

UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica



Corso di Laurea in Informatica

Fondamenti di Data Science e Machine Learning

Progetto: Driver Drowsiness Detection

Relatori

Prof. Giuseppe Polese

Dott.re Stefano Cirillo

Studenti

Domenico Trotta 0522500810

Michele Castellaneta 0522500824

Anno Accademico 2019/2020

Sommario

ABSTRACT	3
1. INTRODUZIONE	4
2. LAVORI PRESENTI IN LETTERATURA.....	5
3. DATASET.....	7
3.1 ANNOTAZIONI E STATISTICHE DEL DATASET	9
4. MODELLO PRELIMINARE	11
4.1 CARICAMENTO DATASET E PRE-PROCESSING.....	12
4.2 STRUTTURA.....	13
4.3 TRAINING MODELLO	16
4.4 TESTING	18
<i>4.4.1 Confusion matrix.....</i>	<i>20</i>
<i>4.4.2 Precision e Recall.....</i>	<i>21</i>
<i>4.4.3 F₁-score.....</i>	<i>22</i>
5. MODIFICHE PROPOSTE	22
5.1 STRUTTURA.....	24
5.2 TRAINING	26
5.3 TESTING	27
6. CONFRONTO TRA I MODELLI	27
7. DATA PROFILING	30
7.1 FD.....	32
7.2 RFD.....	35
<i>7.2.1 Extent</i>	<i>35</i>
<i>7.2.2 Confronto.....</i>	<i>38</i>
7.3 FEATURE SELECTION.....	41
<i>7.3.1 Selezione Univariata.....</i>	<i>42</i>
<i>7.3.2 Feature Importance.....</i>	<i>46</i>
<i>7.3.3 Matrice di Correlazione con Heatmap.....</i>	<i>51</i>
8. CONFRONTO FINALE.....	53
8.1 CONFRONTO CON LAVORI GIÀ SVOLTI.....	53
8.2 SOMMARIO DELLE Sperimentazioni EFFETTUATE	54
9. DESCRIZIONE DEMO	55
9.1 DETECTION CON BUONE CONDIZIONI DI LUCE	55
9.2 ALLARME VISIVO/SONORO	57
9.3 ACQUISIZIONE CON CAMERA IR.....	57
9.4 SCHERMATA IMPOSTAZIONI.....	58
10. CONCLUSIONI.....	59
11. RIFERIMENTI.....	60

Abstract

Secondo i rapporti di analisi sugli incidenti stradali degli ultimi anni, è noto che la principale causa di incidenti stradali con conseguenti decessi, lesioni gravi e lievi è dovuta a un guidatore assonnato. Lo stato di sonnolenza può essere causato dalla mancanza di sonno, assunzione di farmaci, droghe o guida continua per periodi prolungati. Un aumento del tasso di incidenti stradali causati dalla sonnolenza durante la guida indica la necessità di un sistema che rilevi lo stato del guidatore e lo avvisi prima che si verifichino incidenti. Negli ultimi anni, molti ricercatori hanno mostrato interesse per il rilevamento della sonnolenza (Drowsiness Detection). I loro approcci monitorano sostanzialmente le caratteristiche fisiologiche o comportamentali relative al conducente o le misure relative al veicolo utilizzato. Per far fronte a questo problema, proponiamo un algoritmo di monitoraggio degli occhi basato su una rete neurale, per la precisione una rete neurale convoluzionale per determinare lo stato aperto o chiuso dell'occhio e attivare un allarme se il conducente è assonnato. Vengono inoltre presentati risultati sperimentali dettagliati per evidenziare i punti di forza e di debolezza della nostra tecnica. Una precisione del 99% (circa) è stata registrata per la metodologia proposta. Per il test in condizioni reali è stato prodotto un programma che in real-time, grazie all'ausilio di una videocamera, riesce a determinare lo stato degli occhi del conducente.

1. Introduzione

L'elaborato discusso in questo documento tratta di un progetto svolto durante il corso di *Fondamenti di Data Science e Machine Learning*, del corso di *Laurea Magistrale in Informatica* dell'*Università degli Studi di Salerno*.

La sonnolenza di solito si verifica a causa di sonno insufficiente, acquisizione di farmaci e anche a causa della noia causata dalla guida di veicoli per un lungo periodo di tempo. In uno stato di sonnolenza, il conducente potrebbe perdere il controllo del proprio veicolo causando un incidente.

Secondo i rapporti statistici, ogni anno più di 1,35 milioni di persone muoiono in incidenti stradali, mentre 50 milioni di persone ne rimangono ferite [1]. Una delle statistiche più strazianti presenti nel rapporto dell'Organizzazione Mondiale della Sanità (OMS) è che gli incidenti stradali risultano essere la principale causa di morte tra le persone di età compresa tra 5 e 29 anni.

Per evitare gli incidenti stradali, quindi, lo stato del conducente deve essere osservato costantemente.

Lo scopo di tale progetto prevede la costruzione di un modello che sia in grado di effettuare un'operazione di Drowsiness Detection a partire da un dataset disponibile liberamente in rete. Una volta ottenuto tale modello, questo verrà utilizzato per realizzare una demo che effettui la drowsiness detection del soggetto inquadrato.

Le misurazioni che vengono impiegate per il rilevamento della sonnolenza rientrano in tre categorie di base: misurazioni fisiologiche, comportamentali e basate sul veicolo. L'approccio proposto in questo documento utilizza la misura "comportamentale", ovvero funziona interpretando i segnali visivi del conducente.

Il documento è organizzato come segue:

- la sezione 2 presenta un'indagine bibliografica nella quale si discute delle precedenti metodologie di rilevazione della sonnolenza presentate da diversi ricercatori.
- La sezione 3 discute in dettaglio del dataset utilizzato.
- La sezione 4 descrive il primo modello realizzato.
- La sezione 5 descrive i miglioramenti apportati al modello preliminare.
- La sezione 6 descrive un confronto tra i modelli presentati nelle sezioni precedenti.
- La sezione 7 descrive l'attività di data profiling effettuata sul dataset.
- La sezione 8 descrive i risultati ottenuti confrontandoli con i risultati ottenuti dai lavori già presenti in letteratura.
- La sezione 9 illustra il funzionamento della demo real-time realizzata.
- La sezione 10 descrive le conclusioni del lavoro svolto.

2. Lavori presenti in letteratura

In questa sezione si discuterà di varie metodologie che sono state proposte dai ricercatori negli ultimi anni per quanto riguarda l'operazione di Drowsiness detection.

Flores et al [2], nel 2009, hanno presentato un componente per l'Advanced Driver Assistance System (ADAS) in grado di rilevare automaticamente la sonnolenza del conducente. Il modulo utilizza algoritmi di intelligenza artificiale insieme ai dati visivi che vengono acquisiti. Il sistema identifica e monitora il viso e gli occhi e determina la sonnolenza utilizzando Support Vector Machine (SVM). Il sistema è progettato per funzionare in real-time anche in condizioni di luce variabile. Questo sistema, oltre ai battiti delle ciglia, prende in considerazione anche altre distrazioni del guidatore, ad esempio lo sbadiglio, l'inclinazione della testa e l'orientamento del viso. Tutti questi fenomeni vengono monitorati per fornire un grado di accuratezza ancora più elevato rispetto al solo rilevamento del battito delle ciglia.

Vitabile et al [3], nel 2011 hanno presentato un rilevatore di sonnolenza in tempo reale da utilizzare nei veicoli. Una fonte di luce a infrarossi da 850 nm è fissata sul cruscotto dell'auto permettendo l'illuminazione della pupilla. Ciò semplifica il rilevamento degli occhi in quanto la retina dell'occhio ha la proprietà di riflettere il 90% della luce proiettata su di essa. Lo stato di sonnolenza viene identificato quando gli occhi sono chiusi per più dell'80% di un certo periodo di tempo. Tecniche di elaborazione delle immagini efficienti sono combinate con una tecnologia hardware consolidata come Field Programmable Gate Array (FPGA). Ciò consente il rilevamento della sonnolenza in tempo reale e consente al sistema di elaborare un intero frame 720x576 in 16,7 microsecondi. La scalabilità e il riutilizzo del codice di FGPA possono aiutare a ridurre i costi di sviluppo. Sono stati osservati dei false allarms con l'utilizzo di occhiali per la vista o di oggetti che riflettono i raggi infrarossi. Per tale motivo, sono necessari ulteriori lavori per superare le limitazioni nel rilevamento degli occhi per i portatori di occhiali, in modo tale da renderlo disponibile per tutti i tipi di autisti e consentire al sistema di supportare oggetti che riflettono gli infrarossi.

Singh e Kaur [4] nel 2012, hanno proposto un metodo che rileva la sonnolenza usando l'algoritmo SIFT. Scale-Invariant Feature Transform è un algoritmo utilizzato in computer vision che permette di rilevare e descrivere caratteristiche, o feature locali nelle immagini. L'algoritmo è stato pubblicato da David G. Lowe nel 1999.

Viene effettuato il rilevamento delle palpebre in tempo reale utilizzando una webcam con risoluzione 640 x 480. Gli occhi vengono rilevati da ciascun fotogramma e ogni battito di ciglia viene misurato rispetto a un valore medio. Il sistema confronta l'apertura dell'occhio per ogni blink con un valore medio standard e viene attivato un allarme se l'apertura dell'occhio supera questo valore per un certo numero di fotogrammi consecutivi. Gli autori hanno registrato una precisione del 99%. Inoltre, la risoluzione utilizzata, ovvero 640 x 480 non è altissima, ma nonostante questo sono stati ottenuti ottimi risultati. In questo algoritmo, il sistema deve conservare le informazioni sui fotogrammi passati perché le misurazioni del blink degli occhi su una certa quantità di fotogrammi vengono utilizzate per monitorare la sonnolenza.

Chuang-Wen et al [5], nel 2013 hanno introdotto "CarSafe", la prima applicazione per smartphone Android per il rilevamento della sonnolenza. L'applicazione richiede uno smartphone con doppia fotocamera e funziona effettuando uno switch tra le due fotocamere. Le fotocamera anteriore monitora il blinks rate del conducente e la posa della testa per determinare la sonnolenza. La fotocamera posteriore tiene conto di misure basate sul veicolo. Determina la distanza del veicolo con altri veicoli sulla strada per verificare se il conducente è vicino ad altri veicoli e per verificare se si sta cambiando corsia. Carsafe ha una precision dell'83% e una recall del 75%.

Sahayadhas et al [6], nel 2013 hanno usato l'EOG¹ per monitorare i movimenti oculari. I dati ricavati dal EOG vengono quindi utilizzati per rilevare la sonnolenza del conducente. I ricercatori hanno sfruttato le misure basate su segnali fisiologici per rilevare la sonnolenza che risultano essere più accurate e affidabili in quanto utilizzano informazioni sullo stato interno del conducente. Esistono varie altre misure fisiologiche, ad es. EEG (elettroencefalografia), ECG (elettrocardiografia), EMG (elettromiografia) che potrebbero anche essere utilizzati per migliorare ulteriormente l'efficienza del sistema.

¹ Segnale elettrooculogramma che viene registrato per individuare i movimenti degli occhi utili nella stadiazione del sonno. La registrazione si basa sulla differenza di potenziale esistente tra cornea (positiva) e retina (negativa) e viene effettuata mediante elettrodi.

A. Rahman et al [7], nel 2015 hanno presentato un nuovo algoritmo per rilevare la sonnolenza in real-time. L'algoritmo acquisisce il volto e tramite l'algoritmo Viola Jones Cascade Classifier si effettua il rilevamento del volto e degli occhi. Viola Jones utilizza le feature di Haar per il rilevamento degli occhi. Dopo aver ritagliato la regione dell'occhio, individua due angoli degli occhi e un punto sulla palpebra inferiore. Per determinare se l'occhio è chiuso o meno viene calcolato il punto medio dei due angoli superiori degli occhi, e successivamente viene calcolata la distanza dal punto individuato sulla palpebra inferiore. Se tale distanza è zero o vicina allo zero, l'occhio viene classificato come chiuso. Inoltre, se l'occhio rimane chiuso costantemente per 2 o più secondi, si presume che il conducente sia assonnato e viene attivato un allarme. Questa tecnica fornisce risultati estremamente precisi se utilizzata in buone condizioni di illuminazione ed eseguita utilizzando una fotocamera ad alta risoluzione (è stata utilizzata una camera da 16MP).

3. Dataset

Dataset che contengono immagini degli occhi sono generalmente registrati in buone condizioni e contengono immagini ad alta risoluzione, che li rendono adatti per il rilevamento della pupilla, il rilevamento dell'iride, il monitoraggio degli occhi o il rilevamento dello sguardo.

Per questo progetto è stato utilizzato un dataset con un elevato numero di immagini di occhi catturate da diverse fotocamere a infrarossi (NIR) (ad es. Fotocamere Intel Realsense, IDS Imaging).

Il dataset utilizzato si chiama MRL Eye Dataset [8] e risulta essere scaricabile gratuitamente in rete. Questo set di dati contiene immagini a infrarossi in bassa e alta risoluzione, tutte catturate in varie condizioni di luminosità e da diversi dispositivi.

Per ottenere le immagini degli occhi, nella prima fase è stato effettuato un crop manuale di diverse migliaia di regioni oculari da diverse immagini NIR; le immagini di input di esempio sono presentate in Figura 1.



Figura 1 Esempi di immagini catturate da una telecamera a infrarossi per auto.

Nella seconda fase, sono state utilizzate le immagini ritagliate manualmente per addestrare il rivelatore oculare basato sull'istogramma dei gradienti orientati² combinato con il classificatore SVM. Questo rilevatore è stato utilizzato per estrarre automaticamente le regioni oculari. L'esempio del rilevamento della regione dell'occhio è mostrato in Figura 2. Dopo la fase di rilevamento, è stata accuratamente controllata ciascuna regione rilevata e sono stati rimossi i falsi positivi. In totale sono state create e controllate 85.000 immagini di occhi di diverse persone (37 persone diverse) catturate in varie condizioni di illuminazione; il set di dati contiene immagini di diversa qualità con proprietà diverse. Successivamente, verranno mostrati gli esempi di occhi che possono essere trovati nel dataset utilizzato.



Figura 2 Esempi di rilevamento degli occhi utilizzando il rilevatore HOG-SVM che è stato creato per il rilevamento automatico della regione degli occhi.

Ad esempio, gli occhi delle persone con gli occhiali sono mostrati nella prima fila della Fig. 3. I problemi che spesso si verificano con gli occhiali sono i riflessi. Nel dataset, vengono fornite molte immagini con il riflesso. Inoltre, per ogni immagine dell'occhio, sono state fornite informazioni sulla quantità di riflesso in essa presente (basata sulla grandezza dell'area riflessa). In particolare, ogni immagine rientra in una delle seguenti classi: nessun riflesso (la prima riga in Fig. 3), riflesso piccolo (la seconda riga in Fig. 3) e riflesso elevato (la terza riga in Fig. 3).

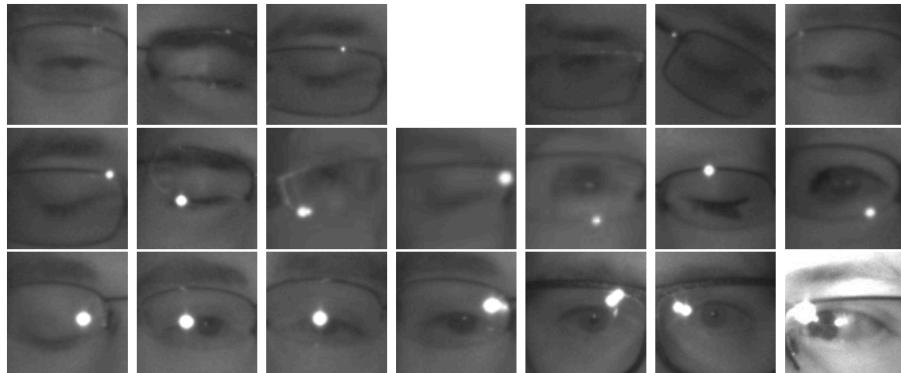


Figura 3 Esempi di occhi con occhiali e riflessi inclusi nel set di dati utilizzato.

