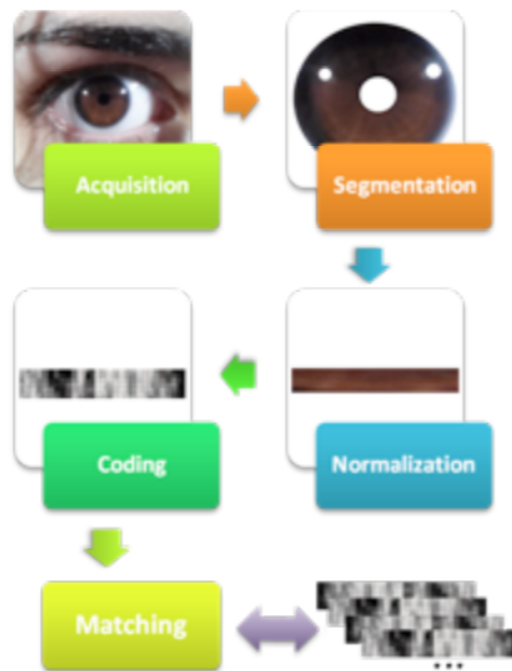


Figura 1.1: Sequenza delle fasi utili al riconoscimento dell'iride

## 1.2 Obiettivi

Gli obiettivi sono quindi la realizzazione di metodologie per effettuare acquisizione, segmentazione, normalizzazione e coding dell'iride in contesti non controllati, in particolare, esse saranno implementate utilizzando un dispositivo comune quale la webcam. Le metodologie in questione faranno parte di un sistema in grado di utilizzare le metodologie precedenti e capace quindi di ottenere, in particolare, riconoscimento, segmentazione e coding dell'iride.

## **Capitolo 2**

# **Artificial Intelligence e Machine Learning**

## 2.1 L'Artificial Intelligence (AI)

Nella sua accezione più semplice, l'intelligenza artificiale (AI) si riferisce a sistemi o macchine che imitano l'intelligenza umana per eseguire determinate attività e che sono in grado di migliorare continuamente in base alle informazioni raccolte. L'intelligenza artificiale si manifesta in varie forme. Di seguito sono riportati alcuni esempi:

- I **chatbot** utilizzano l'intelligenza artificiale per comprendere più rapidamente i problemi dei clienti e fornire risposte più efficaci.
- Gli **assistenti intelligenti** utilizzano l'intelligenza artificiale per analizzare le informazioni importanti provenienti da una grande quantità di dati testuali per migliorare la pianificazione.
- I **motori di raccomandazione** possono fornire consigli automatici per programmi TV in base alle abitudini televisive degli utenti.

L'intelligenza artificiale fornisce i processi e le capacità per potenziare al massimo la riflessione e l'analisi dei dati. Lo scopo dell'intelligenza artificiale è quello di migliorare in modo significativo le abilità e le attività degli esseri umani. Per questo motivo, è una risorsa molto preziosa per le aziende.

Il termine "intelligenza artificiale" viene utilizzato in modo generico per indicare le applicazioni che eseguono attività complesse che in passato richiedevano l'intervento umano, come comunicare con i clienti online o giocare a scacchi. Il termine viene spesso utilizzato in maniera interscambiabile con i termini *Machine Learning* e *Deep Learning*. Tuttavia, ci sono delle differenze: ad esempio, il Machine Learning è incentrato sulla creazione di sistemi che apprendono o migliorano le loro performance in base ai dati che utilizzano. È importante notare che, sebbene il Machine Learning sia sempre definibile come intelligenza artificiale, l'intelligenza artificiale non è sempre equivalente al Machine Learning.

## CAPITOLO 2: Artificial Intelligence e Machine Learning

Per sfruttare appieno il valore dell'intelligenza artificiale, molte aziende stanno investendo in modo significativo in team addetti al data science. *Data science* è un settore interdisciplinare che utilizza metodi scientifici per estrarre valore dai dati e combina le competenze di settori quali la statistica e l'informatica con le conoscenze aziendali per analizzare i dati raccolti da più fonti.

Il fondamento su cui poggia l'intelligenza artificiale, ovvero la capacità di replicare e superare il modo in cui gli umani percepiscono e reagiscono al mondo, sta diventando rapidamente la pietra miliare dell'innovazione. Grazie al Machine Learning, che riconosce gli schemi nei dati e permette di fare previsioni, l'intelligenza artificiale potrà raggiungere notevoli traguardi.

La tecnologia AI sta migliorando le performance e la produttività delle aziende grazie all'automazione dei processi o delle attività che in passato richiedevano l'intervento umano. Inoltre, l'intelligenza artificiale può sfruttare i dati a un livello che nessun essere umano potrebbe mai raggiungere.

[2]



## 2.2 Machine Learning

Il Machine Learning è un sottoinsieme dell'intelligenza artificiale (AI) che si occupa di creare sistemi che apprendono o migliorano le performance in base ai dati che utilizzano. Intelligenza artificiale è un termine generico e si riferisce a sistemi o macchine che imitano l'intelligenza umana. I termini Machine Learning e Intelligenza Artificiale vengono spesso utilizzati insieme e in modo interscambiabile, ma non hanno lo stesso significato. Un'importante distinzione è che sebbene tutto ciò che riguarda il Machine Learning rientra nell'intelligenza artificiale, l'intelligenza artificiale non include solo il Machine Learning.

Attualmente, il Machine Learning è utilizzato in qualsiasi contesto. Quando interagiamo con le banche, acquistiamo online o utilizziamo i social media, vengono utilizzati algoritmi di Machine Learning per rendere la nostra esperienza efficiente, facile e sicura. Il Machine Learning e le tecnologie ad esso associate si stanno sviluppando rapidamente e le loro funzionalità sono in continua evoluzione.

Gli algoritmi sono i motori che alimentano il Machine Learning. I due tipi principali di algoritmi di ML attualmente utilizzati sono gli *algoritmi supervisionati* e gli *algoritmi non supervisionati*. La differenza tra questi due tipi viene definita dal modo in cui ciascun algoritmo apprende i dati per fare previsioni.

### **Machine Learning supervisionato**

Gli algoritmi di Machine Learning supervisionato sono i più utilizzati. Con questo modello, un Data Scientist agisce da guida e insegna all'algoritmo i risultati da generare. Esattamente come un bambino impara a identificare i frutti memorizzandoli in un libro illustrato, nel Machine Learning supervisionato l'algoritmo apprende da un set di dati già etichettato e con un output predefinito.

Esempi di algoritmi di Machine Learning supervisionato sono gli algoritmi di regressione lineare e logistica, di classificazione multiclasse e le macchine a vettori di supporto.

### **Machine Learning non supervisionato**

Il Machine Learning non supervisionato utilizza un approccio più indipendente, in cui un computer impara a identificare processi e schemi complessi senza la guida attenta e costante di una persona. Il Machine Learning non supervisionato implica una formazione basata su dati privi di etichette e per i quali non è stato definito un output specifico.

Utilizzando ancora una volta la precedente analogia, il Machine Learning non supervisionato è simile a un bambino che impara a identificare i frutti osservando i colori e gli schemi, anziché memorizzando i nomi con l'aiuto di un insegnante. Il bambino cercherà le somiglianze tra le immagini e le suddividerà in gruppi, assegnando a ciascun gruppo la nuova etichetta corrispondente.

Esempi di algoritmi di Machine Learning non supervisionato sono gli algoritmi di clustering k-means, l'analisi di componenti principali e indipendenti e le regole di associazione.

Un'altra straordinaria caratteristica del Machine Learning è la sua capacità predittiva. In passato, le decisioni aziendali venivano spesso prese sulla base di risultati storici. Oggi, il Machine Learning utilizza analisi dei dati avanzate per eseguire previsioni. Le organizzazioni possono prendere decisioni proattive e lungimiranti anziché fare affidamento sui dati passati.

## CAPITOLO 2: Artificial Intelligence e Machine Learning

Ad esempio, la manutenzione predittiva può consentire ai produttori, alle aziende energetiche e ad altri settori di prendere l'iniziativa e verificare che le loro operazioni rimangano affidabili e ottimizzate. In un giacimento petrolifero con centinaia di trivelle in funzione, i modelli di Machine Learning possono individuare apparecchiature a rischio di malfunzionamento a breve termine e quindi avvisare i team di manutenzione in anticipo. Questo approccio non solo ottimizza la produttività, ma aumenta le performance delle risorse, dei tempi di attività e la durata delle apparecchiature. Può inoltre ridurre al minimo i rischi per i lavoratori. [3]

## 2.3 Data, Datasets e Preprocessing

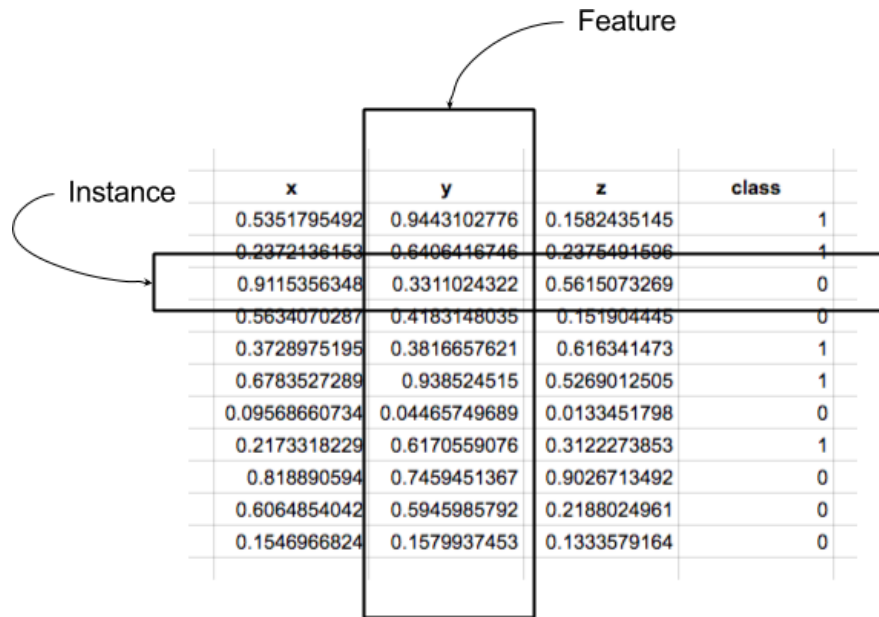
### 2.3.1 Data

I metodi di Machine Learning (ML) apprendono dagli esempi. È importante avere una buona conoscenza dei dati di input e delle varie terminologie utilizzate nella descrizione dei dati. Di seguito è riportata la terminologia fondamentale e necessaria per comprendere al meglio i dati utilizzati per algoritmi di Machine Learning.

- **Istanza:** Una singola riga di dati è chiamata istanza. È un'osservazione dal dominio
- **Feature:** Una singola colonna di dati è chiamata feature. È una componente di un'osservazione ed è anche chiamata attributo di un'istanza di dati. Alcune feature possono essere input per un modello, altre possono essere output o feature da prevedere.
- **Data Type:** Le feature hanno un tipo di dati. Possono essere reali o interi o avere un valore categorico o ordinale. Possono essere stringhe, date, orari e tipi più complessi, ma in genere vengono ridotti a valori reali o categorici quando si lavora con i metodi di Machine Learning tradizionali.
- **Dataset:** Una raccolta di istanze è un set di dati. Quando si lavora con metodi di Machine Learning sono necessari alcuni set di dati per scopi diversi.

In genere, per ottenere un modello di Machine Learning, in particolare nel caso del software prodotto, ossia un modello di classificazione binaria a partire da immagini, viene utilizzato un Dataset contenente sia immagini specifiche, riguardanti il tipo di oggetto da classificare, che immagini non specifiche in cui non è presente l'oggetto da classificare. Questi dati sono

quindi accompagnati da etichette o *label* il cui scopo è quello di indicare l'*output corretto* al modello, durante la fase di training.



	x	y	z	class
	0.5351795492	0.9443102776	0.1582435145	1
	0.2372436153	0.6406416746	0.2375491596	1
	0.9115356348	0.3311024322	0.5615073269	0
	0.5634070287	0.4183148035	0.151904445	0
	0.3728975195	0.3816657621	0.616341473	1
	0.6783527289	0.938524515	0.5269012505	1
	0.09568660734	0.04465749689	0.0133451798	0
	0.2173318229	0.6170559076	0.3122273853	1
	0.818890594	0.7459451367	0.9026713492	0
	0.6064854042	0.5945985792	0.2188024961	0
	0.1546966824	0.1579937453	0.1333579164	0

Figura 2.1: Istanze e Feature

### 2.3.2 Dataset

I Dataset, come descritto precedentemente, sono una raccolta di istanze (righe di dati). I Dataset possono essere strutturati o non strutturati. Si parla di *Dataset strutturato* quando i dati in esso contenuti sono conservati in database, organizzati secondo schemi e tabelle rigide. Si parla di *Dataset non strutturato* quando i dati sono conservati senza alcuno schema particolare. Nel caso dei Dataset non strutturati, i sistemi di gestione di dati utilizzabili sono quelli basati sul modello dell'information retrieval. Alcuni esempi di Dataset strutturati sono i Dataset organizzati secondo strutture quali comma-separated values (CSV), tab-separated values (TSV), JavaScript Object Notation (JSON), eXtensible Markup Language (XML), HyperText Markup Language (HTML) e Structured Query Language (SQL). Nel caso del software sviluppato, i dati sono *non strutturati* poichè si ha a che fare con

immagini non conservate in alcun tipo di struttura e senza una particolare organizzazione.

### 2.3.3 Preprocessing

Quando si affronta un problema di Machine Learning il primo passo consiste nel predisporre di un buon Training set (dataset utilizzato unicamente per la fase di training) a partire dai dataset disponibili in maniera tale da costruire un modello accurato.

È quindi necessaria un'analisi preliminare il cui scopo è quello di evidenziare eventuali criticità e ristrutturare i dati in modo da eliminarne criticità individuate.

Un esempio pratico riguarda la verifica della presenza di valori nulli rispetto alle feature. Dal momento che questi possono compromettere la qualità del modello occorre eliminarli. È possibile procedere in due modi:

1. Rimuovere gli esempi con valori
2. Sostituire i valori nulli con altri calcolati in maniera opportuna (media della relativa colonna)

Nei problemi di classificazione si ha spesso a che fare con categorie non numeriche che possono creare problemi. Occorre mappare opportunamente le classi con valori ordinali.

Nel caso di valori numerici è abbastanza comune ricorrere alla normalizzazione e alla standardizzazione dei dati.

## CAPITOLO 2: Artificial Intelligence e Machine Learning

La normalizzazione consente di mappare i valori nell'intervallo  $[0,1]$  e si ottiene trasformando ciascun datapoint  $x_i$  in

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

dove  $x_{min}$  e  $x_{max}$  sono rispettivamente il minimo e il massimo dell'intervallo di partenza.

La standardizzazione punta a centrare i dati intorno allo 0 e a scalarli tenendo presente la deviazione standard.

Ciascun  $x_i$  diventa

$$x_i = \frac{x_i - \mu}{\sigma}$$

dove  $\mu$  è la media e  $\sigma$  è la deviazione standard.

Naturalmente queste trasformazioni agiscono solo sull'intervallo dei dati ma non sulla loro distribuzione che resta inalterata.[4]

## 2.4 Problematiche legate alla fase di Training

Quando si parla di Training si intende "adattamento" ai dati o "apprendimento" dai dati. Durante questa fase il modello inizia ad apprendere a partire dai dati forniti. Questo passaggio è fondamentale poiché l'output finale (o previsione) del modello si baserà sulla capacità del modello di acquisire i pattern dei dati di addestramento.

Un training improprio può portare a un drastico degrado delle prestazioni del modello al momento dell'implementazione. Visti ad alto livello, esistono due tipi principali di risultati della formazione impropria: *underfitting* e *overfitting*.

### 2.4.1 Underfitting

Quando la complessità del modello è troppo ridotta perché esso apprenda i dati forniti come input, il modello si dice sia in "Underfit". In altre parole, un modello eccessivamente semplice non riesce ad apprendere le tendenze sottostanti ai dati in input. Tale risultato è causato da un modello con bassa varianza<sup>1</sup> e bias elevato.

Visualizzare graficamente un modello in underfitting può essere di aiuto nel determinare se il modello non fornisce dati adeguati durante il training. Il seguente grafico mostra un modello il cui scopo è quello di *classificare* i dati nelle classi *Croce* e *Cerchio*

---

<sup>1</sup>La varianza di una variabile statistica è una funzione che fornisce una misura della variabilità dei valori assunti dalla variabile stessa



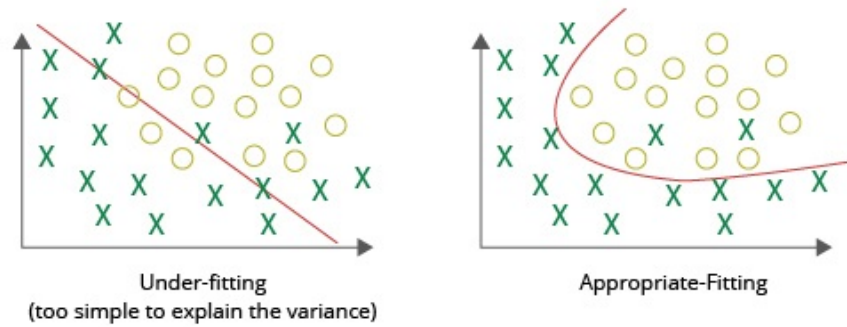


Figura 2.2: Underfitting in un task di classificazione

### 2.4.2 Overfitting

Quando la complessità del modello è troppo elevata rispetto ai dati che sta cercando di apprendere, il modello si dice sia in "Overfit". In altre parole, aumentando la complessità del modello, esso tende ad adattarsi ai noisy data<sup>2</sup> presente e agli outliers<sup>3</sup> presenti nei dati. Il modello ha appreso troppo e quindi non è più in grado di generalizzare. Tale risultato è causato da un modello con alta varianza e basso bias.[5]

Di seguito è mostrato un modello in overfit utilizzando il precedente esempio.