² È un descrittore di caratteristiche usate in computer vision ed in elaborazione delle immagini per il riconoscimento di oggetti.

In Figura 4, gli esempi di set di dati mostrano che i riflessi possono anche verificarsi senza occhiali (ad esempio nella sclera, nella pupilla e nell'iride).



Figura 4 Esempi di riflessi oculari senza occhiali inclusi nel set di dati proposto.

In generale, molte donne usano ciglia e sopracciglia sintetiche, che possono causare il problema ai riconoscitori della direzione dello sguardo e dello stato degli occhi. Pertanto, nel set di dati, sono state fornite anche le informazioni sul genere di ogni persona.

3.1 Annotazioni e statistiche del dataset

Di seguito sono riportate le caratteristiche del dataset utilizzato:

- **ID soggetto:** nel set di dati, sono presenti i dati di 37 persone diverse (33 uomini e 4 donne);
- **ID immagine:** il set di dati è composto da 84.898 immagini;
- **genere [0 - uomo, 1 - donna]:** il set di dati contiene le informazioni sul genere della persona in ciascuna immagine (uomo, donna);
- **occhiali [0 - no, 1 - sì]:** le informazioni riguardanti la presenza o meno degli occhiali nell'immagine sono fornite per ogni immagine (con e senza gli occhiali); 24.001 immagini presentano gli occhiali;
- **stato dell'occhio [0 - chiuso, 1 - aperto]:** questa proprietà contiene le informazioni su i due stati oculari (41.945 chiusi e 42.953 aperti);
- **riflesso [0 - nessuno, 1 - piccolo, 2 - grande]:** sono stati annotati tre stati di riflessione in base alla dimensione del riflesso (66.227 nessun riflesso, 5.962 riflesso piccolo e 12.709 riflesso elevato);
- **condizioni di illuminazione [0 - pessime, 1 - buone]:** in base alla quantità di luce durante l'acquisizione dei video (53.630 pessime, 31.268 buone);
- **ID sensore [01 - RealSense, 02 - IDS, 03 - Aptina]:** il set di dati contiene le immagini catturate da tre diversi sensori (sensore Intel RealSense RS 300 con risoluzione 640×480 , sensore IDS Imaging con risoluzione 1280×1024 e sensore Aptina con risoluzione 752×480).

Di seguito sono riportati alcuni grafici che graficamente mostrano la distribuzione di alcune delle caratteristiche analizzate:

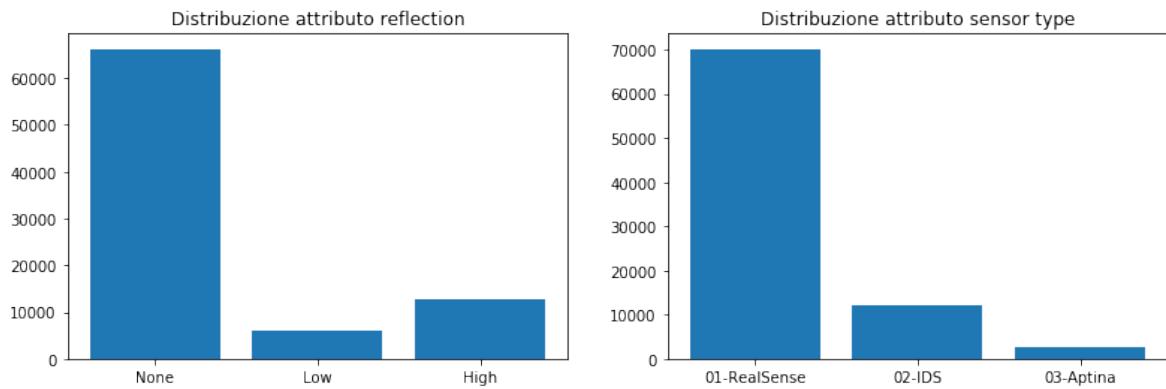


Figura 5 Distribuzione reflection a sinistra e distribuzione sensor type a destra.

L'attributo eye state, rappresentante l'etichetta del dataset, presenta la seguente distribuzione:

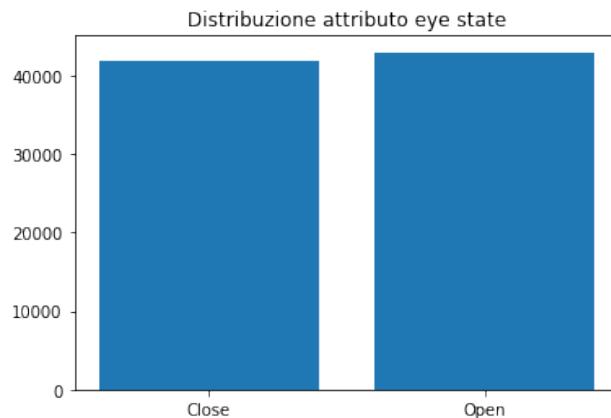


Figura 6 Distribuzione eye state.

Come si può osservare dalla Figura 6, il dataset presenta una distribuzione pressoché identica per quanto riguarda lo stato degli occhi.

In sintesi, il set di dati contiene immagini di occhi a bassa risoluzione, immagini con riflessi negli occhi o con riflessi sugli occhiali causati dall'illuminatore IR posto di fronte alla persona. Sono incluse anche alcune immagini oculari in cui la testa non è orientata verso la fotocamera. La varietà dei tipi di immagini può rendere più difficile il rilevamento di parti oculari, ma permette di addestrare un modello che sia più robusto ad eventuali variazioni di posa e illuminazione.

4. Modello preliminare

Nella seguente sezione verrà descritto il lavoro svolto per la creazione di un primo modello basato sulle CNN realizzate basate su lavori già svolti [9].

Le Convolutional Neural Networks (**CNN**) sono una famiglia di reti neurali largamente impiegate nell'ambito di computer vision e, più generalmente, con dati che hanno relazioni spaziali.

Se dovessimo rappresentarle graficamente, quello che ne verrebbe fuori è un grafo con una serie di passi dove, partendo da un input complesso, vengono estratte man mano informazioni, chiamate **features**, sempre più semplici e facilmente riconoscibili dal computer.

Le CNN seguono quindi un'architettura a livelli, tipicamente non ciclica. I livelli più importanti sono i **layer di convoluzione**, da cui prende il nome.

La convoluzione è un'operazione matematica che ricorda il funzionamento di uno scanner. Il risultato è funzione della traslazione di una funzione sull'altra, al variare del tempo. Poichè tipicamente le CNN ricevono in input dei dati bidimensionali, come ad esempio le immagini, avremo una o più matrici in input ed una chiamata kernel che verrà traslata al fine di calcolarne il risultato, chiamato feature map. Il passo di traslazione prende il nome di stride. A seguito di un layer di convoluzione, è comune trovare un layer di sotto-campionamento che riduca le dimensioni delle matrici in output. Le operazioni sono più comuni sono:

- **MaxPooling**: ogni cella della matrice risultante contiene il valore massimo della sottomatrice di partenza.
- **AveragePooling**: il valore di ciascuna cella è la media dei valori della sottomatrice di partenza.

Infine, ritroviamo i **layer full connected** che possono essere visti come dei vettori di neuroni. L'output di ciascun neurone è inviato a tutti i neuroni del layer successivo, con diversi valori di peso.

Il nostro obiettivo è stato quello di ottenere una base sulla quale effettuare delle analisi, descritte nei paragrafi successivi.

Il codice utilizzato per effettuare le operazioni descritte è riportato nel file Jupyter Notebook *Drive Drowsiness Detection.ipynb*.

4.1 Caricamento dataset e Pre-processing

Il primo task effettuato riguarda il caricamento delle immagini e il loro pre-processing.

La lettura delle immagini è stata effettuata tramite il metodo **imread** della libreria OpenCV. Successivamente, si è effettuato il resize (**cv2.resize**) e il reshape delle immagini in modo tale che quest'ultime avessero tutte la stessa dimensione. In particolare, la dimensione scelta è stata (24, 24, 1) dove i primi due numeri indicano l'altezza e la larghezza dell'immagine, mentre il terzo indica il numero di canali della stessa. Poiché le immagini sono state lette in scala di grigi, il terzo numero risulta essere uno.

Si è poi provveduto all'estrazione delle feature "genere", "occhiali", "riflesso", "condizione di illuminazione" e "tipo di sensore" oltre che delle etichette "stato dell'occhio" dal nome dell'immagini, creando un vettore che contenesse le singole feature per ogni immagine.

Di seguito è riportato il confronto tra un'immagine originale del dataset e la relativa immagine ottenuta dopo averne effettuato il caricamento e il pre-processing.



Figura 7 Sulla sinistra l'immagine originale e sulla destra l'immagine (24, 24, 1) in scala di grigi caricata.

Infine, come ultima operazione è stata effettuata una normalizzazione delle immagini dividendo i pixel di ogni immagine per 255, in modo tale che questi risultassero avere valori tra 0 e 1. Con tale operazione i valori dei pixel risultano essere più piccoli (si noti che questi valori piccoli rappresentano ancora l'immagine originale), con il beneficio che il tempo richiesto dal modello per convergere si riduce significativamente.

Inoltre, come sarà mostrato nei successivi paragrafi, il lavoro non si è limitato alla creazione di modelli che considerassero soltanto le immagini, ma verranno considerate anche tutte le altre feature di cui dispone il dataset.

Precedente alla creazione del modello, è stata l'operazione di splitting dei dati in training set e test set, ottenuto tramite la funzione **train_test_split** di Scikit Learn. Si è deciso di utilizzare l'80% (67918 immagini) dei dati come training set e il restante 20% (16980 immagini) come test set.

Inoltre, dopo aver effettuato lo splitting, le immagini dei due set sono state salvate nelle rispettive cartelle “Training” e “Test”, in modo tale da prevederne un rapido caricamento nelle successive esecuzioni.

4.2 Struttura

Il modello di training utilizzato è rappresentato da una CNN, basata sulla libreria Keras. La scelta è ricaduta sulla tipologia di rete CNN in quanto essa viene spesso applicata alla analisi di immagini, su cui il progetto di Drive Drowsiness Detection si basa. La struttura mostrata di seguito riguarda il primo modello creato.

Il modello creato è costituito da:

Input Layer	Input: immagini 24 x 24 in scala di grigi (1 canale)
Convolutional layer (CL 1)	Output filters: 32 Kernel size: 3 x 3 Activation: ReLU
Pooling layer (PL 1)	Pool size: 1 x 1
Convolutional layer (CL 2)	Output filters: 32 Kernel size: 3 x 3 Activation: ReLU
Pooling layer (PL 2)	Pool size: 1 x 1
Convolutional layer (CL 3)	Output filters: 64 Kernel size: 3 x 3 Activation: ReLU
Pooling layer (PL 2)	Pool size: 1 x 1
Dropout Layer	Percentage: 25 %
Flatten Layer	
Fully connected layer (FCL 1)	Output: 128 nodi Activation: ReLU
Dropout Layer	Percentage: 50 %
Fully connected layer (FCL 4)	Output: 2 nodi Activation: Softmax

Tabella 1 Struttura del modello con le sole immagini

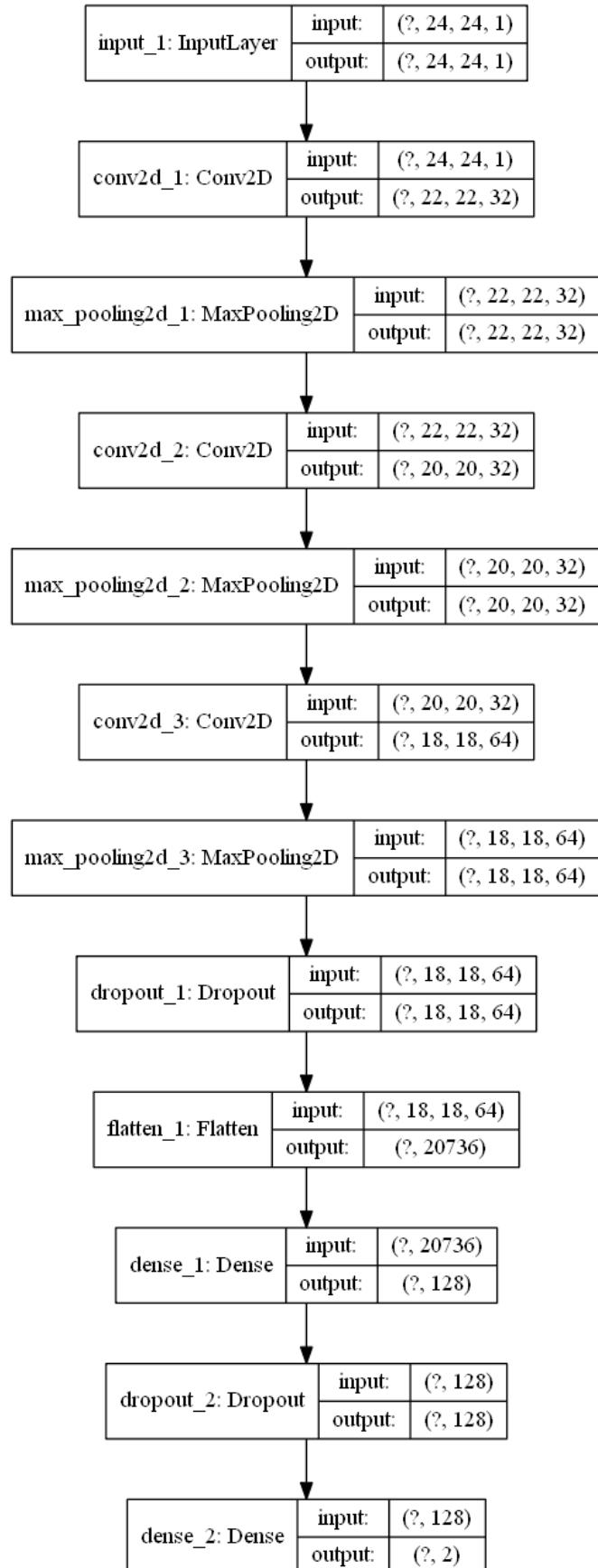


Figura 8 Struttura del modello creato

Tale modello considera come input le sole immagini. Volendo considerare anche altre features di cui il dataset dispone, si rende necessaria una modifica alla rete. Dopo il livello di Flatten, infatti, viene aggiunto un nuovo livello, che permette di concatenare le immagini con le features, appositamente inserite in un nuovo livello di Input separato. Tale operazione verrà descritta nella sezione 5.

Una volta definita la struttura del modello, questo si può creare tramite la funzione **Model** di TensorFlow, che permette successivamente di compilarlo. Per i parametri della compilazione, si è deciso di optare per l'ottimizzatore Adam (computazionalmente molto efficiente), mentre si è scelta la *sparse_categorical_crossentropy* come loss function. La cross entropy è una loss function, utilizzata per misurare la dissomiglianza tra la distribuzione delle etichette di classe osservate e le probabilità previste dell'appartenenza alla classe. L'uso della cross entropy categorica sparsa con 2 classi restituisce un risultato finale compreso tra 0 e 1, che è esattamente ciò di cui si necessita per il progetto.