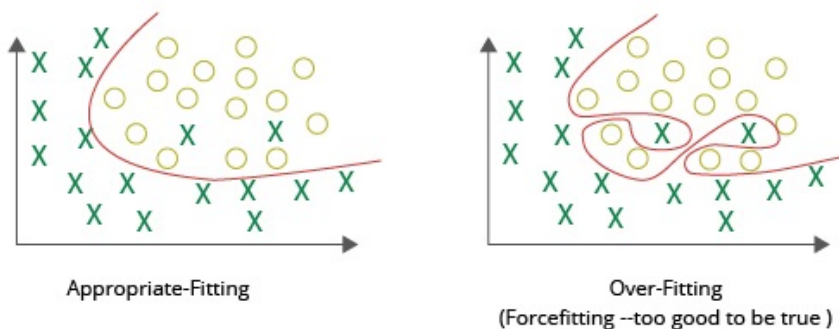


Figura 2.3: Overfitting in un task di classificazione

<sup>2</sup>Dati con un'abbondanza di informazioni addizionali. Dati corrotti o disturbati.

<sup>3</sup>Dati che deviano in maniera significativa rispetto al resto dei dati, possono essere causati da errori di misurazione.

## **Capitolo 3**

# **Neural Network e Convolutional Neural Network**

## **3.1 Neural Network (NN)**

Ispirate dalla struttura del cervello, le Neural Network (NN) sono la risposta per rendere i computer capaci di compiere attività tipicamente svolte dall'uomo.

### **3.1.1 Cosa sono le Neural Network?**

I cervelli umani interpretano il contesto delle situazioni del mondo reale in una maniera non concepibile dalle macchine. Le reti neurali furono sviluppate per la prima volta negli anni '50 per affrontare questo problema. Una rete neurale artificiale è un tentativo di simulare la rete di neuroni che formano un cervello umano in modo che il computer sia in grado di apprendere e prendere decisioni in modo umano. Le reti neurali vengono create programmando computer in maniera tale da comportarsi come se fossero cellule cerebrali interconnesse.

### **3.1.2 Come funzionano le Neural Network?**

Le reti neurali artificiali utilizzano diversi livelli di elaborazione matematica per dare un senso alle informazioni che vengono fornite. Tipicamente, una rete neurale artificiale dispone di un numero variabile di neuroni artificiali, da dozzine a milioni di essi. I neuroni artificiali vengono chiamati unità, e sono disposti in una serie di strati. Il livello di input riceve varie forme di informazioni dal mondo esterno, tali informazioni sono i dati che la rete intende elaborare o conoscere. Dall'unità di input, i dati passano attraverso una o più unità nascoste. Il compito dell'unità nascosta è trasformare l'input in qualcosa che l'unità di output può usare.

La maggior parte delle reti neurali è completamente connessa da uno strato all'altro. Queste connessioni sono ponderate: maggiore è il numero, maggiore è l'influenza che un'unità ha su un'altra, simile a un cervello uma-

no. Man mano che i dati passano attraverso ciascuna unità, la rete impara di più sui dati. Dall'altro lato della rete si trovano le unità di output: è qui che la rete risponde ai dati che sono stati forniti e processati.

Affinché le reti neurali possano apprendere, devono disporre di un'enorme quantità di informazioni che fanno parte del training set. Ad esempio, quando si cerca di insegnare ad una rete neurale come differenziare un gatto da un cane, il training set conterrà migliaia di immagini etichettate come immagini di cani e migliaia di immagini etichettate come immagini di gatti in modo da poter insegnare alla rete neurale tale differenza.

Una volta finita la fase di training, il modello prodotto proverà a classificare i dati futuri in base a ciò che viene percepito nelle diverse unità. Durante il periodo di addestramento, l'output della macchina viene confrontato con la descrizione fornita dall'uomo di ciò che dovrebbe essere osservato. Se sono uguali, la macchina è convalidata. Se non corrispondono, vengono utilizzate tecniche per regolare il suo apprendimento, andando indietro attraverso i livelli per modificare l'equazione matematica.

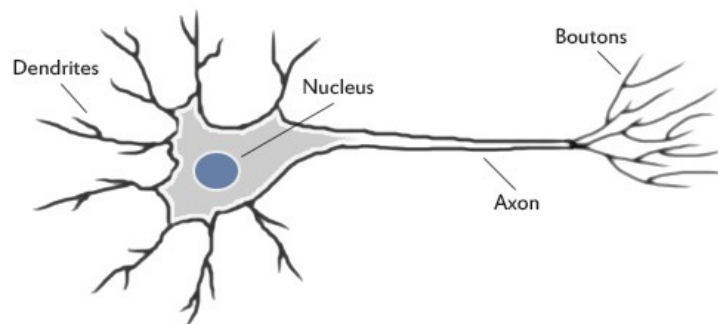


Figura 3.1: Biological Neuron

### 3.1.3 Come modelliamo i neuroni artificiali?

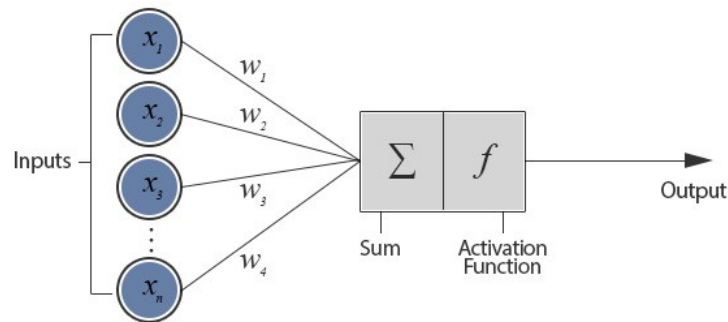


Figura 3.2: Artificial Neuron

La figura rappresenta un neurone artificiale collegato con  $n$  altri neuroni artificiali che quindi riceve  $n$  input ( $x_1, x_2, \dots, x_n$ ). Questa configurazione è chiamata **Perceptron**. Gli ingressi ( $x_1, x_2, \dots, x_n$ ) e i pesi ( $w_1, w_2, \dots, w_n$ ) sono numeri reali che possono essere positivi o negativi. Il Perceptron è costituito da pesi, da un processore di somma, da una funzione di attivazione e da un processore di soglia (noto come **bias**). Il *bias* di un algoritmo è l'insieme di assunzioni che il modello utilizza per prevedere gli output in base agli input non familiari, ossia non ancora incontrati, i pesi (*weights*) invece sono delle costanti che indicano l'importanza della variabile associata nel determinare il valore dell'output del modello.

Tutti gli ingressi sono pesati individualmente, sommati e passati alla funzione di attivazione. Esistono molti tipi distinti di funzioni di attivazione, ma una delle più semplici è la funzione a gradini. Una funzione a gradini genererà output 1 se l'ingresso è superiore a una determinata soglia, altrimenti genererà output 0.

### CAPITOLO 3: Neural Network e Convolutional Neural Network

Successivamente, verranno presentati i *vettori di input* ottenuti da un training set, il Perceptron modificherà i pesi in base alla seguente equazione:

$$\forall i \ W(i) = W(i) + A(T - A) * P(i)$$

Nota: in realtà l'equazione è

$$W(i) = W(i) + a * g' * (T - A) * P(i)$$

dove  $g'$  è la derivata della funzione di attivazione ed  $a$  è il learning rate. Il calcolo della derivata di una funzione a gradini può diventare complicato, inoltre esso non è di fondamentale importanza nella comprensione del topic in questione, quindi tale calcolo verrà ignorato.

$W$  è il vettore dei pesi,  $P$  è il vettore di input,  $T$  è l'output corretto che il Perceptron avrebbe dovuto conoscere e  $A$  è l'output dato dal Perceptron stesso.

Quando tutti i vettori di training sono stati presentati al Perceptron e non si presentano errori, allora esso è stato allenato con successo.

### 3.1.4 Cosa avviene nel Perceptron?

Il perceptron, di volta in volta, somma tutti gli input e li separa in 2 categorie, quelli che causano un *segnale di fire*<sup>1</sup> e quelli che non lo causano. Cioè, sta "disegnando" la linea:

$$w_1x_1 + w_2x_2 = t$$

I punti su un lato della linea rientrano in una categoria, i punti sull'altro lato rientrano nell'altra categoria. La linea ottenuta è solo una delle infinite linee che possono essere ottenute per la separazione delle categorie.