4.3 Training modello

Una percentuale del training set è stata utilizzata come validation set. Quest'ultimo cambia in modo randomico ad ogni epoca, permettendo così di considerare diversi dati su cui valutare il modello ed ottenere quindi una misura più precisa della bontà dello stesso. In particolare, si è utilizzato il parametro *validation_split* del metodo fit assegnandogli il valore 0.1, assegnando al validation set, in tal modo, una grandezza pari al 10% del training set. Per ridurre l'overfitting del modello si è utilizzata la funzione EarlyStopping di Keras mostrata di seguito:

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=2, min_delta=0.0001,
                                             verbose=1, baseline=None, restore_best_weights=True)
```

Figura 9 Funzione EarlyStopping offerta da Keras.

- **monitor**: indica il valore da monitorare;
- **patience**: indica il numero di epoche senza alcun miglioramento dopo il quale l'allenamento verrà interrotto.
- **min_delta**: indica la variazione minima della quantità monitorata per poterla qualificare come un miglioramento, ovvero una variazione assoluta inferiore a min_delta, non verrà considerata come miglioramento.
- **verbose**: livello di verbosità;
- **baseline**: valore di base per la quantità monitorata. Il training si interromperà se il modello non mostra miglioramenti rispetto al valore di base.
- **restore_best_weights**: indica se ripristinare i pesi del modello dell'epoca con il miglior valore della quantità monitorata. Se settato a False, vengono utilizzati i pesi del modello ottenuti nell'ultimo step del training.

I risultati da questo ottenuti, come l'accuracy e la loss sul training e validation set sono stati salvati in un file testuale specifico per il modello. Inoltre, sono stati creati dei grafici raffiguranti l'andamento della loss e dell'accuracy all'aumentare del numero delle epoche.

Il modello è stato stoppato automaticamente dopo solo 14 epoche per evitare l'overfitting.

La Tabella 2 mostra i risultati registrati sul modello addestrato al termine della fase di training.

Nome	Dim. Immagini	Epoche	Train loss	Train accuracy	Val accuracy	Val loss
img_14_0.25_3CL _0.2	24 x 24	14	0.064	0.976	0.982	0.048

Tabella 2 Risultati del training del modello

I risultati nella *Tabella 2* indicano che il modello presenta un'accuracy sul validation set (6792 immagini) del 98% circa. Inoltre, il tempo medio impiegato per ogni epoca del training è di circa 10 secondi.

In Figura 10 sono riportati i grafici che descrivono l'andamento della loss e dell'accuracy all'aumentare del numero di epoch.

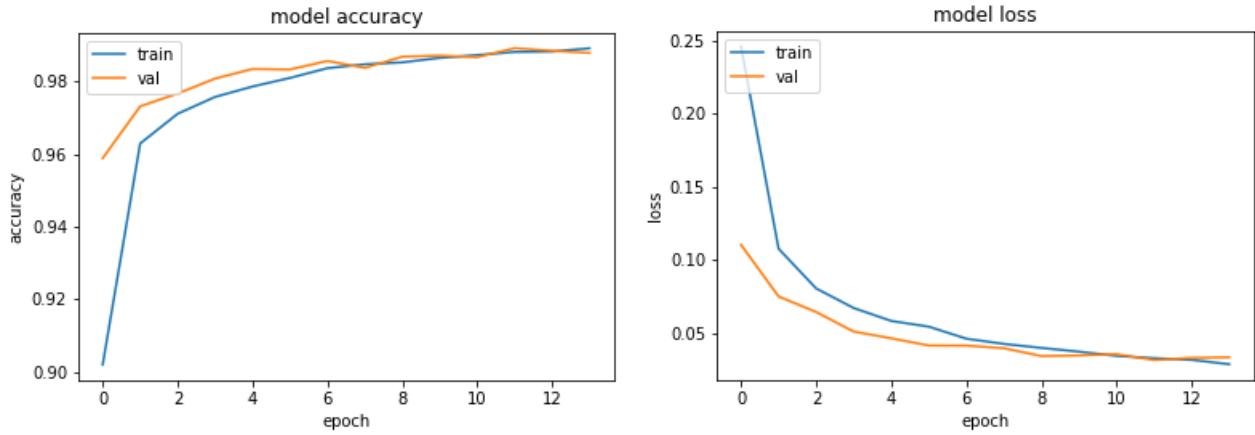


Figura 10 Andamento di accuracy e loss del modello descritto nel 4.

Dai grafici si può osservare come l'accuracy risulta essere superiore al 98% e la loss inferiore a 0.05. Inoltre, si nota come l'addestramento sia stato interrotto non appena le due curve abbiano iniziato a separarsi, e questo per evitare il fenomeno dell'overfitting.

Inoltre, dopo aver effettuato il training del modello si è provveduto al suo salvataggio per poterlo testare nella demo illustrata nella sezione 9. Il formato utilizzato per il salvataggio del modello è il formato HDF5³.

³ Hierarchical Data Format versione 5 (HDF5), è un formato di file open source che supporta dati di grandi dimensioni, complessi ed eterogenei.

4.4 Testing

Una volta effettuato il training del modello, questo è stato testato sul test set tramite la funzione evaluate di Keras. I risultati ottenuti dal modello sul test set (test loss e test accuracy) sono stati aggiunti al file testuale descritto precedentemente.

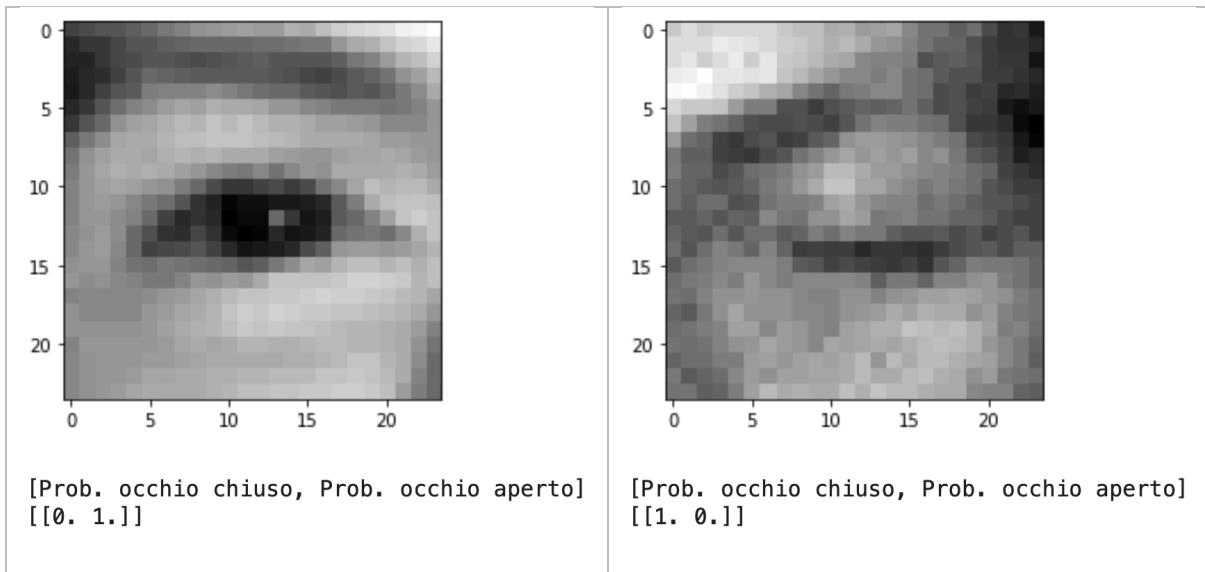
Nella *Tabella 3* sono riportati i risultati ottenuti dal modello dopo aver effettuato la fase di testing.

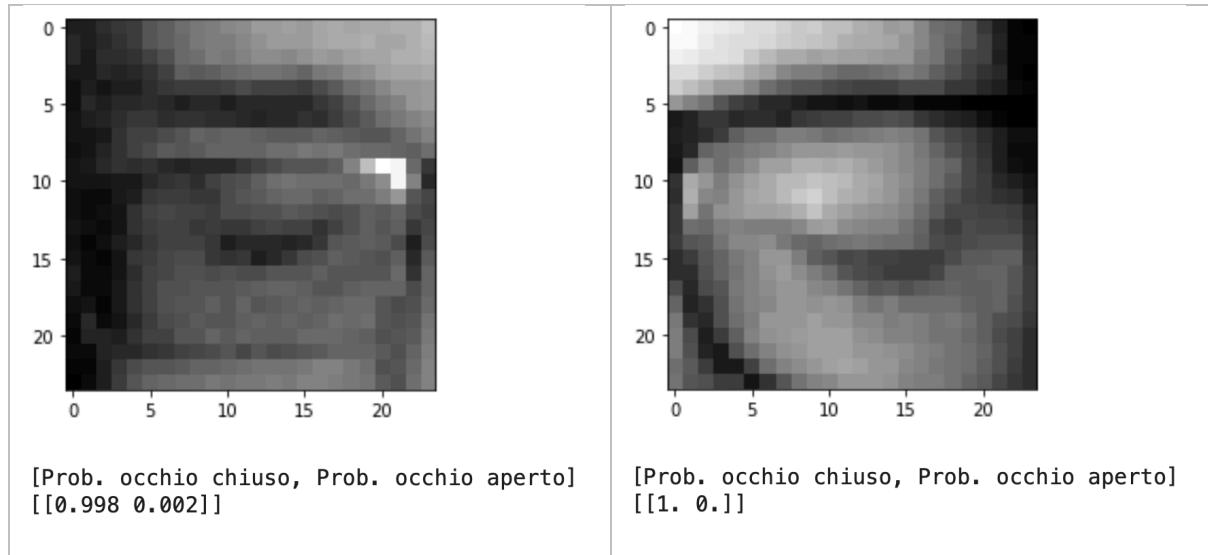
Nome	Epoche	Test loss	Test accuracy
img_14_0.25_3CL_0.2	14	0.040	0.986

Tabella 3 Risultati ottenuti sul test set.

Come si può notare dai dati in *Tabella 3*, la loss del modello sul test set è simile alla loss registrata sul validation set, quindi in linea con i risultati ipotizzati.

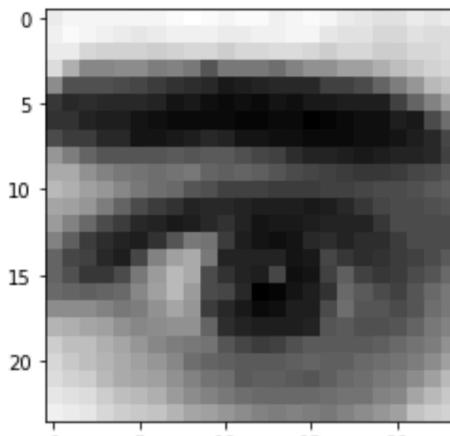
Vengono ora considerate le previsioni effettuate dal modello su alcune immagini del test set.



*Figura 11 Esempi di predictions.*

Come si può notare dai risultati, il modello risulta essere piuttosto preciso nonostante la risoluzione delle immagini sia di solo 24 x 24 pixel.

Infine, viene mostrato il risultato ottenuto con una immagine non facente parte del test set. Tale immagine è stata acquisita tramite la webcam del laptop. La risoluzione della webcam utilizzata è pari a 1280 x 720 px.

*Figura 12 Prediction su una immagine acquisita tramite webcam.*

In questo caso il modello ha classificato l'occhio come aperto con probabilità certa.

4.4.1 Confusion matrix

Per un ulteriore analisi dell'accuratezza ottenuta è stata calcolata la confusion matrix relativa al modello appena discusso. La matrice di confusione, detta anche tabella di errata classificazione, restituisce una rappresentazione dell'accuratezza di classificazione statistica. Ogni colonna della matrice rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali.

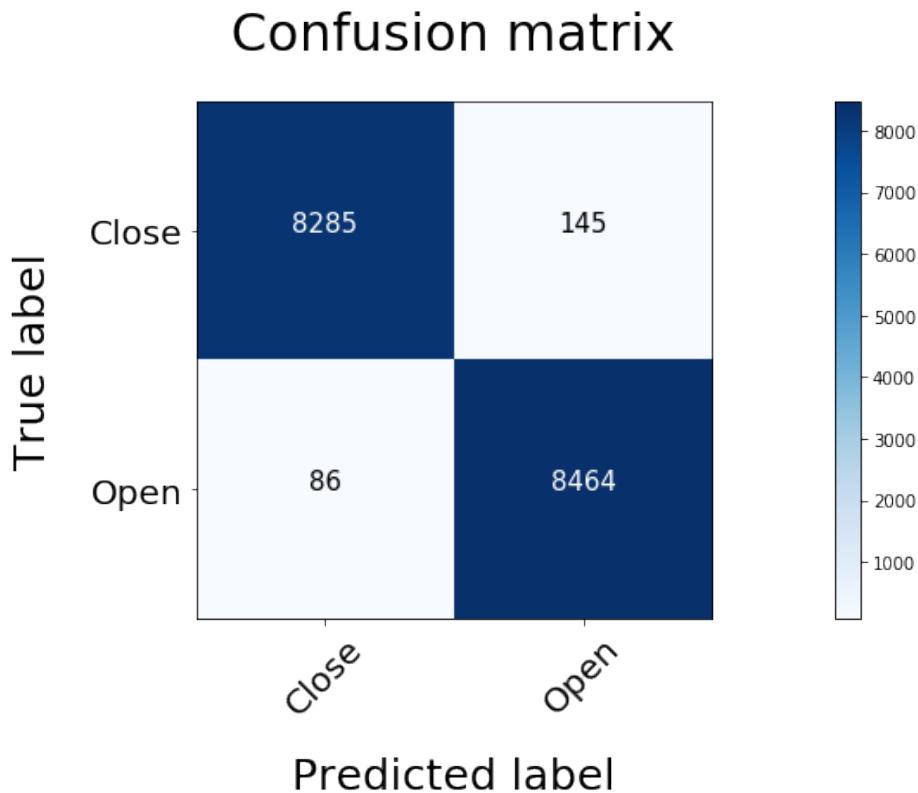


Figura 13 Confusion matrix.

Come si può notare dalla matrice illustrata in Figura 13, per 145 immagini su poco meno di 17000 immagini di test il modello ha predetto che l'occhio fosse aperto quando in realtà era chiuso, mentre per 86 immagini si è verificato l'opposto, ovvero il sistema ha predetto chiuso quando in realtà lo stato dell'occhio era aperto.

Pertanto, l'errore risulta essere piuttosto basso in quanto il sistema ha sbagliato a classificare soltanto 231 su un insieme di 16980 immagini.

Per completezza di seguito verranno mostrati i risultati relativi alla recall, alla precision e all'f1 - score.

4.4.2 Precision e Recall

Precision e **recall**, sono due comuni classificazioni statistiche. La precision può essere vista come una misura di *esattezza* o fedeltà, mentre la recall è una misura di *completezza*.

In un processo di classificazione statistica, la precision per una classe è il numero di veri positivi (il numero di oggetti etichettati correttamente come appartenenti alla classe) diviso il numero totale di elementi etichettati come appartenenti alla classe (la somma di veri positivi e falsi positivi, che sono oggetti etichettati erroneamente come appartenenti alla classe). La recall in questo contesto è definita come il numero di veri positivi diviso il numero totale di elementi che effettivamente appartengono alla classe (per esempio la somma di veri positivi e falsi negativi, che sono oggetti che non sono stati etichettati come appartenenti alla classe ma dovrebbero esserlo).

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Figura 14 Formula Recall e Precision.

Nella tabella di seguito sono riportati i risultati ottenuti:

Precision	0.9831
Recall	0.9899

Tabella 4 Risultati Precision e Recall.

4.4.3 F₁-score

Nell'analisi statistica della classificazione binaria, l'**F₁ score** (nota anche come **F-score** o **F-measure**, letteralmente "misura F") è una misura dell'accuratezza di un test. La misura tiene in considerazione precision e recall del test. L'F₁ viene calcolato tramite la media armonica di precision e recall:

$$F1\ Score = 2 * \frac{precision * recall}{precision + recall}$$

Figura 15 Formula F1-score.

Può assumere valori compresi fra 0 e 1. Assume valore 0 solo se almeno uno dei due vale 0, mentre assume valore 1 se sia precisione che recupero valgono 1.

Il valore ottenuto per il modello è di 0.9865.