#### Limiti dei Perceptron

Non tutti gli insiemi di input possono essere divisi attraverso una linea. Nel caso in cui sia possibile, si parla di input *linearmente separabili*. Se i vettori non sono linearmente separabili, la fase di apprendimento non raggiungerà mai un punto in cui tutti i vettori sono classificati in maniera corretta, e bisognerà utilizzare un approccio diverso. [6]

---

<sup>1</sup>il nome deriva dalla biologia: quando i neuroni inviano un segnale nel cervello, si parla di firing

## 3.2 Convolutional Neural Network (CNN)

Le Convolutional Neural Network (CNN) sono una categoria di reti neurali che si sono dimostrate molto efficaci in settori quali il riconoscimento e la classificazione delle immagini. Le CNN, come le reti neurali (NN), sono costituite da neuroni con pesi e bias. Ogni neurone riceve diversi input, ne prende una somma, la passa attraverso una funzione di attivazione e risponde con un output, in pratica sembrano essere equivalenti alle reti neurali. Diviene quindi naturale porsi la domanda: **Qual è la differenza tra CNN e NN?**

Tra le varie differenze, la risposta è data dal fatto che le CNN hanno una caratteristica fondamentale: *lavorano su volumi*, ossia, invece di ricevere in input un vettore di valori, ricevono in input un'immagine, che è composta quindi da 3 vettori, uno per dimensione.



### 3.2.1 Convolution e funzionamento

Per capire al meglio il funzionamento delle Convolutional Neural Network (CNN) bisogna comprendere il concetto di **Convolution**(da cui il nome). Per farlo, utilizzeremo un filtro di dimensione arbitraria, in questo caso le dimensioni saranno  $5 \times 5 \times 3$ .

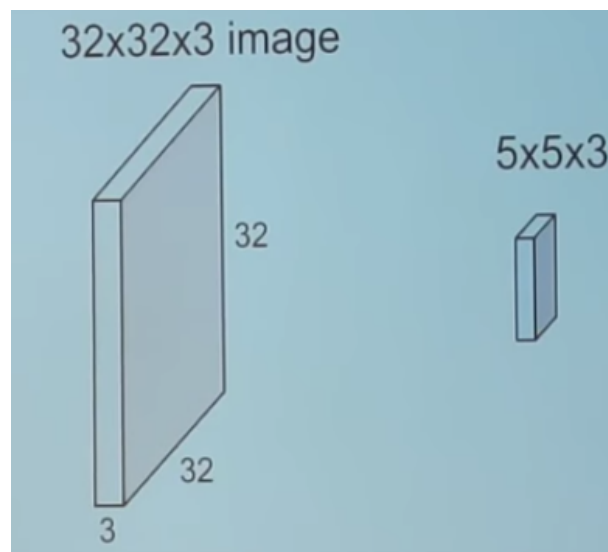


Figura 3.3: Immagine e filtro considerati

## CAPITOLO 3: Neural Network e Convolutional Neural Network

Una volta ottenuto il filtro, lo si fa scorrere sopra l'intera immagine e di volta in volta si calcola il prodotto tra il filtro e le sezioni dell'immagine sulle quali ci si trova, il risultato del prodotto sarà uno scalare.

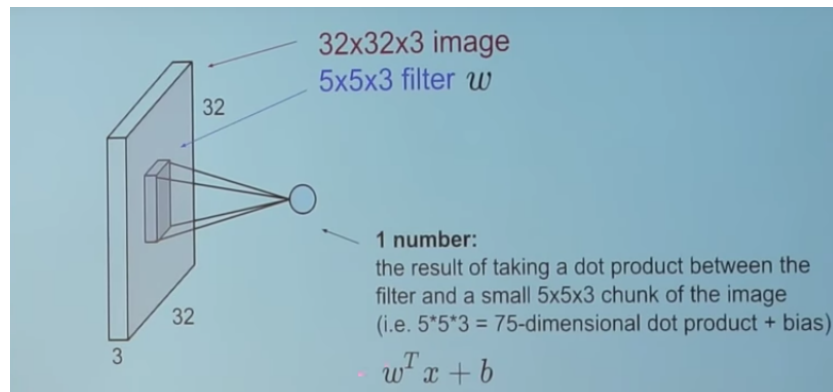


Figura 3.4: Il prodotto appena descritto

Una volta calcolati tutti i prodotti si otterrà il seguente risultato:

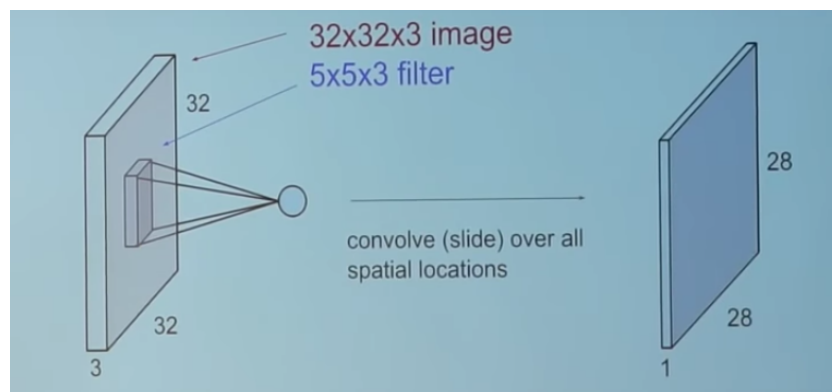


Figura 3.5: Risultato finale

Tornando alle CNN, si utilizza un cosiddetto **Convolutional Layer** (strato convoluzionale) che costituisce la parte fondamentale di una Convolutional Neural Network.

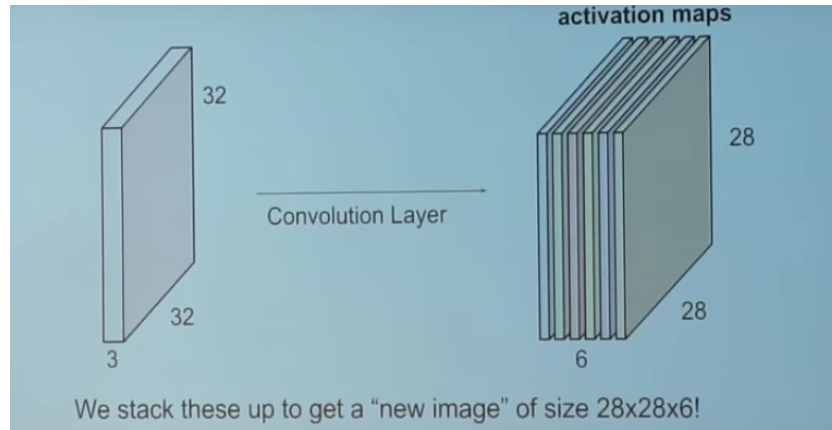


Figura 3.6: Convolutional Layer

Lo strato convoluzionale contiene un insieme di filtri indipendenti (6 nell'esempio mostrato). Ogni filtro è convoluto con l'immagine indipendentemente dagli altri, risultando quindi in 6 *feature maps* differenti (una feature map non è altro che l'output dell'applicazione del layer all'immagine). [7]

### 3.2.2 Perché non è stato utilizzato CNN?

Come già detto precedentemente, le reti neurali convoluzionali sono una categoria di reti neurali molto efficaci per task di classificazione delle immagini. Nonostante ciò, durante l'attività di progettazione del software, in particolare durante la ricerca di Dataset contenenti immagini di iridi per il training della CNN, è stata riscontrata una notevole mancanza di dati: il dataset finale contiene 2500 immagini, da dividere in training set e test set, un numero piuttosto basso. Tutto ciò, insieme alla estrema specificità dei dati causa overfitting da parte della CNN utilizzata, è stato quindi utilizzato un nuovo approccio: *Viola-Jones* e *Haar Cascades*

## **Capitolo 4**

### **Viola Jones e Haar Cascades**

## 4.1 Viola Jones

La classificazione delle immagini è un campo in rapida crescita e l'utilizzo delle reti neurali convoluzionali (CNN) e di altre tecniche di apprendimento è in rapida crescita. Tuttavia, prima che le CNN diventassero di utilizzo comune, un'altra tecnica era ampiamente utilizzata e continua ad essere utilizzata: *Viola-Jones*. Mentre una CNN è un singolo classificatore che osserva un'immagine per intero e applica operazioni matriciali per arrivare a una classificazione, Viola-Jones adotta un approccio d'insieme. Ciò significa che Viola-Jones utilizza molti classificatori diversi, ognuno dei quali osserva una diversa porzione dell'immagine. Ogni singolo classificatore è più debole (meno accurato, produce più falsi positivi) rispetto al classificatore finale perché riceve meno informazioni. Quando i risultati di ciascun classificatore vengono combinati, tuttavia, viene prodotto un classificatore forte.

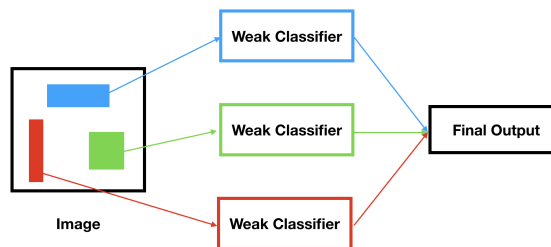


Figura 4.1: Classificatori in Viola-Jones

A causa della natura dell'algoritmo, il metodo Viola-Jones è limitato alle attività di classificazione binaria (come il rilevamento di oggetti) e ha un periodo di training piuttosto lungo. Tuttavia, esso classifica le immagini rapidamente grazie al fatto che ogni classificatore debole richiede solo un piccolo numero di parametri, inoltre, con un numero sufficiente di classificatori deboli, l'algoritmo ha un basso tasso di falsi positivi<sup>1</sup>.

<sup>1</sup>Un falso positivo è un risultato in cui il modello prevede erroneamente la classe come positiva.

### 4.1.1 Features e Integral Image

Uno dei primi contributi chiave apportati nel documento[8] che introduce Viola-Jones è stato un insieme di semplici feature da utilizzare nel riconoscimento delle immagini. Nella maggior parte delle attività, i valori dei pixel sono le caratteristiche immesse nell'algoritmo. Tuttavia, Viola e Jones hanno introdotto le seguenti nuove feature.

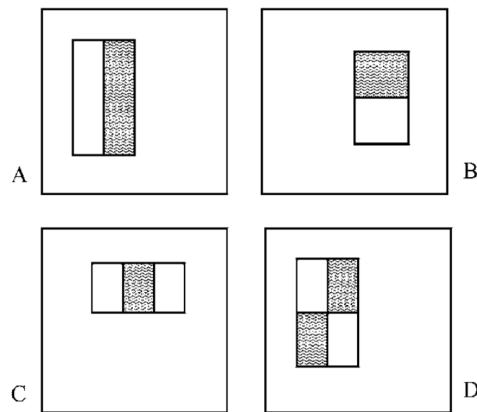


Figura 4.2: A e B sono feature a due rettangoli, C è una feature a tre rettangoli e D è una feature a 4 rettangoli. Immagine tratta dal documento originale.

La somma dei pixel nei rettangoli non ombreggiati viene sottratta dalla somma dei pixel nei rettangoli ombreggiati. È facile vedere che anche per le immagini di piccole dimensioni ci sono molte feature (oltre 160.000 per un'immagine 24 x 24). Poiché l'algoritmo richiede l'iterazione di tutte le feature, è necessario calcolarle in modo molto efficiente. Per fare questo, Viola e Jones hanno introdotto le *Integral Image*, L'immagine integrale è definita dalla seguente relazione ricorsiva:

$$\begin{aligned}
ii(-1, y) &= 0 \\
s(x, -1) &= 0 \\
s(x, y) &= s(x, y-1) + i(x, y) \\
ii(x, y) &= ii(x-1, y) + s(x, y)
\end{aligned}$$

Figura 4.3: Relazione ricorsiva dell'immagine integrale

$s(x, y)$  è la somma cumulativa delle righe nel punto  $(x, y)$ ,  $ii(x, y)$  è il valore dell'immagine integrale nello stesso punto e  $i(x, y)$  è il valore del pixel in quel punto. Questa relazione non dice altro che l'immagine integrale in un punto  $(x, y)$  è la somma di tutti i pixel in alto a sinistra rispetto al pixel corrente. Ciò semplifica il calcolo della somma dei pixel in una regione rettangolare come mostrato di seguito.

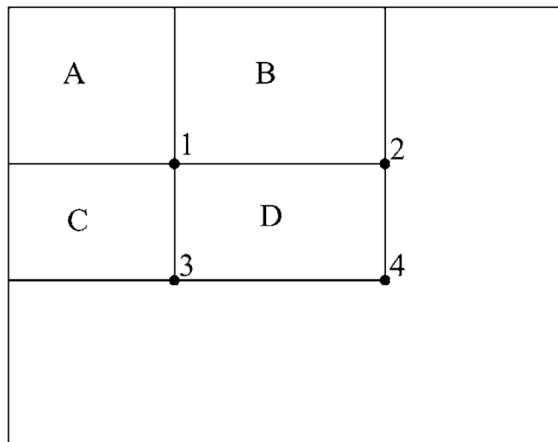


Figura 4.4: Calcolo dei pixel in una regione rettangolare

La somma dei pixel nella regione  $D$  è  $ii(4) + ii(1) - ii(2) - ii(3)$  che non sono altro che quattro riferimenti ad array.

### 4.1.2 Training e Weak Classifiers

Il ciclo principale di training richiede la scelta del miglior classificatore debole, ma esiste un classificatore debole per ogni possibile caratteristica. Per questo motivo, dobbiamo creare tutte le feature prima di iniziare a implementare il ciclo principale di training.

Quando verranno identificati i classificatori deboli ottimali da utilizzare in seguito nell'algoritmo, sarà necessario valutare ciascuna feature per ogni esempio di training. Per risparmiare computazione, verrà effettuata la valutazione di ogni feature prima di iniziare il training dei classificatori. Questa scelta è più efficiente perché ogni classificatore deve essere riqualificato ogni volta che ne viene selezionato uno nuovo.

Viola-Jones utilizza una serie di classificatori deboli e pondera i loro risultati insieme per produrre la classificazione finale. Ogni classificatore è "debole" perché da solo non può adempiere con precisione all'attività di classificazione. Ogni classificatore debole osserva una singola feature  $f$ , ha una soglia  $\theta$  e una polarità  $p$  per determinare la classificazione di un esempio di training.

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta, \\ 0 & \text{otherwise} \end{cases}$$

Figura 4.5: Classificatore debole

La polarità può essere -1 o 1. Quando  $p = 1$ , il classificatore debole produce un risultato positivo quando  $f(x) < \theta$  o quando il valore della feature è inferiore alla soglia. Quando  $p = -1$  il classificatore debole produce un risultato positivo per  $f(x) > \theta$ .



Si noti che ogni feature è la somma delle regioni rettangolari positive con la somma delle regioni rettangolari negative sottratte da esse. Il prossimo passo dell'algoritmo è scegliere il miglior classificatore debole. Per fare ciò, dobbiamo trovare la soglia e la polarità ottimali per ognuno di essi.

L'addestramento dei classificatori deboli è la parte più costosa dal punto di vista computazionale dell'algoritmo. Ogni volta che un nuovo classificatore debole viene selezionato come il migliore, tutti gli altri devono essere riqualeficati poiché gli esempi di allenamento sono pesati in maniera diversa. Tuttavia, esiste un modo efficiente per trovare la soglia e la polarità ottimali per un singolo classificatore debole utilizzando i pesi. Innanzitutto, si ordinano i pesi in base al valore della feature alla quale corrispondono, dopodiché si scorre attraverso la matrice dei pesi e si calcola l'errore se la soglia è stata scelta per essere quella feature. Infine, si trovano soglia e polarità aventi il minimo errore. I possibili valori per una soglia sono i valori della feature su ciascun esempio di allenamento. L'errore può essere misurato da:

$$e = \min(S^+ + T^- - S^-, S^- + T^+ - S^+)$$

Dove  $T$  rappresenta la somma totale dei pesi e  $S$  rappresenta la somma dei pesi di tutti gli esempi visti finora. Gli apici  $+$  e  $-$  indicano a quale classe appartiene la somma. Concettualmente, questo errore confronta quanti esempi verranno classificati in modo errato se tutti gli esempi al di sotto della posizione corrente sono etichettati come negativi con quanti esempi saranno classificati in modo errato se tutti gli esempi al di sotto della posizione corrente sono etichettati come positivi (tenendo conto di come ogni esempio è pesato).

In questa maniera, è possibile valutare l'errore di ogni possibile soglia in tempo costante ( $O(1)$ ) e l'errore di tutte le soglie in tempo lineare ( $O(n)$ ). La soglia è impostata sul valore della feature per cui l'errore è minimo. La polarità è determinata da quanti esempi positivi ed esempi negativi si trovano a sinistra (minore) e a destra (maggiore) della soglia. Se rimangono altri esempi positivi a sinistra della soglia allora  $p = 1$ . Altrimenti,  $p = -1$ .

## CAPITOLO 4: Viola Jones e Haar Cascades

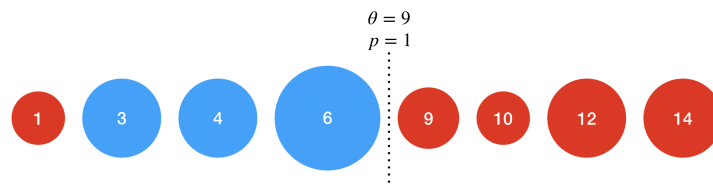


Figura 4.6: In questo esempio, i numeri indicano i valori delle feature e la dimensione delle bolle indica i loro pesi relativi. Chiaramente, l'errore verrà minimizzato quando qualsiasi feature con un valore inferiore a 9 viene classificata come blu. Ciò corrisponde a una soglia di 9 con una polarità di 1.