5. Modifiche proposte

In questa sezione verranno illustrate le modifiche apportate al modello riportato nella sezione 4. Come prima operazione abbiamo verificato se un aumento delle dimensioni delle immagini passate in input al modello portasse ad un miglioramento delle prestazioni. Successivamente si è provveduto ad addestrare un modello che considerasse anche tutte le altre feature presenti nel dataset.

Si è deciso di aumentare le dimensioni delle immagini per osservare il comportamento del modello descritto precedentemente all'aumentare della risoluzione. In Figura 13 sono state riportate due immagini per mettere in evidenza la differenza di risoluzione.

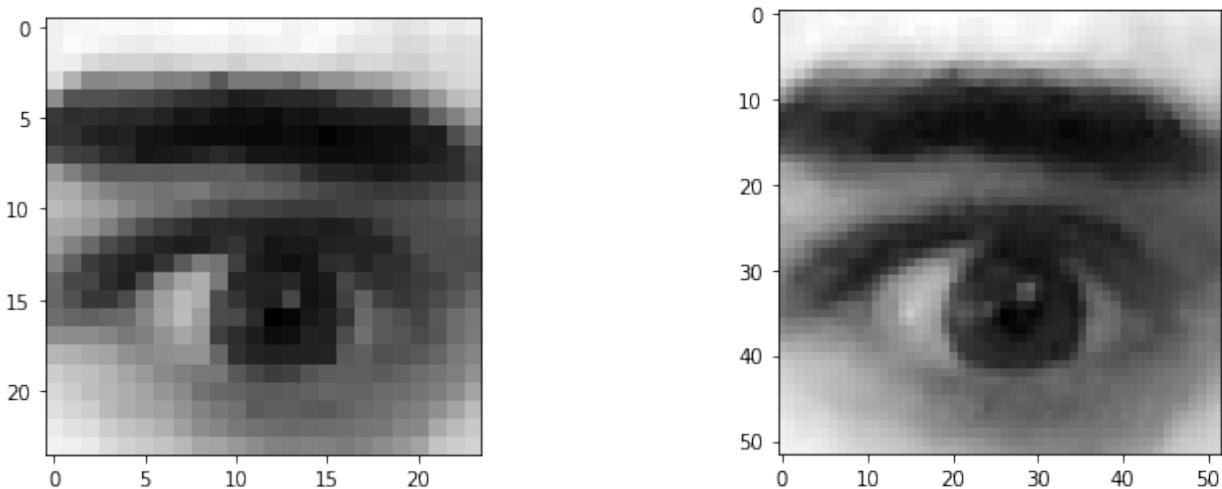


Figura 16 Sulla sinistra l'immagine 24 x 24, mentre sulla destra l'immagine 52 x 52.

Come si può osservare dalla Figura 16, l’immagine a destra presenta il doppio della risoluzione rispetto all’immagine sulla sinistra e questo chiaramente permette di avere più dettagli.

Per constatare come questi dettagli aggiuntivi vanno ad impattare sul modello si è resa necessaria la modifica del modello illustrato precedentemente. Quest’ultimo prendeva in input immagini 24 x 24, mentre adesso riceverà in input immagini 52 x 52.

La prima modifica effettuata riguarda l’operazione di resize delle immagini caricate, che ha comportato anche una modifica della dimensione dell’input della rete neurale.

Pertanto, il nuovo modello presenta una struttura simile a quella mostrata in Figura 8, l’unica differenza riguarda le dimensioni dell’input layer.

Effettuando l’addestramento del modello con immagini 52 x 52 si è riscontrato un aumento dei tempi richiesti per ogni epoca, dovuto alle maggiori dimensioni delle immagini. Infatti, il tempo medio richiesto per ogni epoca è passato da circa 10 secondi richiesti per un’immagine 24 x 24 a circa 52 secondi per un’immagine 52 x 52.

Per quanto concerne i risultati ottenuti alla fine della fase di training, essi sono riportati nella tabella seguente, nella quale sono presenti anche i risultati ottenuti sul modello che considera le immagini con dimensione 24 x 24.

Nome	Dim. Immagini	Epoche	Train loss	Val loss	Train accuracy	Val accuracy
img_14_0.25_3CL_0.2	24 x 24	14	0.064	0.048	0.976	0.982
img_15_0.25_3CL_0.2	52 x 52	15	0.068	0.056	0.974	0.980

Tabella 5 Confronto modelli

Dai dati riportati in Tabella 5 si può notare come i risultati ottenuti con il nuovo modello siano quasi identici a quelli registrati sul modello precedente. Pertanto, l’aver aumentato le dimensioni delle immagini ha portato ad un aumento dei tempi richiesti per il training, non giustificato dai risultati ottenuti.

Dopo aver constatato che la dimensione dell’immagine non ha un grosso impatto sui risultati ottenuti si è provato a considerare tutte le feature presenti nel dataset. In questo modo la rete durante l’addestramento considererà delle informazioni aggiuntive di cui non disponeva, e sarà interessante osservare come tali informazioni influenzino il risultato.

Da sottolineare che i lavori precedenti illustrati nella sezione 2, non hanno effettuato un’operazione di questo tipo, in quanto si sono limitati all’utilizzo delle sole immagini acquisite per l’individuazione dello stato dell’occhio.

Di seguito quindi, verranno illustrate le modifiche che si sono rese necessarie per far sì che il modello durante l’addestramento considerasse anche altre feature oltre alle immagini.

Infine, verranno discussi i risultati ottenuti su tale modello.

5.1 Struttura

Per considerare anche le seguenti feature:

- **genere**
- **occhiali**
- **riflesso**
- **condizioni di illuminazione**
- **ID sensore**

è necessario modificare la struttura del modello in modo tale da inserire un ulteriore layer chiamato *concatenate layer*. Tale layer si occupa di unire l'output dell'operazione di flatten con un nuovo input layer del modello che è rappresentato delle cinque feature elencate sopra.

Di seguito è mostrata la struttura del modello dopo aver introdotto il concatenate layer e un nuovo input layer.

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	(None, 24, 24, 1)	0	
conv2d_4 (Conv2D)	(None, 22, 22, 32)	320	input_3[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 22, 22, 32)	0	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 20, 20, 32)	9248	max_pooling2d_4[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 20, 20, 32)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 18, 18, 64)	18496	max_pooling2d_5[0][0]
max_pooling2d_6 (MaxPooling2D)	(None, 18, 18, 64)	0	conv2d_6[0][0]
dropout_3 (Dropout)	(None, 18, 18, 64)	0	max_pooling2d_6[0][0]
flatten_2 (Flatten)	(None, 20736)	0	dropout_3[0][0]
input_4 (InputLayer)	(None, 5)	0	
concatenate_2 (Concatenate)	(None, 20741)	0	flatten_2[0][0] input_4[0][0]
dense_3 (Dense)	(None, 128)	2654976	concatenate_2[0][0]
dropout_4 (Dropout)	(None, 128)	0	dense_3[0][0]
dense_4 (Dense)	(None, 2)	258	dropout_4[0][0]

Tabella 6 Struttura del modello in forma tabellare

Dalla Tabella 6 si può notare che dopo l'operazione di flatten sono stati aggiunti due nuovi layer, ovvero un input layer composto da cinque elementi e il concatenate layer che si occupa di concatenare le nuove feature all'output del flatten.

Per una visione del nuovo modello creato si rimanda alla Figura 17 illustrata di seguito.

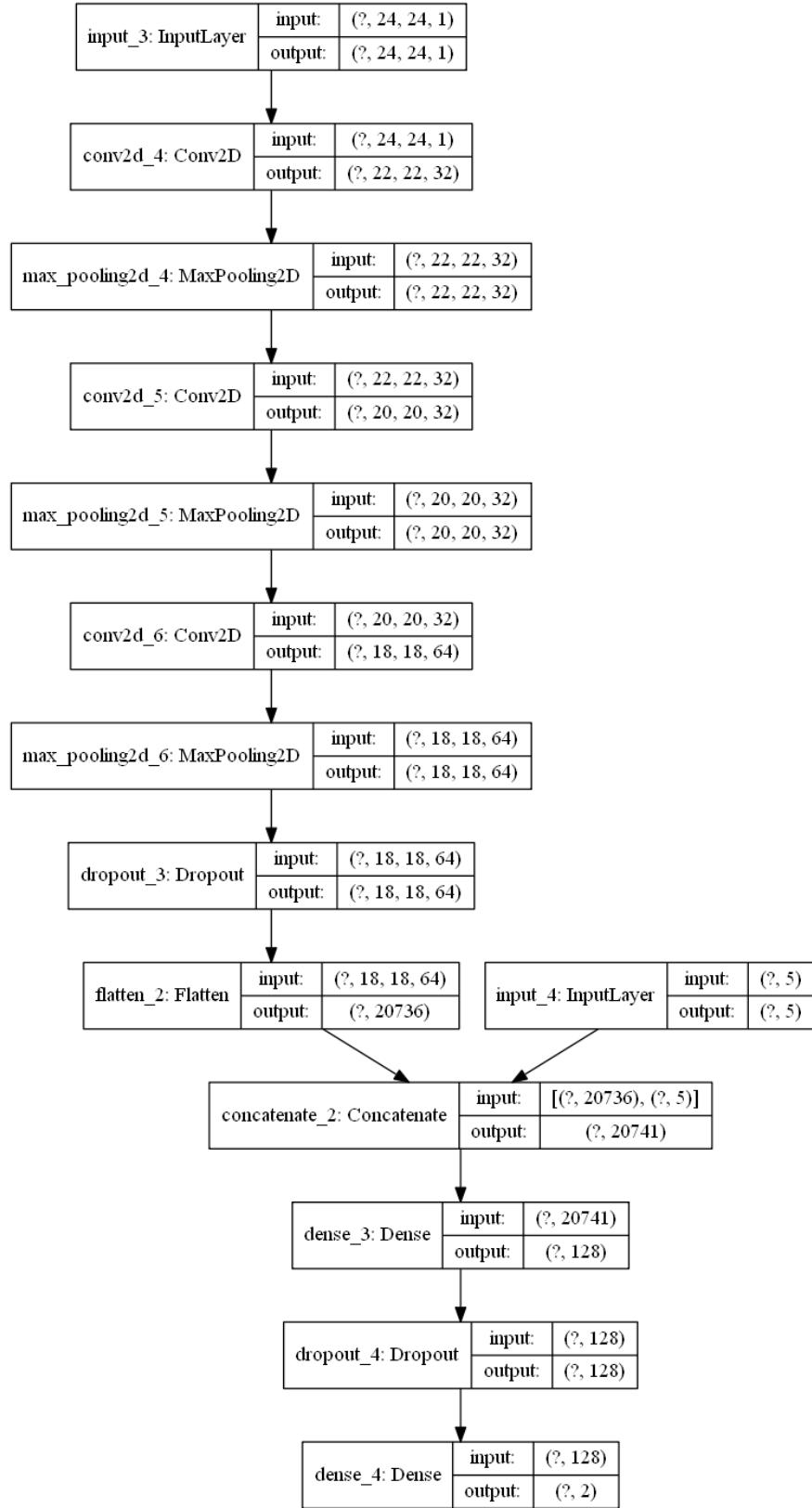


Figura 17 Struttura del modello che considera anche le altre feature del dataset

5.2 Training

Nella tabella di seguito sono riportati i risultati registrati dopo aver effettuato l'addestramento del modello illustrato in Figura 17.

Nome	Dim. Immagini	Epoche	Train loss	Val loss	Train accuracy	Val accuracy
all_14_0.25_3CL_0.2	24 x 24	14	0.057	0.044	0.987	0.984

Tabella 7 Risultati del training.

Considerando anche altre informazioni oltre alle sole immagini il tempo medio richiesto per eseguire un'epoca è di circa 11 secondi. Pertanto, il modello appena descritto risulta impiegare mediamente un secondo in più per ogni epoca rispetto al modello descritto nella sezione 4. Di seguito sono riportati l'andamento della loss e dell'accuracy all'aumentare delle epoche.

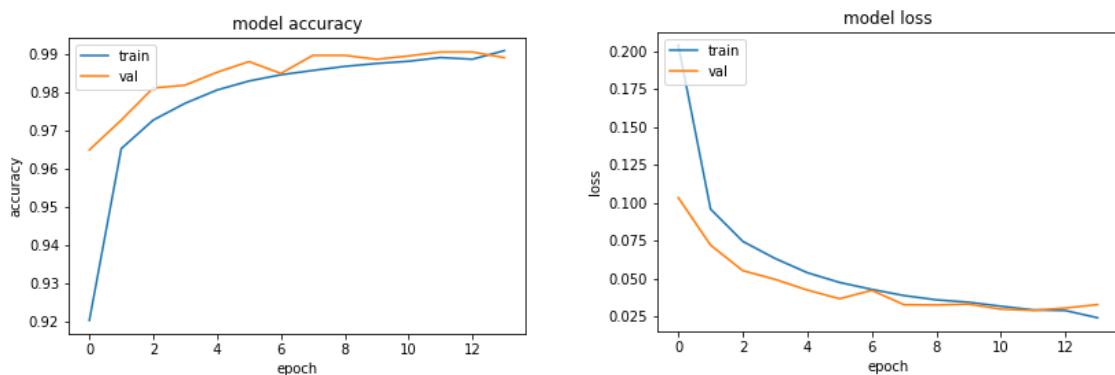


Figura 18 Andamento di accuracy e loss del modello descritto nel 5.

Osservando i grafici mostrati in Figura 18, si può notare la somiglianza che vi è con quelli mostrati in Figura 10. Anche in questo caso, l'accuracy risulta essere circa del 99%, mentre la loss è circa 0.04. Inoltre, si può notare come anche in questo caso l'addestramento sia stato interrotto non appena le due curve iniziarono a separarsi.

5.3 Testing

Di seguito, infine, sono riportati i risultati che il modello ha fatto registrare sul test set.

Nome	Dim. Immagini	Epoche	Test loss	Test accuracy
all_14_0.25_3CL_0.2	24 x 24	14	0.041	0.985

Tabella 8 Risultati ottenuti sul test set.

Come si può notare anche in questo caso l'accuracy ottenuta risulta essere alta e la loss bassa. Inoltre, come ci aspettavamo, la loss e l'accuracy registrate sul test sono simili alla loss e all'accuracy registrate sul validation set.

6. Confronto tra i modelli

In tale sezione si discuterà delle valutazioni fatte sui modelli illustrati nelle sezioni precedenti, ovvero quello che considera solo le immagini e nessun'altra feature e quello che considera anche le altre feature oltre che alle sole immagini. Le valutazioni fatte per esso sono state effettuate anche per tutti gli altri modelli creati. I risultati sono riportati nella sezione 8.

Inoltre, si è provveduto a valutare i risultati ottenuti su due ulteriori modelli, che però sono stati addestrati su dataset differenti da quello da noi utilizzato.