Una volta addestrati tutti i classificatori deboli, possiamo trovare il migliore semplicemente scorrendo attraverso tutti i classificatori e calcolando l'errore medio pesato di ciascuno di essi.

### 4.1.3 Strong Classifier

L'ultimo passaggio riguarda il classificatore finale. Dobbiamo compilare il classificatore forte a partire dai classificatori deboli. Il classificatore forte è definito come:

$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t, \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_t = \ln\left(\frac{1}{\beta_t}\right)$$

Figura 4.7: Definizione classificatore forte

Il coefficiente  $\alpha$  indica quanto ciascun classificatore debole è rilevante nella decisione finale e dipende dall'errore poiché è il logaritmo naturale dell'inverso di  $\beta$ . La somma pesata delle decisioni dei classificatori deboli viene confrontata con la metà della somma degli  $\alpha$ . [9]

## 4.2 Considerazioni riguardanti Viola-Jones

In base a quanto descritto finora, la struttura dell'algoritmo di classificazione Viola-Jones rende l'approccio particolarmente robusto sia per la selezione di feature che per la classificazione di immagini. Il documento originale di Viola-Jones introduce vari concetti per ridurre il tempo di classificazione. Durante la fase di progettazione, quindi, sono state considerate diverse opzioni per ottimizzare l'algoritmo e rendere l'approccio quanto più performante possibile soprattutto in base alle considerazioni descritte nella sezione 3.2.2 riguardanti la mancanza e l'estrema specificità dei dati.

## 4.3 Haar Cascades

Haar Cascades è un algoritmo di Machine Learning utilizzato per identificare oggetti in un'immagine o in un video. È basato sul concetto di feature proposto da Paul Viola e Michael Jones nel loro articolo "Rapid Object Detection using a Boosted Cascade of Simple Features" nel 2001. L'idea è che viene utilizzata una funzione a cascata, addestrata a partire da immagini positive e negative. Viene utilizzato quindi per rilevare oggetti all'interno di immagini. È noto per essere in grado di rilevare volti e parti del corpo in un'immagine, ma può essere addestrato per identificare quasi qualsiasi oggetto.

L'algoritmo è composto da 4 fasi:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

### 4.3.1 Haar Feature e Adaboosting

Prendiamo il rilevamento del volto come esempio. Inizialmente, l'algoritmo necessita di immagini positive di volti e immagini negative senza volti per addestrare il classificatore. Quindi dobbiamo estrarre feature da esso.

Il primo passo è quello di raccogliere le *Haar Features*. Una Haar Feature considera le regioni rettangolari adiacenti in una posizione specifica in una finestra di rilevamento, somma le intensità dei pixel in ciascuna regione e calcola la differenza tra queste somme. Per rendere questo procedimento molto rapido, vengono utilizzate le *Integral Image*

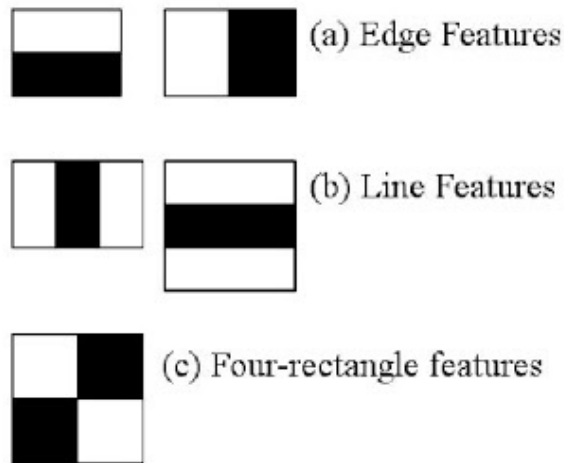


Figura 4.8: Haar Feature

Si noti che la maggior parte delle feature calcolate è irrilevante. Si consideri l'immagine riportata in seguito. La riga superiore mostra due feature rilevanti. La prima feature selezionata sembra concentrarsi sulla proprietà che la regione degli occhi è spesso più scura della regione del naso e delle guance. La seconda feature selezionata si basa sulla proprietà che gli occhi sono più scuri del ponte del naso. Contemporaneamente, però, le stesse finestre che si applicano sulle guance o in qualsiasi altro posto sono irrilevanti.

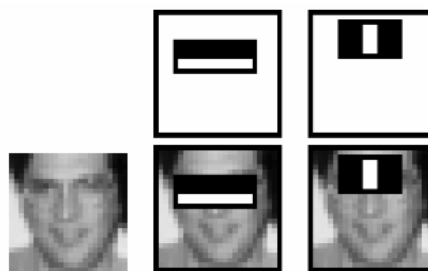


Figura 4.9: Esempio di Haar Feature

Come possiamo selezionare le migliori funzionalità tra le oltre 160.000 funzionalità? La risposta a tale domanda risiede in un approccio chiamato *Adaboost*, che seleziona le migliori feature e addestra i classificatori che le usano. Questo algoritmo costruisce un classificatore “forte” come una combinazione lineare di classificatori “deboli” pesati. Il processo è il seguente:

Durante la fase di rilevamento, una finestra della dimensione del target viene spostata sull’immagine di input e per ciascuna sottosezione dell’immagine vengono calcolate le Haar Feature. Questa differenza viene successivamente confrontata con una soglia appresa che separa gli oggetti della classe di oggetti di interesse da quelli che non vi appartengono. Poiché ogni Haar Feature è solo un “classificatore debole” (la sua qualità di rilevazione è leggermente migliore di una supposizione casuale) è necessario un gran numero di Haar Feature per descrivere un oggetto con sufficiente precisione, di conseguenza si organizzano le Haar Feature in *Cascade Classifiers* (classificatori a cascata) per formare un classificatore forte.

### 4.3.2 Cascade Classifier

Il classificatore a cascata è costituito da una raccolta di fasi, in cui ogni fase è un insieme di *Weak Learner*. I Weak Learner sono semplici classificatori chiamati *decision stumps*. Per ogni fase il training avviene utilizzando una tecnica nota come *boosting*. Il boosting fornisce la capacità di addestrare un classificatore altamente accurato prendendo una media ponderata delle decisioni prese dai classificatori deboli.

Ogni fase del classificatore identifica la regione definita dalla posizione corrente della finestra scorrevole come positiva o negativa. Positivo indica che è stato trovato un oggetto e negativo indica che non sono stati trovati oggetti.

Se l’etichetta è negativa, la classificazione di questa regione è completa e il rilevatore fa scorrere la finestra nella *posizione* successiva. Se l’etichetta è positiva, il classificatore passa la regione alla *fase* successiva. Il rilevatore

segnala un oggetto trovato nella posizione corrente della finestra quando la fase finale classifica la regione come positiva.

Le fasi sono progettate per rifiutare campioni negativi il più rapidamente possibile. L'ipotesi è che la stragrande maggioranza delle finestre non contenga l'oggetto di interesse. Al contrario, i *true positive* sono rari e vale la pena dedicare del tempo alla verifica.

- Un **true positive** si verifica quando un campione positivo è correttamente classificato.
- Un **false positive** si verifica quando un campione negativo viene erroneamente classificato come positivo.
- Un **false negative** si verifica quando un campione positivo viene erroneamente classificato come negativo.

Per un corretto funzionamento, ogni fase della cascata deve avere un basso tasso di false negative. Se una fase etichetta erroneamente un oggetto come negativo, la classificazione si interrompe e non è possibile correggere l'errore. Tuttavia, ogni fase può avere un alto tasso di falsi positivi. Anche se il rilevatore etichetta erroneamente un non-oggetto come positivo, è possibile correggere l'errore nelle fasi successive. L'aggiunta di più fasi riduce il tasso complessivo di false positive, ma riduce anche il tasso complessivo di true positive.

Il training del classificatore a cascata richiede una serie di campioni positivi e una serie di immagini negative. È necessario fornire una serie di immagini positive con le regioni di interesse specificate che verranno utilizzate come campioni positivi. È inoltre necessario fornire un set di immagini negative da cui la funzione genera automaticamente campioni negativi. [10]

## 4.4 Considerazioni riguardanti Haar Cascades

In base a quanto descritto nelle precedenti sezioni, un approccio basato sull'algoritmo Haar Cascades può portare ottime prestazioni se utilizzato in maniera corretta. Rispetto all'approccio basato su Viola-Jones, Haar Cascades si presta più efficiente e più fluido in task di rilevazione di individui *in qualsiasi contesto*. Nel prossimo capitolo viene mostrato come Haar Cascades è stato utilizzato nella stesura del software.