Per poter valutare due o più modelli è necessario aver provveduto al salvataggio del modello dopo averne effettuato il training. La prima operazione effettuata consiste nel caricamento dei modelli sopraccitati tramite la funzione **load_model** che restituisce il modello compilato.

Prima di effettuare una comparazione dei modelli descritti nelle sezioni 4 e 5 si vuole mostrare cosa accade ad uno dei modelli creati quando si aumenta il numero di epoche.

Di seguito sono riportati i due grafici che descrivono l'andamento della loss e dell'accuracy al l'aumentare del numero di epoche. In questo caso sono state considerate 25 epoche.

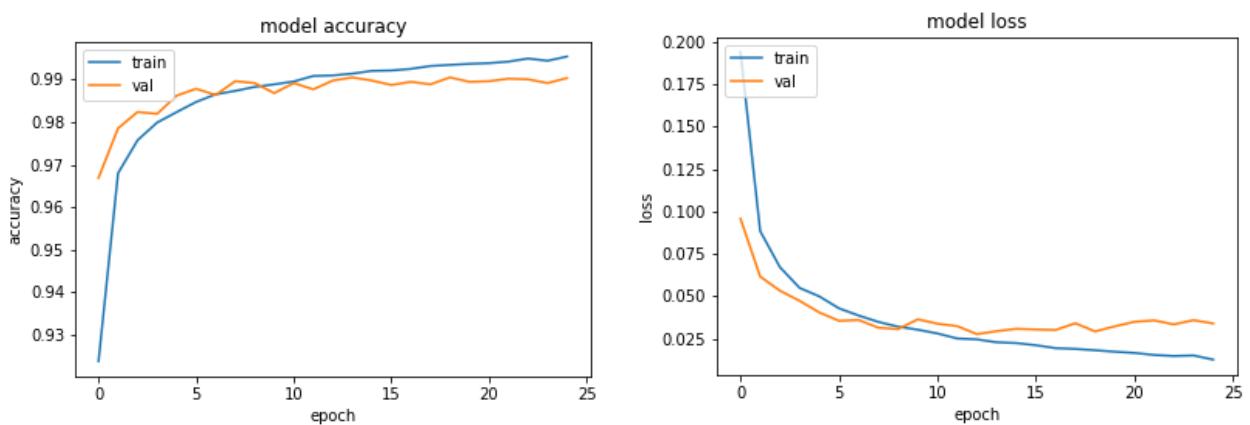


Figura 19 Andamento di accuracy e loss del modello descritto nel 6.

Come ci aspettavamo, in tutti e due i grafici mostrati in Figura 19 le due curve tendono a separarsi sempre più all'aumentare del numero del numero di epoche. Infatti, la curva di colore blu che rappresenta la loss e l'accuracy sul training tende a diminuire e aumentare rispettivamente al crescere del numero di epoche, mentre la loss e l'accuracy registrate sul validation set rimane stabile. Il fenomeno appena descritto ci indica che il modello si sta adattando troppo ai dati di training, pertanto non ha senso aumentare il numero di epoche in quanto nemmeno con un numero elevato di epoche otterremmo un miglioramento dei risultati sul validation set.

Nella tabella mostrata di seguito sono riportati i dati sul test set ottenuti dai due modelli citati precedentemente e di un nuovo modello, simile a quello descritto nella sezione 5, ma che presenta un numero di epoche maggiore.

Nome	Dim. Immagini	Epoche	Test loss	Test accuracy
img_14_0.25_3CL_0.2	24 x 24	14	0.040	0.986
all_14_0.25_3CL_0.2	24 x 24	14	0.041	0.985
all_25_0.25_3CL_0.2	24 x 24	25	0.048	0.988

Tabella 9 Confronto modelli.

Da una lettura della Tabella 9 si evince che i tre modelli presentano dei risultati simili sia sulla loss che sul training.

Si valuteranno ora i risultati che ottenuti utilizzando il nostro test set (costituito da circa 7000 immagini) su dei modelli addestrati con dataset differenti. Da sottolineare che i risultati che verranno mostrati sono stati registrati su modelli che presentano la stessa struttura della rete neurale convoluzionale illustrata nei paragrafi precedenti.

Uno dei due modelli è stato reperito sul sito sul quale è illustrata le rete neurale utilizzata per la creazione del modello preliminare [9], mentre l'altro modello è stato addestrato utilizzando un dataset ottenuto tramite l'unione di diversi campioni di dataset individuati in rete. L'unione di diversi insiemi di immagini ci ha permesso di ottenere un nuovo dataset contenente poco più di 5000 immagini.

Di seguito sono riportati i risultati registrati sul test set per entrambi i modelli citati.

Nome	Test loss	Test accuracy
Modello scaricato [9]	1.031	0.703
Modello con nuovo dataset	0.853	0.649

Tabella 10 Risultati modelli addestrati con un diverso training set.

Si può notare come la loss e l'accuracy ottenuta sul test set per entrambi i modelli sia superiore e inferiore rispettivamente ai risultati ottenuti sui modelli proposti precedentemente. Pertanto, dopo un'analisi di questo tipo è possibile affermare che il risultato finale risulta essere funzionale al dataset utilizzato per il training. Questo significa, che se il dataset è composto da immagini che ritraggono l'oggetto in analisi (nel nostro caso l'occhio) in diverse pose, illuminazioni e considerando un utente collaborativo e non collaborativo, allora il dataset risulterà essere particolarmente robusto e il modello potrà effettuare il training considerando un'ampia varietà di immagini differenti.

Informazioni più dettagliate su come deve essere strutturato un dataset per essere piuttosto robusto sono riportate nella sezione 3.

7. Data Profiling

Successivamente alla creazione del modello si è provveduto ad effettuare diverse operazioni di data profiling, come l'estrazione di dipendenze funzionali, dipendenze funzionali rilassate [10] e features selection. Tecniche di tale genere aiutano nella scoperta, comprensione e organizzazione dei dati. Operazione preliminare di tali processi è stata la creazione di due differenti csv. Questi sono stati utilizzati come input per i vari algoritmi di FD discovery e features selection.

Il primo file csv riportava tutte le features del dataset, cioè:

- Image_Name
- Glasses
- Genders
- Reflections
- Image_quality
- Sensor_type
- Eye_state

Il secondo file csv comprendeva tutte le features del dataset tranne l'attributo "Eye_state", che rappresenta la feature target del dataset.

Image_Name	Glasses	Genders	Reflections	Image_quality	Sensor_type	Eye_state
s0001_01883_0_0_1_0_0_01.png	0	0	0	0	1	1
s0001_00947_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_00979_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_02992_0_1_1_2_0_01.png	1	0	2	0	1	1
s0001_03036_0_1_1_2_0_01.png	1	0	2	0	1	1
s0001_00819_0_0_0_0_0_01.png	0	0	0	0	1	0
s0001_00827_0_0_0_0_0_01.png	0	0	0	0	1	0
s0001_03235_0_1_1_2_0_01.png	1	0	2	0	1	1
s0001_03008_0_1_1_2_0_01.png	1	0	2	0	1	1
s0001_01837_0_1_0_0_0_01.png	1	0	0	0	1	0
s0001_01206_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_01005_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_01865_0_0_1_0_0_01.png	0	0	0	0	1	1
s0001_02974_0_1_1_2_0_01.png	1	0	2	0	1	1
s0001_01403_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_01962_0_0_1_0_0_01.png	0	0	0	0	1	1
s0001_01739_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_02644_0_1_1_0_0_01.png	1	0	0	0	1	1
s0001_01102_0_1_0_2_0_01.png	1	0	2	0	1	0
s0001_03131_0_1_1_2_0_01.png	1	0	2	0	1	1
s0001_00920_0_0_0_0_0_01.png	0	0	0	0	1	0
s0001_01984_0_0_1_0_0_01.png	0	0	0	0	1	1

Figura 20 File CSV con etichette.

Image_Name	Glasses	Genders	Reflections	Image_quality	Sensor_type
s0001_01883_0_0_1_0_0_01.png	0	0	0	0	1
s0001_00947_0_1_0_2_0_01.png	1	0	2	0	1
s0001_00979_0_1_0_2_0_01.png	1	0	2	0	1
s0001_02992_0_1_1_2_0_01.png	1	0	2	0	1
s0001_03036_0_1_1_2_0_01.png	1	0	2	0	1
s0001_00819_0_0_0_0_0_01.png	0	0	0	0	1
s0001_00827_0_0_0_0_0_01.png	0	0	0	0	1
s0001_03235_0_1_1_2_0_01.png	1	0	2	0	1
s0001_03008_0_1_1_2_0_01.png	1	0	2	0	1
s0001_01837_0_1_0_0_0_01.png	1	0	0	0	1
s0001_01206_0_1_0_2_0_01.png	1	0	2	0	1
s0001_01005_0_1_0_2_0_01.png	1	0	2	0	1
s0001_01865_0_0_1_0_0_01.png	0	0	0	0	1
s0001_02974_0_1_1_2_0_01.png	1	0	2	0	1
s0001_01403_0_1_0_2_0_01.png	1	0	2	0	1
s0001_01962_0_0_1_0_0_01.png	0	0	0	0	1

Figura 21 File CSV senza etichette.

7.1 FD

La prima operazione è stata quella di ricercare l'esistenza di dipendenze funzionali all'interno del dataset. Tale operazione è stata effettuata tramite il tool COD3. Il file di input utilizzato per l'algoritmo è stato quello che non prevedeva il target attribute. In aggiunta, il file è stato modificato in modo da eliminare l'attributo Image_Name, non rilevante allo scopo.

Glasses	Genders	Reflections	Image_quality	Sensor_type	Eye_state
0	0	0	0	1	1
1	0	2	0	1	0
1	0	2	0	1	0
1	0	2	0	1	1
1	0	2	0	1	1
0	0	0	0	1	0
0	0	0	0	1	0
1	0	2	0	1	1
1	0	2	0	1	1
1	0	0	0	1	0
1	0	2	0	1	0
1	0	2	0	1	0
0	0	0	0	1	1
1	0	2	0	1	1
1	0	2	0	1	0
0	0	0	0	1	1
1	0	2	0	1	0
1	0	0	0	1	1
1	0	2	0	1	0
1	0	2	0	1	1
0	0	0	0	1	0
0	0	0	0	1	1
1	0	0	0	1	0
1	0	0	0	1	1
0	0	0	0	1	1
1	0	0	0	1	0
1	0	2	0	1	1

Figura 22 Struttura del file input per l'algoritmo COD3.

Applicando l'algoritmo per l'FD discovery su tale file non sono state rilevate dipendenze funzionali.

Tuttavia, esaminando i log del tool, si è notato che inizialmente erano state identificate quattro differenti dipendenze, che diventano non più valide oltre un certo numero di tupla.