## **Capitolo 5**

# **Realizzazione e sviluppo del software**

## 5.1 L'approccio utilizzato

Il software è stato realizzato utilizzando due Haar Cascades attraverso i quali vengono riconosciuti occhi e viso. Gli Haar Cascades utilizzati sono nel formato XML e sono distribuiti in maniera *open source* da *Intel Corporation*. Gli Haar Cascades sono stati utilizzati grazie ai CascadeClassifier forniti all'interno della libreria *OpenCV*.

Il software prodotto, ottenuto attraverso l'impiego del linguaggio di programmazione *Python*, è costituito di due script, *ExtractSector.py* e *IrisSegmentation.py*, che verranno descritti in seguito. Da un punto di vista ad alto livello, il software ottiene la riproduzione live utilizzando una webcam, tale riproduzione viene analizzata di frame in frame, ogni frame è successivamente portato in grayscale per permettere un'analisi più efficiente. Una volta fatto ciò, vengono individuati viso, occhi, e iride, e su richiesta dell'utente verrà effettuata la segmentazione, polarizzazione e codifica delle iridi degli occhi.

```
By downloading, copying, installing or using the software you agree to this license.
If you do not agree to this license, do not download, install,
copy or use the software.

Intel License Agreement
For Open Source Computer Vision Library

Copyright (C) 2000, Intel Corporation, all rights reserved.
Third party copyrights are property of their respective owners.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistribution's of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.

* Redistribution's in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
```

Figura 5.1: Intel License Agreement

## CAPITOLO 5: Realizzazione e sviluppo del software

\* The name of Intel Corporation may not be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Intel Corporation or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Figura 5.2: Intel License Agreement

## 5.2 Codice

### 5.2.1 Lo script Iris Segmentation

Iris Segmentation è lo script contenente la maggior parte delle operazioni effettuate dal sistema. Come già accennato precedentemente, vengono utilizzati due Haar Cascades per il riconoscimento di viso e occhi. Lo script inizia ottenendo la riproduzione live a partire dalla webcam che verrà mostrata in una finestra, le cui dimensioni sono specificate in riga 12 e 13.

```
1 import cv2
2 import math
3 import numpy as np
4 import ExtractSector
5
6 # Load Haar Cascades to identify faces and eyes
7 face_cascade = cv2.CascadeClassifier('CascadeClassifiers/haarcascade_face.xml')
8 eye_cascade = cv2.CascadeClassifier('CascadeClassifiers/haarcascade_eye.xml')
9
10 # Obtain the livestream using device 0 (webcam)
11 cap = cv2.VideoCapture(0)
12 cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 500)
13 cap.set(cv2.CAP_PROP_FRAME_WIDTH, 500)
```

Figura 5.3: Fasi iniziali

Una volta inizializzata la cattura, bisogna prendere le immagini di frame in frame, in un loop infinito, questa operazione viene eseguita attraverso il metodo *read* in riga 17. Come vedremo successivamente, il loop può essere arrestato su input dell'utente. Ottenute le immagini, introduciamo un concetto chiave, chiamato *roi*, ossia la regione di interesse(region of interest) che viene analizzata in ogni dato istante.

## CAPITOLO 5: Realizzazione e sviluppo del software

```
15 while True:
16     # Extract frames from the livestream. These are the images we're going to analyze.
17     ret, img = cap.read()
18     # If unable to gain access to the webcam then the program will stop running
19     if ret is False:
20         print("Error gaining access to the webcam")
21         break
22
23     # Get the grayscale versions of the images to improve results
24     gray_roi = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
25     # Delete some of the noise from the image to improve its quality for the elaboration phase
26     roi_gray_blur = cv2.medianBlur(gray_roi, 5)
27
28     # Detect faces and eyes section:
29     faces = face_cascade.detectMultiScale(img, 1.3, 5)
```

Figura 5.4: Fasi iniziali e processing

La regione di interesse, inizialmente, corrisponde all'intera immagine, che verrà convertita in grayscale per risultati più precisi. In seguito, attraverso l'utilizzo del metodo *medianBlur* cancelleremo parte del noise contenuto all'interno dell'immagine in maniera tale da permettere al sistema di lavorare in maniera più efficace durante la fase di elaborazione.

```
28 # Detect faces and eyes section:
29 faces = face_cascade.detectMultiScale(img, 1.3, 5)
30 for (x, y, w, h) in faces:
31     # Draw rectangle around the detected faces
32     cv2.rectangle(img, (x, y), (x + w, y + h), (255, 255, 255), 1)
33     # Add a tag above the face rectangle
34     cv2.putText(img, "Face", (x-5, y-5), cv2.FONT_ITALIC, 0.4, (255, 255, 255))
35
36     # Compute y coordinates to define the area containing the eyes section
37     section1 = math.ceil(y+h/1.75)
38     section2 = math.ceil(y+h/4.5)
39
40     # Compute x coordinates to define the area containing the eyes section
41     roi_color_eyes = img[section2:section1, x:x+w]
42     roi_gray_eyes = roi_gray_blur[section2:section1, x:x+w]
43
44     # Detect eyes
45     height, width, _ = roi_color_eyes.shape
46     eyes = eye_cascade.detectMultiScale(roi_color_eyes, 1.2, 10)
```

Figura 5.5: Riconoscimento del viso

Una volta terminate le operazioni di processing dell'immagine viene avviata la procedura principale. Essa comincia effettuando una ricerca dei volti all'interno dell'immagine considerata nel dato istante, nel caso positivo, ossia quando vengono riconosciuti dei volti all'interno dell'immagine, l'algoritmo non fa altro che mostrare all'utente un rettangolo attorno ai volti attraverso la finestra mostrata precedentemente.

## CAPITOLO 5: Realizzazione e sviluppo del software

Durante la fase di progettazione è stato introdotto un concetto, denominato *fascia degli occhi*, che non è altro che l'area del viso all'interno della quale sono presenti gli occhi. Il software, quindi, una volta rilevati dei volti, calcola la fascia degli occhi, che diventerà la nuova region of interest. All'interno della nuova region of interest verrà avviata una nuova scansione, in particolare, verranno ricercati gli occhi considerando *unicamente* la nuova immagine la cui area corrisponde alla region of interest. Questa scelta migliora le prestazioni sia in termini di computazione, sia in termini di tempo impiegato.

## CAPITOLO 5: Realizzazione e sviluppo del software

In questa sezione di codice viene disegnato un rettangolo attorno ad ogni occhio, in particolare indicando precisamente l'occhio sinistro e l'occhio destro.

Arrivati a questo punto, è stata completata la fase iniziale di riconoscimento del viso e degli occhi.

```
47     for(ex, ey, eh, ew) in eyes:
48         # Draw a rectangle around the eyes
49         cv2.rectangle(roi_color_eyes, (ex, ey), (ex+ew, ey+eh), (255, 255, 255), 1)
50         # Add tags above the eyes rectangles
51         if ex+ew < width//2:
52             cv2.putText(roi_color_eyes, "Right eye", (ex-3, ey-3), cv2.FONT_ITALIC, 0.4, (255, 255, 255))
53         else:
54             cv2.putText(roi_color_eyes, "Left eye", (ex - 3, ey - 3), cv2.FONT_ITALIC, 0.4, (255, 255, 255))
55     # Notice: the eyes scan will be made only inside the eyes section to optimize the research and to save resources
```

Figura 5.6: Riconoscimento degli occhi

Viene mostrata una finestra contenente la cattura live e le informazioni minimali per l'utilizzo del software. Grazie alle procedure indicate precedentemente, quando verrà riconosciuto un volto verrà disegnato un rettangolo attorno ad esso, stesso procedimento per gli occhi.

```
57     # Add text for GUI commands and show the livestream frame by frame
58     cv2.putText(img, "Spacebar = Iris Segmentation  ESC = Exit", (0, 440), cv2.QT_FONT_NORMAL, 0.8, (255, 255, 255))
59     cv2.imshow('img', img)
```

Figura 5.7: Completamento GUI

Arrivati a questo punto, utilizziamo una variabile

$k$

per catturare gli input dell'utente, in base al tipo di tasto premuto, verrà effettuata un'azione. Quando viene premuto il tasto ESC verrà chiuso il programma.

```
61     # Wait user's inputs - based on the inputs some actions will be taken
62     k = cv2.waitKey(30) & 0xff
63
64     # ESC Button - Close the program and livestream
65     if k == 27:
66         print("ESC Button pressed - The program will stop")
67         break
```

Figura 5.8: Input utente

## CAPITOLO 5: Realizzazione e sviluppo del software

Di seguito è riportata la procedura di segmentazione e codifica dell'iride. Per avviare la procedura l'utente dovrà premere il tasto spacebar, come indicato sulla Graphic User Interface (GUI). Viene utilizzato il metodo *HoughCircles* per permettere l'identificazione della pupilla e della sclera.<sup>1</sup> L'idea è la seguente: una volta individuate le pupille e le sclere, attraverso l'utilizzo di una flag per distinguere l'occhio considerato, viene effettuata la segmentazione delle iridi, in particolare verranno salvate due immagini distinte, una per occhio, contenenti appunto i due occhi. Queste immagini verranno utilizzate in seguito. Per motivi di design, vengono salvate anche tutte le immagini ottenute durante il processing e durante l'esecuzione del software. L'ultima sezione di codice mostra la fase di chiusura della cattura

```
69 # Spacebar Button - Compute eyes segmentations
70 if k == 32:
71     try:
72         # Pupil and iris detection
73         circles = cv2.HoughCircles(roi_gray_eyes, cv2.HOUGH_GRADIENT, 1, img.shape[0] / 64, param1=150, param2=15,
74                                   minRadius=5, maxRadius=15)
75         if circles is not None:
76             circles = np.uint16(np.around(circles))
77             # We will use a j flag to distinguish which eye is being processed (0 = right, 1 = left)
78             j = 0
79             for i in circles[0, :]:
80                 if j == 0:
81                     right_eye = roi_color_eyes[i[1] - (i[2]-i[2]//3):i[1] + (i[2]-i[2]//3),
82                                                i[0] - (i[2]-i[2]//3):i[0] + (i[2]-i[2]//3)]
83                     cv2.imwrite("ProcessingImages/RightSeg.png", right_eye)
84                     j = 1
85                 else:
86                     left_eye = roi_color_eyes[i[1] - (i[2]-i[2]//3):i[1] + (i[2]-i[2]//3),
87                                                i[0] - (i[2]-i[2]//3):i[0] + (i[2]-i[2]//3)]
88                     cv2.imwrite("ProcessingImages/LeftSeg.png", left_eye)
89
90             # Save all images gained from the processing phases
91             cv2.imwrite("ProcessingImages/EyeSection.png", roi_color_eyes)
92             cv2.imwrite("ProcessingImages/LeftEye.png", roi_color_eyes[0:height, width//2:width])
93             cv2.imwrite("ProcessingImages/RightEye.png", roi_color_eyes[0:height, 0:width // 2])
```

Figura 5.9: Segmentazione dell'iride

e il rilascio della webcam, precedute dal metodo segmentation dello script Extract Sector, descritto in seguito.

---

<sup>1</sup>La sclera è la membrana bianca attorno all'occhio



```

95         # Compute segmentation
96         ExtractSector.segmentation()
97         break
98
99     except (TypeError, NameError):
100         print("Error while processing. Try again")
101
102     cap.release()
103     cv2.destroyAllWindows()

```

Figura 5.10: Segmentazione dell'iride e Extract Sector

### 5.2.2 Lo script Extract Sector

Lo script Extract Sector svolge la fase finale, al fine della quale vengono ottenute le codifiche delle due iridi degli occhi. In particolare, viene utilizzato *image slicer* per sezionare l'iride considerata, dopodichè viene effettuata la polarizzazione a partire dalle stesse immagini, e infine la codifica. La

```

1  import numpy as np
2  import image_slicer
3  from PIL import Image
4  import warnings
5
6
7  def segmentation():
8
9      try:
10         # To make things cleaner:
11         warnings.simplefilter(action='ignore', category=FutureWarning)
12
13         # Left eye segmentation process:
14
15         # Obtain eye sections
16         image_slicer.slice('ProcessingImages/LeftSeg.png', 4)

```

Figura 5.11: Il metodo segmentation

procedura è identica per entrambi gli occhi. Si inizia ottenendo le quattro sezioni dell'iride e scalandole tutte alla stessa grandezza (in particolare la grandezza della più piccola sezione per evitare di danneggiare l'output finale). Una volta ottenute le sezioni, si esegue la polarizzazione unendo le immagini in un'unica immagine, sulla quale viene eseguita la codifica.

## CAPITOLO 5: Realizzazione e sviluppo del software

```
18 # Load eye sections previously obtained
19 list_im_left = ['ProcessingImages/LeftSeg_01_01.png', 'ProcessingImages/LeftSeg_01_02.png',
20                'ProcessingImages/LeftSeg_02_01.png', 'ProcessingImages/LeftSeg_02_02.png']
21 images_left = [Image.open(i).convert('L') for i in list_im_left]
22
23 # Select the smallest image, then resize all the others so that the dimensions will match one another
24 min_shape = sorted([(np.sum(i.size), i.size) for i in images_left])[0][1]
25 images_comb_left = np.hstack((np.asarray(i.resize(min_shape)) for i in images_left))
26
27 # Save the image containing the left eye segmentation
28 images_comb = Image.fromarray(images_comb_left)
29 images_comb.save('SegmentationOutput/Left_Segmentation.jpg')
```

Figura 5.12: Occhio sinistro

```
31 # Right eye segmentation process:
32
33 # Obtain eye sections
34 image_slicer.slice('ProcessingImages/RightSeg.png', 4)
35
36 # Load eye sections previously obtained
37 list_im_right = ['ProcessingImages/RightSeg_01_01.png', 'ProcessingImages/RightSeg_01_02.png',
38                 'ProcessingImages/RightSeg_02_01.png', 'ProcessingImages/RightSeg_02_02.png']
39 images_right = [Image.open(i).convert('L') for i in list_im_right]
40
41 # Select the smallest image, then resize all the others so that the dimensions will match one another
42 min_shape = sorted([(np.sum(i.size), i.size) for i in images_right])[0][1]
43 images_comb_right = np.hstack((np.asarray(i.resize(min_shape)) for i in images_right))
44
45 # Save the image containing the right eye segmentation
46 images_comb = Image.fromarray(images_comb_right)
47 images_comb.save('SegmentationOutput/Right_Segmentation.jpg')
48 except FileNotFoundError:
49     print("Error during segmentation phase. Try again")
```

Figura 5.13: Occhio destro

## **Capitolo 6**

### **Conclusioni e sviluppi futuri**

## **6.1 Conclusioni**

Il software realizzato ha lo scopo di mostrare il potenziale di un framework di riconoscimento e segmentazione dell'iride in contesti non controllati, utilizzando un dispositivo di utilizzo comune. Attraverso ulteriori sviluppi tecnologici e attraverso dispositivi ad-hoc per il riconoscimento dell'iride è possibile ottenere un sistema in grado di raggiungere le capacità di un framework basato su riconoscimento delle impronte digitali, possibilmente anche superandole.

## **6.2 Sviluppi futuri**

Il software utilizza una comune webcam di scarsa qualità, sostituendo l'hardware con un dispositivo ad-hoc è possibile raggiungere risultati molto più rilevanti. Un possibile miglioramento si basa sull'approccio alla CNN, che però necessita di un dataset molto più ampio e strutturato di quelli attualmente esistenti.

# Ringraziamenti

Ringrazio il prof. Abate per avermi guidato nella stesura di questo lavoro e per avermi trasmesso la passione necessaria al conseguimento di questo traguardo. Desidero ringraziare il mio tutor, Silvio Barra, che è sempre stato gentile e disponibile nel seguirmi e nel risolvere ogni mio dubbio per tutta la durata del tirocinio interno. Ringrazio chi ha condiviso con me questi tre anni universitari, in particolare voglio ringraziare Andrea, Mario, Igor e Luca per avermi sostenuto e aiutato nonostante tutto. Un ringraziamento speciale alla mia famiglia, in particolare a mia madre e mio padre che grazie al supporto costante e ai loro insegnamenti mi hanno permesso di raggiungere questo traguardo e di diventare la persona che sono.

# Bibliografia

- [1] *Riconoscimento dell'iride in condizioni critiche.* url: [http://mondodigitale.aicanet.net/2014-5/articoli/08\\_Riconoscimento\\_dell\\_iride\\_in\\_condizioni\\_critiche.pdf](http://mondodigitale.aicanet.net/2014-5/articoli/08_Riconoscimento_dell_iride_in_condizioni_critiche.pdf).
- [2] *Cos'è l'intelligenza artificiale?* url: <https://www.oracle.com/it/artificial-intelligence/what-is-artificial-intelligence.html>.
- [3] *Cos'è il machine learning?* url: <https://www.oracle.com/it/artificial-intelligence/what-is-machine-learning.html>.
- [4] *Preprocessing dei dati.* url: <https://www.emmecilab.net/preprocessing-dei-dati/>.
- [5] *Underfitting and overfitting in Machine Learning.* url: <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>.
- [6] *Neural Networks introduction.* url: <https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb>.
- [7] *CNN Explained.* url: <https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8>.
- [8] *Viola-Jones paper.* url: <http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf>.

- [9] *Understanding and implementing Viola-Jones.* url: <https://medium.com/datadriveninvestor/understanding-and-implementing-the-viola-jones-image-classification-algorithm-85621f7fe20b>.
- [10] *How do Haar Cascades work?* url: <https://www.quora.com/How-do-Haar-cascades-work>.