```

170451 09:21:24> tupla> 34091 $
170452 09:21:24> Execution Time: COD3 - Preprocessing: 0 ms -> $
170453 09:21:24> Time> 2020/06/25 09:21:24 $
170454 09:21:24> Dipendenze totali finali: - incrementale> 4 $
170455 09:21:24> F1=182;F2=44;F3=144;F4=104;F5=183416 $
170456 09:21:24> tupla> 34092 $
170457 09:21:24> Execution Time: COD3 - Preprocessing: 0 ms -> 100.0 ns $
170458 09:21:24> Time> 2020/06/25 09:21:24 $
170459 09:21:24> Dipendenze totali finali: - incrementale> 0 $
170460 09:21:24> F1=197;F2=55;F3=159;F4=104;F5=183416 $
170461 09:21:24> tupla> 34093 $

```

Figura 23 Algoritmo COD3: Punto di terminazione di validità delle FD.

```

09:21:24> ( 3 ) -> ( 2 )
( 1 ) -> ( 2 )
( 5 ) -> ( 2 )
( 4 ) -> ( 2 )

```

Figura 24 Algoritmo COD3: Dipendenze rilevate fino alla tupla 34091.

Nello specifico, non avendo considerato l'attributo Image_Name, la corrispondenza degli attributi era la seguente:

Nome	Numero
Glasses	1
Genders	2
Reflections	3
Image_quality	4
Sensor_type	5

Tabella 11 Mapping delle features

Per cui, le corrispondenze trovate erano:

Glasses → Genders
 Reflections → Genders
 Image_quality → Genders
 Sensor_type → Genders

Si è quindi provveduto ad analizzare il file allo scopo di trovare il motivo della terminazione della validità della dipendenza.

	Glasses	Genders	Reflections	Image_quality	Sensor_type
34082	0	0	0	1	1
34083	0	0	0	1	1
34084	0	0	0	1	1
34085	0	0	0	0	1
34086	0	0	0	1	1
34087	0	0	0	1	1
34088	0	0	0	1	1
34089	0	0	0	1	1
34090	0	0	0	1	1
34091	0	0	0	0	1
34092	0	1	1	1	1
34093	0	1	0	1	1
34094	0	1	0	1	1
34095	0	1	0	1	1
34096	0	1	0	1	1
34097	0	1	0	1	1
34098	0	1	1	1	1
34099	0	1	0	1	1
34100	0	1	0	1	1
34101	0	1	0	0	1
34102	0	1	1	1	1
34103	0	1	0	0	1
34104	0	1	0	1	1
34105	0	1	0	1	1
34106	0	1	0	1	1
34107	0	1	0	1	1
34108	0	1	0	1	1
34109	0	1	0	1	1
34110	0	1	0	0	1

Figura 25 Dettaglio del file csv in cui si interrompe la validità delle dipendenze.

Come si può analizzare dalla **Errore. L'origine riferimento non è stata trovata.** nella tupla 34092 si ha la prima occorrenza del valore 1 per l'attributo "Genere". Ciò vuol dire che dalla tupla 1 alla tupla 34091 i valori di "Genere" erano tutti uguali, e pari a 0. Il cambiamento del valore di tale attributo ha provocato la terminazione della validità delle dipendenze.

Le dipendenze rilevate sono state considerate banali e non utili ai fini della feature selection, in quanto basate sulla considerazione di un valore univoco sulla parte destra.

7.2 RFD

Recentemente si è posto un interesse sulle cosiddette Relaxed Functional Dependencies [9], vale a dire delle dipendenze funzionali inesatte che rilassano su uno o più vincoli delle dipendenze funzionali canoniche. Non essendo state rilevate dipendenze funzionali canoniche sul dataset, si è provveduto a ricercare RFD che rilassavano sull'extent e sul confronto. A tal fine sono stati utilizzati gli algoritmi BIRD e DOMINO [11] per trovare le RFD che rilassavano sull' extent, con quest'ultimo utilizzato anche per rilevare le RFD che rilassavano sul comparison.

7.2.1 Extent

Rilassare sull'extent significa che la dipendenza funzionale vale per "quasi tutte" le tuple, o su un sottoinsieme di esse. Diversi sono i casi per cui una dipendenza potrebbe non valere per tutte le tuple, come la presenza di valori mancanti, differenti tipi di dati o errori presenti negli stessi.

Applicando l'algoritmo sul dataset considerato, è stata rilevata la seguente dipendenza funzionale approssimata:

$$(Genders \text{ Reflections} \text{ Sensor_type}) \rightarrow (Glasses)$$

con soglia di extent g3-error di 0.9.

Il g3-error corrisponde al minimo numero di tuple da rimuovere da una istanza di relazione r affinché una dipendenza funzionale $X \rightarrow Y$ valga. Nello specifico caso, ciò significa che, affinché la dipendenza trovata valga, sarebbe necessario eliminare il 90% delle tuple. Nonostante la soglia della dipendenza sia molto alta, si è effettuata la costruzione di un nuovo modello che considerasse tutte le features tranne la feature Glasses, che rappresenta la parte destra della dipendenza funzionale, quindi la feature che è possibile derivare dalle altre. La struttura del modello, a meno delle features utilizzate per il training, è la medesima del modello descritto nella sezione 5.

In Figura 26 sono mostrati i grafici riportanti l'andamento della loss e dell'accuracy al crescere del numero di epoch:

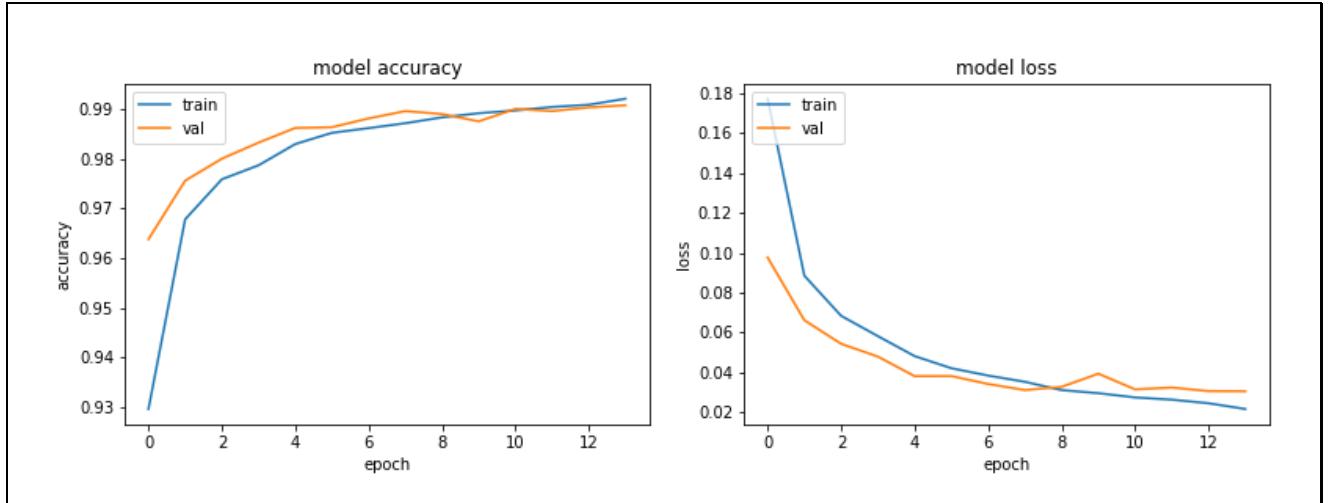


Figura 26 Andamento di accuracy e loss del modello descritto in 7.2.1.

I risultati del modello sono mostrati nella tabella 12:

Nome	Mean epoch training time	Training		Validation		Test	
		Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
all-glasses_14_0.25_3CL_0.2_24	10.926	0.051	0.980	0.043	0.984	0.032	0.989

Tabella 12 Risultati modello descritto in 7.2.1

Di seguito viene mostrato il confronto tra il modello discusso nel 6. e il modello discusso in questo paragrafo.

Nome	Mean epoch training time	Training		Validation		Test	
		Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
all-glasses_14_0.25_3CL_0.2_24	10.926	0.051	0.980	0.043	0.984	0.032	0.989
all_14_0.25_3CL_0.2_24	10.772	0.057	0.978	0.044	0.984	0.041	0.985
	+ 0.154	- 0.06	+ 0.002	-0.001	0	- 0.009	+ 0.004

Tabella 13 Confronto tra i modelli di 5. e 7.2.1

Come si può notare dalla tabella, i risultati ottenuti non variano di quantità rilevanti.

7.2.2 Confronto

Rilassare le dipendenze funzionali sul confronto significa usare paradigmi di matching approssimato per confrontare i valori dell'attributo sul lato sinistro e quelli del lato destro della dipendenza funzionale. Ciò permette di catturare relazioni semantiche tra gruppi di valori che sembrano essere simili, piuttosto che identici. Generalmente, questo tipo di RFD viene utilizzato per risolvere problemi di data quality, query optimization e knowledge discovery. Queste si basano su una soglia di comparison, che è la differenza massima ammissibile tra i valori di due tuple su un attributo.

Tramite l'algoritmo DOMINO, si è provato ad estrarre RFD basate sul confronto dal dataset utilizzato nel progetto. Nel fare ciò si sono utilizzate differenti soglie di comparison: 1, 2, 4 e 15.

Utilizzando una soglia di comparison pari ad uno non sono stati trovati risultati, vale a dire che non vi sono due attributi X e Y per cui la differenza massima tra i valori di X sia minore ad uno e la differenza massima tra i valori di Y sia minore ad 1.

Di seguito vengono riportati i risultati ottenuti con le diverse soglie:

SOGGLIA 2

***** DISCOVERED RFDs *****

0@2.0->0@2.0	cc:0.9685232103301251
1@2.0->0@2.0	cc:0.8694514821618544
0@2.0->1@2.0	cc:0.5990717281682707

SOGGLIA 4

***** DISCOVERED RFDs *****

0@4.0->0@2.0	cc:0.9685232103301251
1@4.0->0@2.0	cc:0.8694514821618544
0@4.0->1@2.0	cc:0.5990717281682707

SOGGLIA 15

***** DISCOVERED RFDs *****

0@15.0->0@2.0	cc:0.9685232103301251
1@15.0->0@2.0	cc:0.8694514821618544
0@15.0->1@2.0	cc:0.5990717281682707

Nonostante le tre diverse soglie, le RFD sul confronto e le cc⁴ (le soglie delle RFD minimali) rilevate dal tool sono le medesime:

$$0 \rightarrow 0$$

$$1 \rightarrow 0$$

$$0 \rightarrow 1$$

E cioè:

$$\text{Glasses} \rightarrow \text{Glasses}$$

$$\text{Genders} \rightarrow \text{Glasses}$$

$$\text{Glasses} \rightarrow \text{Genders}$$

Poichè la prima dipendenza è banale, ed il modello senza la feature Glasses è stato descritto nell' 7.2.1, si discuterà del modello sviluppato includendo tutte le features tranne Genders. La struttura del modello, a meno delle features utilizzate per il training, è la medesima del modello descritto nel 6.

In Figura 27 sono mostrati i grafici riportanti l'andamento della loss e dell'accuracy al crescere del numero di epoch:

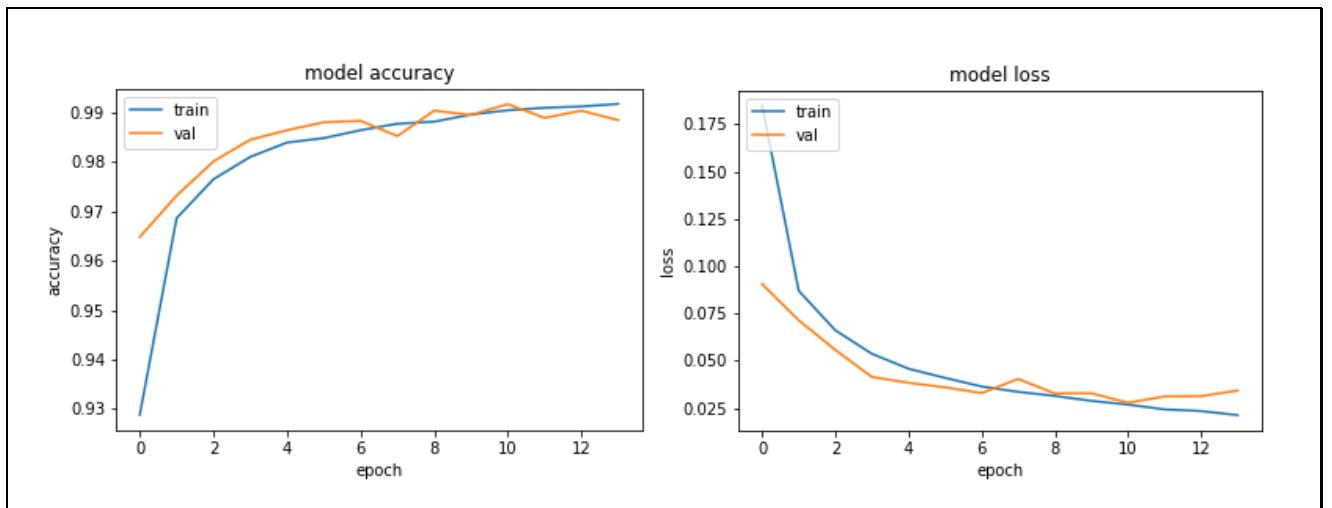


Figura 27 Andamento di accuracy e loss del modello descritto in 7.2.2.

⁴ Per i dettagli dell'algoritmo DOMINO, si faccia riferimento alla sezione 3.4 del [11]

I risultati del modello sono mostrati nella Tabella 14:

Nome	Mean epoch training time	Training		Validation		Test	
		Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
all-genders_14_0.25_3CL_0.2_24	10.995	0.050	0.981	0.042	0.985	0.040	0.987

Tabella 14 Risultati modello descritto in 7.2.2

Di seguito viene mostrato il confronto tra il modello discusso nel 6. e il modello discusso in questo paragrafo.

Nome	Mean epoch training time	Training		Validation		Test	
		Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
all-genders_14_0.25_3CL_0.2_24	10.995	0.050	0.981	0.042	0.985	0.040	0.987
all_14_0.25_3CL_0.2_24	10.772	0.057	0.978	0.044	0.984	0.041	0.985
	+ 0.223	-0.007	+ 0.003	-0.002	+ 0.001	-0.001	+ 0.002

Tabella 15 Confronto tra i modelli di 5. e 7.2.2

Come si può notare dalla Tabella 15, i risultati ottenuti non variano di quantità rilevanti.

7.3 Feature selection

La feature selection è un importante concetto del machine learning, che può influire in maniera rilevante sulle performance di un modello. Infatti, features irrilevanti possono penalizzare le performance dello stesso, per cui si rende necessaria una attenta fase di scelta delle feature da utilizzare. La feature selection può essere intesa come un processo in cui vengono selezionate (automaticamente o manualmente) le feature più rilevanti per la feature target, cioè la variabile predittiva.

In generale, una operazione di questo tipo consente di:

1. Ridurre l'Overfitting:

Meno features corrispondono a una minore probabilità di effettuare decisioni basate su noisy data.

2. Migliorare l'Accuracy

In questo progetto sono state utilizzate tre tecniche tra le diverse presenti in letteratura. Esse sono:

1. Selezione Univariata
2. Feature Importance
3. Matrice di Correlazione con Heatmap

Nel seguito verranno mostrate le diverse tecniche applicate sul dataset utilizzato nel progetto, con degli snippet di codice per mostrarne il funzionamento.

Operazione preliminare alle tecniche di feature selection è stata quella di separare la colonna target dalle altre. Ciò è stato effettuato tramite il codice di Figura 28.

```
data = pd.read_csv("dataset_with_label.csv")
X = data.iloc[:,1:6] #independent columns
y = data.iloc[:, -1] #target column eye_state
```

Figura 28 Separazione attributo target.

7.3.1 Selezione Univariata

I test statistici sono uno dei modi utilizzati per selezionare le features che hanno la più alta correlazione con la variabile di output.

La selezione univariata seleziona le migliori features basati su test statistici univariati. Viene confrontata ogni feature alla variabile target, per esaminare se vi è una relazione statisticamente rilevante fra loro. Analizzando la relazione tra una feature e la variabile target, le altre features vengono ignorate. Per tale motivo tale tecnica viene chiamata univariata. Da ciò ne deriva che ogni feature ha un proprio punteggio. Alla fine del processo, i punteggi delle features vengono confrontati, selezionando solo quelli più alti.

Uno dei test statistici che è possibile utilizzare è quello del chi-squared (χ^2).

Il test del χ^2 è usato in statistica per verificare l'indipendenza di due eventi. Nella feature selection, invece, viene utilizzato per testare se l'occorrenza di uno specifico termine e l'occorrenza di una specifica classe sono indipendenti.

Formalmente, dato un documento D , stimiamo la seguente quantità per ogni termine e li ordiniamo per punteggio:

$$\chi^2(D, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

Dove:

- N è la frequenza osservata ed E rappresenta la frequenza attesa
- e_t è 1 se il documento contiene il termine t , 0 altrimenti
- e_c è 1 se il documento è nella classe c , 0 altrimenti

Per ogni feature (termine), un alto punteggio nel test χ^2 indica che l'ipotesi nulla di indipendenza

H_0 : "la classe del documento non ha nessuna influenza sulla frequenza del termine"

deve essere rifiutata, e l'occorrenza del termine e della classe sono dipendenti. In questo caso la feature deve essere considerata.

La classe SelectKBest della libreria di Scikit-learn consente di effettuare un insieme di test statistici per selezionare uno specifico numero k di features. Tra questi è presente anche il test del χ^2 .

Di seguito viene mostrato il codice per restituire le 5 features del dataset insieme ai relativi punteggi, utilizzando il test χ^2 .

```
#apply SelectKBest class to extract top 5 features
bestfeatures = SelectKBest(score_func=chi2, k=5)
fit = bestfeatures.fit(X,y)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score'] #naming the dataframe columns
print(featureScores.nlargest(5,'Score')) #print 5 features
```

Figura 29 Snippet per la selezione univariata.

Il risultato di tali operazioni è mostrato nella Figura 30.

	Specs	Score
2	Reflections	2132.935200
4	Sensor_type	1449.041316
0	Glasses	1384.871264
1	Genders	758.903912
3	Lighting_conditions/Image_quality	262.925175

Figura 30 Risultati selezione univariata.

Analizzando la Figura 30, si può notare che le features che sembrano essere più correlate all'attributo target risultano essere nell'ordine "Reflections", "Sensor_type" e "Glasses".

Si è creato un nuovo modello che considerasse tali features, oltre alle immagini. La struttura del modello, a meno delle features utilizzate per il training, è la medesima del modello descritto nella sezione 5.

In Figura 31 sono mostrati i grafici riportanti l'andamento della loss e dell'accuracy al crescere del numero di epoch:

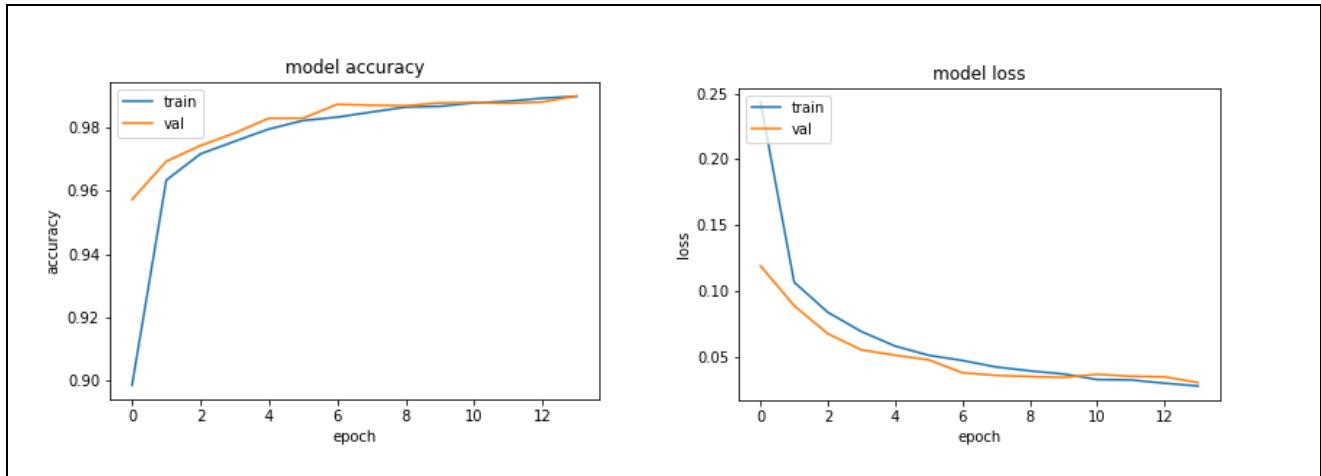


Figura 31 Andamento di accuracy e loss del modello descritto in 7.3.1.

I risultati del modello sono mostrati nella Tabella 16:

Nome	Training		Validation		Test	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
img,reflections,sensorType ,glasses_14_0.25_3CL_0. 2_24	0.064	0.976	0.050	0.982	0.035	0.988

Tabella 16 Risultati modello descritto in 7.3.1.

Di seguito viene mostrato il confronto tra il modello discusso nella sezione 5. e il modello discusso in questo paragrafo.

Nome	Training		Validation		Test	
	Loss	Accura cy	Loss	Accura cy	Loss	Accura cy
img,reflections,sensorType ,glasses_14_0.25_3CL_0. 2_24	0.064	0.976	0.05	0.982	0.035	0.988
all_14_0.25_3CL_0.2_24	0.057	0.978	0.044	0.984	0.041	0.985
	+ 0.007	- 0.002	+ 0.006	- 0.002	- 0.006	+ 0.003

Tabella 17 Confronto tra i modelli di 5. e 7.3.1.

Come si può notare dalla Tabella 17, i risultati ottenuti non variano di quantità rilevanti.

7.3.2 Feature Importance

Alternativamente alla selezione univariata, si può considerare l'importanza di ciascuna feature del dataset utilizzando una apposita proprietà del modello, cioè quella della "feature importance". Questa restituisce un punteggio per ogni feature del dataset: più è alto il punteggio e più la feature è rilevante rispetto alla feature target.

Tale proprietà è una classe propria dei classificatori ad alberi. Per usarla, quindi, ci sarà la necessità di usare un'istanza di questo tipo di classificatori.

In particolare, viene utilizzato il cosiddetto Extremely Randomized Trees Classifier o più semplicemente Extra Trees Classifier, che è una tecnica di ensemble learning che aggrega i risultati di diversi decision trees non correlati raccolti in una foresta, per poi restituire il risultato di classificazione di quest'ultima. Esso differisce da un tradizionale classificatore Random Forest nel modo di costruire i decision trees nella foresta. Ogni Decision Tree nella Extra Trees Forest è costruito a partire dai dati di training originali. Successivamente, per ogni nodo, ad ogni albero viene passato in input un campione casuale delle k features dall'insieme delle feature, da cui ogni decision tree deve selezionare le migliori feature per dividere i dati basandosi su un particolare criterio (spesso si utilizza l'indice Gini). Tale campione casuale delle features porta alla creazione di diversi decision tree non correlati.

$$Gini\ Index = 1 - \sum_{i=1}^n p_i^2$$

con

p_i = probabilità che un oggetto venga classificato in una determinata classe

Per eseguire l'operazione di feature selection utilizzando la struttura della foresta descritta, durante la costruzione della foresta, per ogni feature, viene calcolata la riduzione normalizzata totale nel criterio matematico utilizzato nella decisione dello split delle feature. Considerando come criterio matematico l'indice Gini, la riduzione normalizzata totale viene chiamata Gini Importance. Per effettuare la feature selection, ogni feature è ordinata per importanza in modo discendente, a seconda della propria Gini Importance. Successivamente, si considerano le prime k feature, a seconda del k scelto dall'utente.

Nello snippet di codice seguente viene mostrato come sia possibile estrarre i punteggi per le 5 feature del dataset. Il criterio utilizzato di default dall'ExtraTreesClassifier è l'indice Gini.

```
model = ExtraTreesClassifier()
model.fit(X,y)
#use inbuilt class feature_importances_ of tree based classifiers
print(model.feature_importances_)
```

Figura 32 Costruzione del classificatore ad albero.

Stampando il contenuto della variabile `model.feature_importances_` si ottiene una lista di valori indicante l'importanza di ognuna delle feature del dataset.

```
[0.17010451 0.07457591 0.19342344 0.02566359 0.53623254]
```

Figura 33 Risultati della feature importance.

L'ordine dei risultati restituiti si riferisce a quello del salvataggio degli attributi nel dataframe. Per permettere una visualizzazione più chiara è possibile creare un grafico che mostri, oltre al valore delle feature, anche la loro denominazione.

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```

Figura 34 Snippet per la costruzione dell'istogramma della feature importance.

Eseguendo il codice si ottiene il seguente grafico:

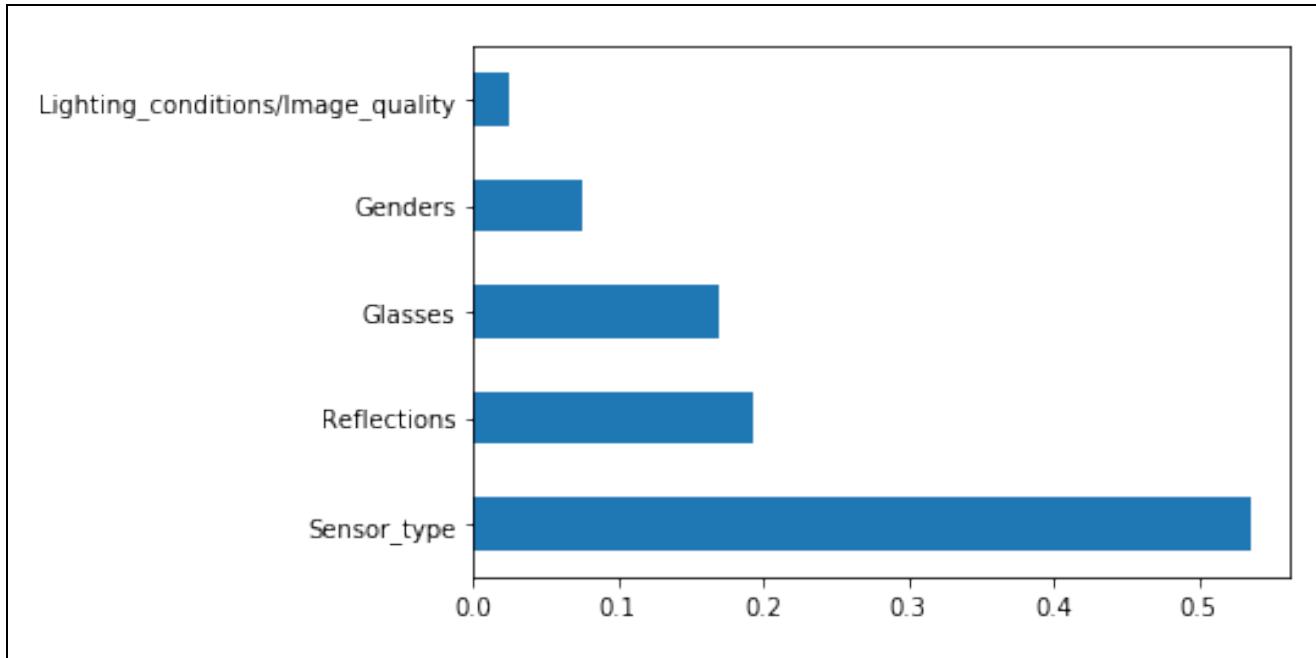


Figura 35 Istogramma della feature importance.

Dal grafico in Figura 35, è possibile osservare che la feature che ha riportato il maggiore punteggio è stata "Sensor_type", differentemente a quanto accaduto con il metodo della selezione univariata descritto nel 7.3.1. Le altre features del dataset presentano dei punteggi di importanza molto minori di quello ottenuto da "Sensor_type", addirittura inferiori alla metà di quest'ultimo.

Si è quindi creato un nuovo modello che considerasse la feature "Sensor_type" oltre alle immagini. La struttura del modello, a meno delle features utilizzate per il training, è la medesima del modello descritto nella sezione 5.

In Figura 36 sono mostrati i grafici riportanti l'andamento della loss e dell'accuracy al crescere del numero di epoch:

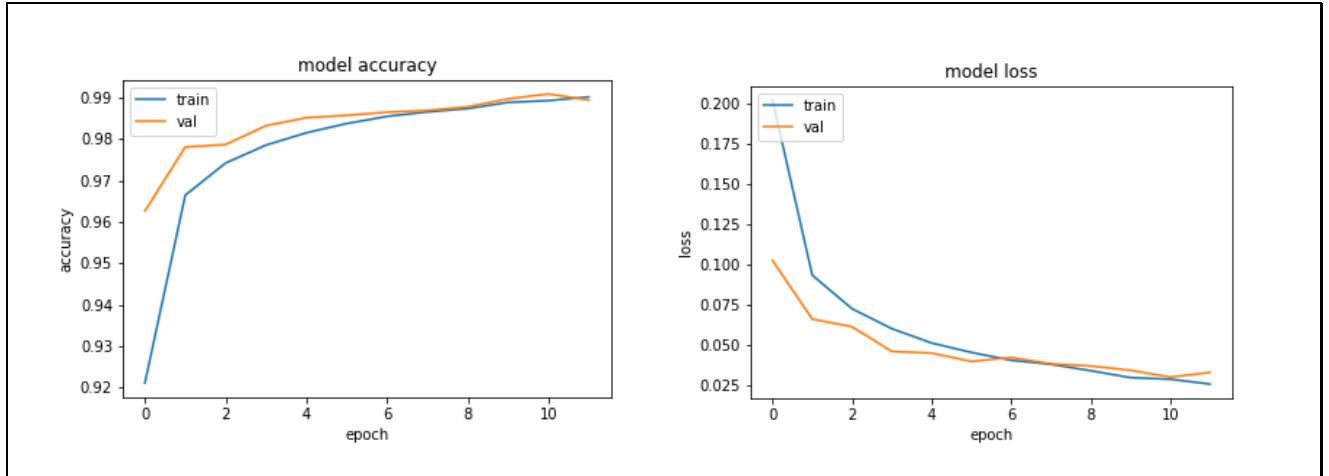


Figura 36 Andamento di accuracy e loss del modello descritto in 7.3.2.

I risultati del modello sono mostrati nella Tabella 18:

Nome	Training		Validation		Test	
	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
img,sensorType_12_0.25_3 CL_0.2_24	0.059	0.977	0.047	0.983	0.039	0.987

Tabella 18 Risultati modello descritto in 7.3.2.

Di seguito viene mostrato il confronto tra il modello discusso nella sezione 5. e il modello discusso in questo paragrafo.

Nome	Training		Validation		Test	
	Loss	Accura cy	Loss	Accura cy	Loss	Accura cy
img,sensorType_12_0.25 _3CL_0.2_24	0.059	0.977	0.047	0.983	0.039	0.987
all_14_0.25_3CL_0.2_24	0.057	0.978	0.044	0.984	0.041	0.985
	+ 0.002	- 0.001	+ 0.003	- 0.001	-0.002	+ 0.002

Tabella 19 Confronto tra i modelli di 5. e 7.3.2.

Come si può notare dalla Tabella 19, i risultati ottenuti non variano di quantità rilevanti.

7.3.3 Matrice di Correlazione con Heatmap

Un ulteriore metodo per la feature selection è rappresentato dall'analisi della matrice di correlazione delle features. Essa mostra come le features sono correlate rispetto alle altre o all'attributo target. La correlazione può essere positiva o negativa. Nel primo caso l'incremento del valore della feature provoca l'incremento di quello dell'attributo target, mentre nel secondo l'incremento del valore della feature provoca il decremento del valore dell'attributo target. Il modo più comune di calcolare la correlazione è tramite l'utilizzo del coefficiente di Pearson. Esso misura soltanto le relazioni lineari tra due variabili, non riuscendo a rilevare relazioni non-lineari. Il valore del coefficiente di Pearson varia da -1 a +1, dove:

- + 1 descrive una correlazione lineare positiva perfetta
- - 1 descrive una correlazione lineare negativa perfetta
- 0 indica che non vi è nessuna correlazione

Una matrice di correlazione, quindi, non è altro che l'insieme dei valori di correlazione tra tutte le coppie di attributi. Essa è simmetrica e presenta tutti gli elementi diagonali pari ad 1, essendo 1 il valore della correlazione di un elemento con sé stesso.

Per una migliore lettura della stessa, la matrice di correlazione può essere associata ad una heatmap. Questa è una tecnica di data visualization che mostra la magnitudine di un fenomeno attraverso un colore.

Tramite la heatmap è facile verificare quali features sono più correlate alla variabile target. Il seguente codice mostra come sia possibile creare una matrice di correlazione associata ad una heatmap, ottenuta tramite la libreria Seaborn.

```
#get correlations of each features in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(15,15))
#plot heat map
g=sns.heatmap(data[top_corr_features].corr(), annot=True, cmap="RdYlGn")
```

Figura 37 Snippet per la costruzione della matrice di correlazione con heatmap.

Eseguendo le operazioni, si ottiene il risultato in Figura 38.

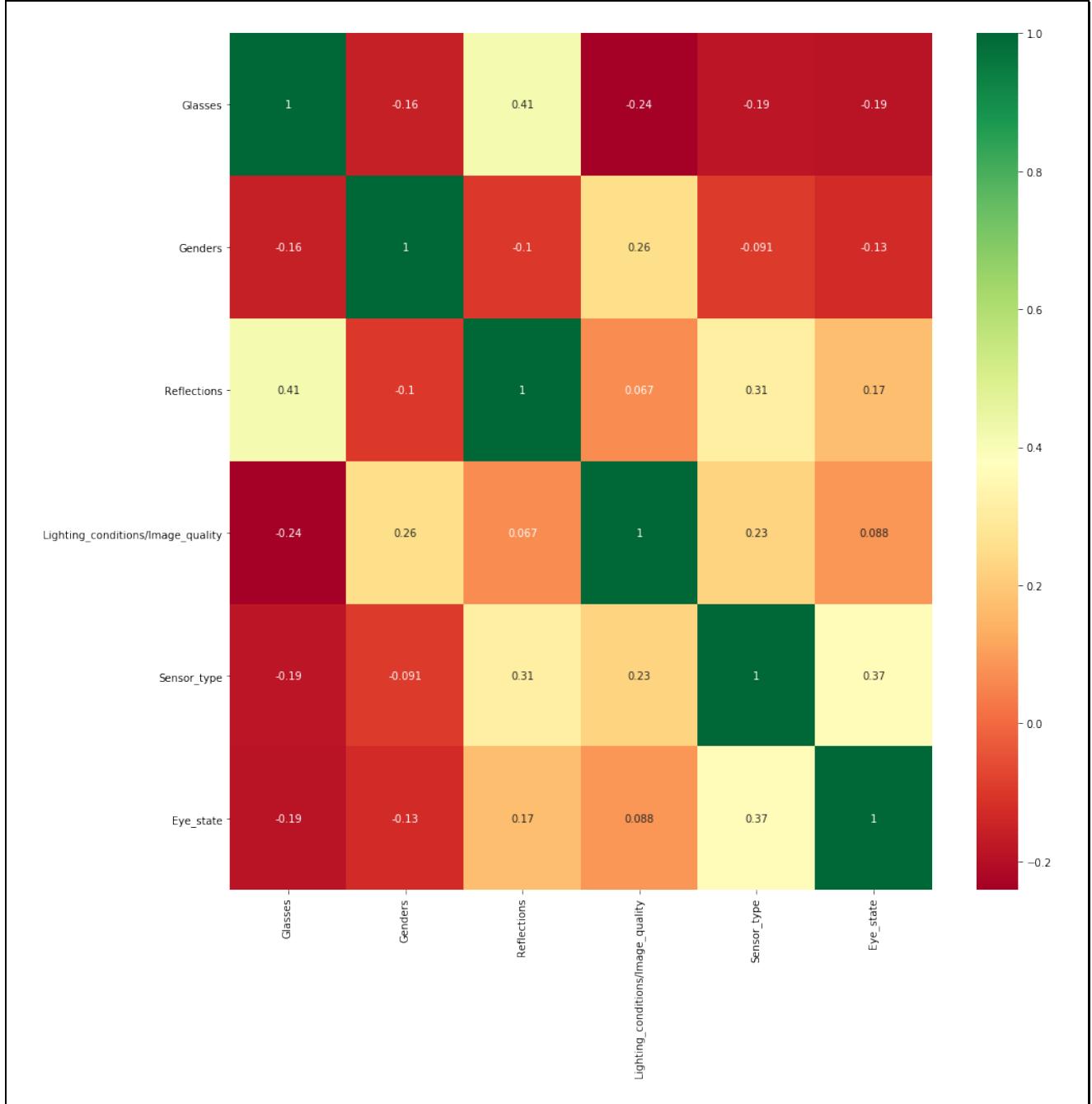


Figura 38 Matrice di correlazione con heatmap.

Analizzando l'immagine in Figura 38, si nota che nessuna feature è fortemente correlata con la variabile target "Eye_state": la correlazione più alta è risultata essere 0.37, corrispondente alla feature "Sensor_type".

Nello specifico caso in questione, l'applicazione dei risultati ottenuti da tale tecnica di feature selection non migliora le performance del modello. Infatti, come si è visto dall'analisi della tecnica di Feature Importance nella sezione 7.3.2, il modello che considera solo immagini e la feature Sensor_type non ottiene risultati molto diversi dal modello utilizzato nella sezione 5.

8. Confronto finale

In questa sezione verrà mostrato un confronto tra i lavori già svolti e la tecnica proposta e successivamente verrà mostrato un sommario delle diverse sperimentazioni effettuate che sono state descritte nelle sezioni precedenti.

8.1 Confronto con lavori già svolti

Autori	Tecnica	Accuracy	Testing Data	Intrusivo	Scarsa illuminazione	Tipo camera	Funzion a con occhiali da vista/ sole.	Funzion a con faccia inclinata	Rea l tim e	Error rate
Mandee p et al 2012. [3]	Eye Blink Monitoring using Mean Sift Algorithm	99.4%	Non specificato	No	No	Webcam	Non specificat o	No	Si	1%
Salvatore, et al 2011.[2]	FPGA based prototypin g	Non specificata	Non specificato	No	Si	IR CCD Camera	Non specificat o	Si	Si	Non specificato
Flares, et al 2009.[1]	Face and Eye monitoring based on neural networks & visual informatio n	95.6%	5 video	No	Si	Monocula r Camera	Si	Si	Si	4.4%
Chuang-Wen, et al 2013.[4]	Computer Vision & Machine Learning Algorithm	83%	12 video	No	Non specificato	Dual camera smartpho ne	Non specificat o	Non specificat o	Si	17%
A.Sahay adhas. et al 2013.[5]	Electroculogram and vehicle based measures	Non specificata	15 video	Si	Si	Non specificat o	Si	Si	No	Non specificato
A. Rahman , et al 2015.[6]	Real Time Drowsiness Detection using Eye Blinks Monitoring	94%	16 video	No	No	16MP webcam	No	Si	Si	6%
Tecnica proposta	Real Time Drowsiness Detection using	99%	Circa 7000 immagini	No	Si	7MP Webcam,	Solo con occhiali da vista	Si	Si	1%

	Convolutio nal Neural Network (CNN)					5MP Infrared Camera.					
--	--	--	--	--	--	----------------------------	--	--	--	--	--

Tabella 20 Confronto tecniche.

La Tabella 20 illustra il confronto tra la tecnica proposta e le metodologie precedenti. La tecnica proposta ha un'accuratezza stimata del 99% che risulta essere quasi uguale all'algoritmo proposto da Mandeep et al che presenta la massima accuratezza tra le tecniche sopra menzionate. La qualità distintiva di questo approccio è che è stato testato su un test set abbastanza grande (circa 7000 immagini), che include condizioni di illuminazione e risoluzioni della telecamera variabili. L'algoritmo proposto è abbastanza economico e può essere implementato utilizzando solo una webcam e non richiede altri dispositivi hardware per funzionare. Inoltre, non è invadente e funziona bene in real-time. Per quanto riguarda le condizioni di scarsa illuminazione si consiglia l'utilizzo di una camera infrared. Infine, se gli occhi sono coperti con occhiali da sole, l'algoritmo fallisce nel rilevamento degli occhi e pertanto, non si ottengono i risultati sperati.

8.2 Sommario delle sperimentazioni effettuate

In questo sezione è riportata una tabella riepilogativa delle diverse sperimentazioni effettuate e dei risultati ottenuti su ciascuno di esse. La seguente tabella è stata estratta dal file *Sommario sperimentazioni.xlsx*.

Model Name	Glasses	Genders	Reflections	image_quality	Sensor_type	Epochs	% Dropout	Convolutional Layers	% Validation	% Test set	Train loss	Train accuracy	Val loss	Val accuracy	Test loss	Test accuracy	
img_15_0.25_3CL_0.2_24						15	0.25 - 0.5	3	10.0	20	0.0686	0.9745	0.0565	0.9800	0.0462	0.9868	
img_14_0.25_3CL_0.2_24						14	0.25 - 0.5	3	10.0	20	0.0648	0.9760	0.0480	0.9823	0.0400	0.9863	
all_25_0.25_3CL_0.2_52	✓	✓	✓	✓	✓	25	0.25 - 0.5	3	10.0	20	0.0521	0.9802	0.0512	0.9831	0.0615	0.9866	
all_25_0.25_3CL_0.2_24	✓	✓	✓	✓	✓	25	0.25 - 0.5	3	10.0	20	0.0370	0.9862	0.0381	0.9871	0.0487	0.9883	
all_14_0.25_3CL_0.2_24	✓	✓	✓	✓	✓	14	0.25 - 0.5	3	10.0	20	0.0574	0.9787	0.0443	0.9849	0.0417	0.9858	
CEWDdataset_img_25_0.25_3CL_0.2_24						25	0.25 - 0.5	3	10.0	0.001	0.2376	0.8972	0.1954	0.9216	0.8538	0.6493	
img,sensorType_12_0.25_3CL_0.2_24						✓	12	0.25 - 0.5	3	10.0	20	0.0599	0.9777	0.0477	0.9837	0.0391	0.9873
img,reflections,sensorType,glasses_14_0.25_3CL_0.2_24	✓					✓	14	0.25 - 0.5	3	10	20	0.0641	0.9763	0.0504	0.9821	0.035	0.9884
all-glasses_14_0.25_3CL_0.2_24		✓	✓	✓	✓	14	0.25 - 0.5	3	10	20	0.051	0.98	0.0430	0.984	0.032	0.989	
all-genders_14_0.25_3CL_0.2_24	✓		✓	✓	✓	14	0.25 - 0.5	3	10	20	0.05	0.981	0.0420	0.985	0.04	0.987	

Figura 39 Sommario sperimentazioni effettuate.

Sul file Excel, per agevolare la lettura dei risultati è stata fornita la possibilità di ordinare i modelli in base ad uno dei valori di interesse, come Test loss, Test accuracy,

9. Descrizione demo

In questa sezione verrà descritta la demo realizzata per testare il modello in casi reali. La demo creata permette in real-time di determinare se gli occhi del soggetto inquadrato sono aperti o chiusi. Nel caso gli occhi del soggetto inquadrato rimangono chiusi per un tempo superiore ad un valore preimpostato, viene lanciato un allarme sonoro e visivo.

Per la realizzazione della demo si è fatto uso di OpenCV (*Open Source Computer Vision Library*), ovvero, una libreria software multipiattaforma utilizzata nell'ambito della visione artificiale in real-time.

La libreria offre un insieme di algoritmi di visione artificiale e machine learning sia classici che all'avanguardia. Tali algoritmi possono essere utilizzati per rilevare e riconoscere volti, identificare oggetti, classificare azioni umane in video, tracciare oggetti in movimento, estrarre modelli 3D di oggetti, trovare immagini simili in un database di immagini, ecc.

Per ulteriori informazioni si rimanda alla documentazione di OpenCV [12].

Nei paragrafi seguenti, tramite delle immagini verrà illustrato il funzionamento della demo, mentre per la visione del codice realizzato si rimanda al file Live_Drowsiness_Detection.py.

9.1 Detection con buone condizioni di luce

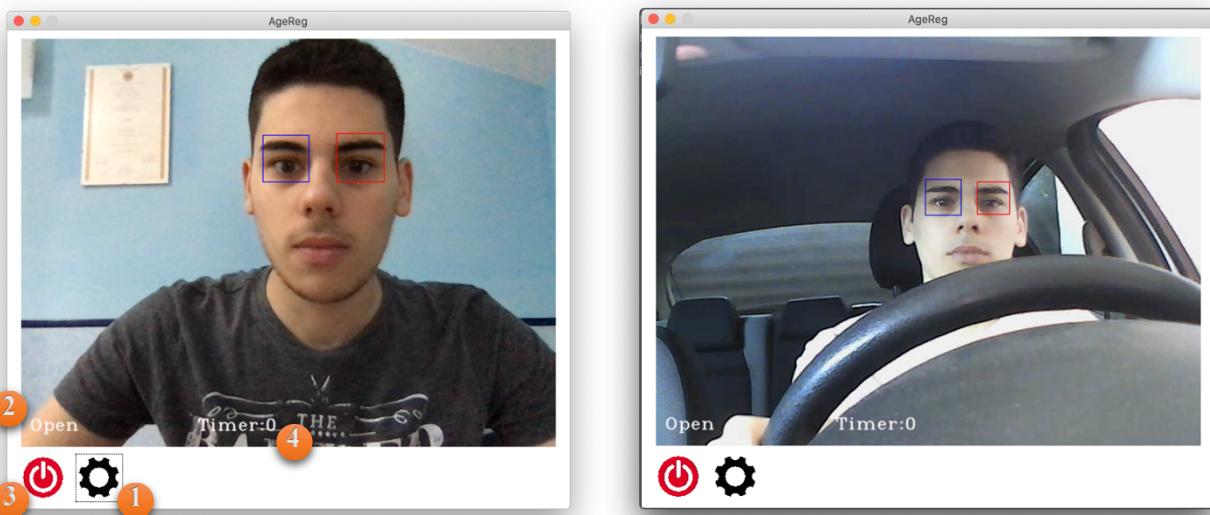


Figura 40 Drowsiness detection con buone condizioni di luce.

- 1 Tasto per accedere alle impostazioni della demo descritte nel paragrafo 9.4.
- 2 Viene mostrato lo stato degli occhi, in questo caso “Open”.
- 3 Tasto per chiudere la demo.

- 4 Viene mostrato un timer che ci permette di visualizzare per quanto tempo il soggetto ha tenuto gli occhi chiusi. Inoltre, se si tengono costantemente gli occhi chiusi per un intervallo di tempo predefinito verrà riprodotto un allarme sonoro e visivo come mostrato nel paragrafo successivo.

Nell'immagine a destra è stata utilizzata una camera con una qualità inferiore a quella utilizzata per l'acquisizione dell'immagine sulla sinistra. Tale camera è stata posta all'interno di un'automobile per simulare il comportamento del sistema in un ambiente quanto più realistico possibile. Inoltre, anche la distanza dagli occhi risulta essere maggiore rispetto alla camera utilizzata per la prima acquisizione.

Per quanto concerne il rilevamento degli occhi si sono utilizzati classificatori a cascata basati su feature di Haar che è un metodo di rilevamento di oggetti molto efficace proposto da Paul Viola e Michael Jones nel 2001. Le feature si estraggono applicando all'immagine una serie di maschere binarie. Applicare una maschera in una determinate posizione significa sommare i valori dei pixel nell'area bianca e sottrarre il valore dei pixel nell'area nera. Di seguito sono riportati degli esempi di Feature di Haar.

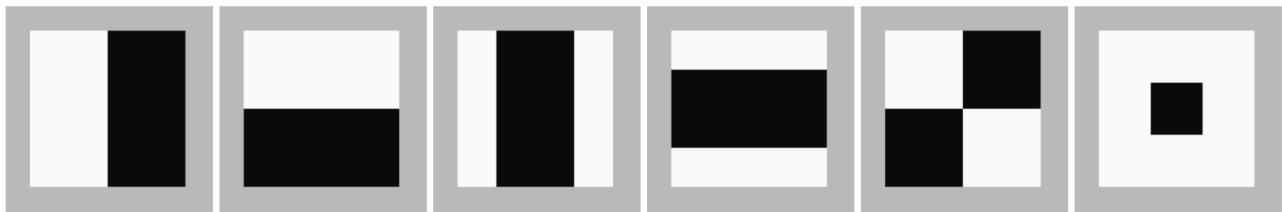


Figura 41 Esempi di Feature di Haar. Nelle aree chiare e nelle aree scure l'area sottesa viene sommata o sottratta rispettivamente.

Per ulteriori informazioni si rimanda alla documentazione di Cascade Classifier [13].

9.2 Allarme visivo/sonoro

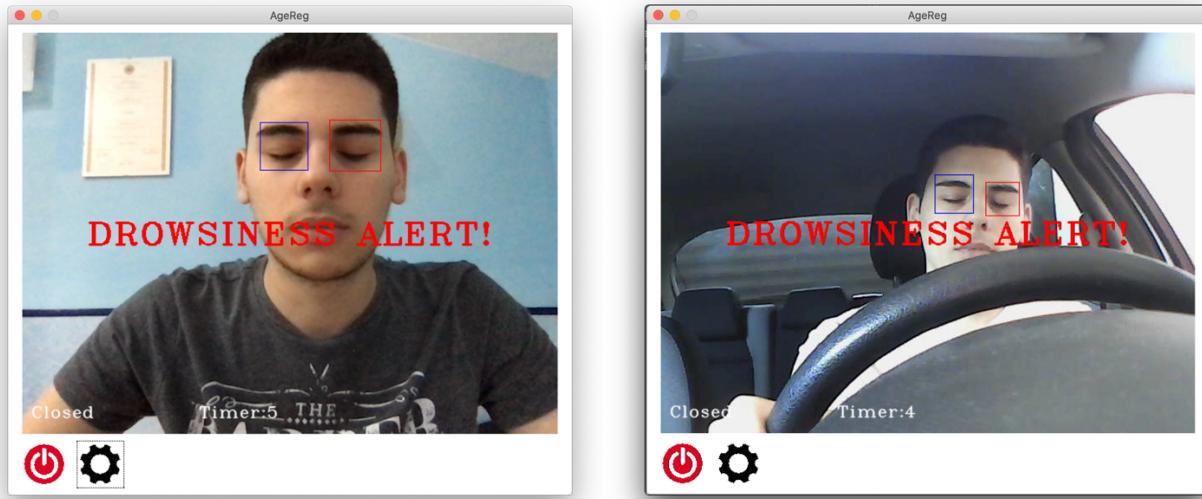


Figura 42 Allarme sonoro/visivo.

Come accennato precedentemente, dopo un intervallo temporale predefinito l’utente viene avvisato tramite un allarme sia visivo che sonoro. Tale allarme permette al conducente di svegliarsi nel caso in cui stesse dormendo ed evitare un eventuale incidente.

9.3 Acquisizione con camera IR

Di seguito sono riportate le acquisizioni effettuate con una camera Infrared per simulare la guida del veicolo in assenza di illuminazione.

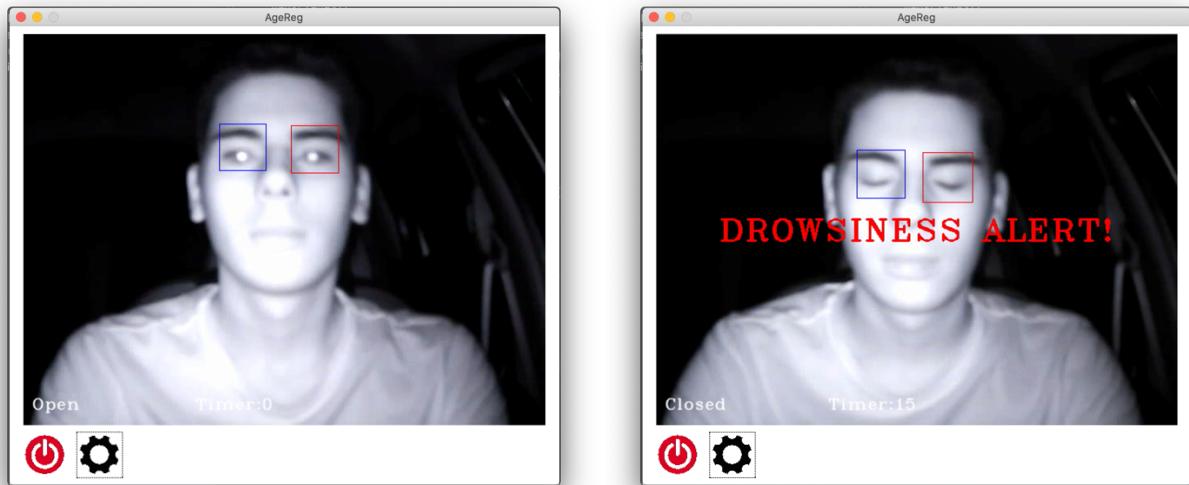


Figura 43 Acquisizioni effettuate con camera IR.

Da notare, che nonostante la bassa qualità della camera infrared a disposizione, il sistema riesce comunque a rilevare un’eventuale situazione di pericolo avvertendo l’utente con un allarme sonoro. Chiaramente, si è riusciti ad ottenere un tale risultato poiché all’interno del

dataset sono presenti anche immagini di scarsa qualità che hanno fatto sì che il modello risulti essere piuttosto robusto a variazioni di illuminazione e di qualità dell'immagine acquisita.

9.4 Schermata impostazioni

Il menu impostazioni della demo realizzata è mostrato nell'immagine seguente.



Figura 44 Impostazioni.

La schermata mostrata in Figura 44 offre le seguenti opzioni:

- 1 Possibilità di modificare la soglia del riconoscimento degli occhi. Se gli occhi non vengono riconosciuti vi è la possibilità di abbassare la soglia, mentre nel caso in cui si voglia essere più selettivi ed evitare il più possibile false alarms è possibile alzare tale soglia.
- 2 Per confermare le modifiche effettuate è necessario cliccare la spunta verde.

10. Conclusioni

In questo progetto di ricerca è stato presentato un nuovo approccio per sviluppare un sistema di drowsiness detection del conducente non intrusivo, basato sulle reti neurali artificiali. Tale sistema utilizza tecnologie avanzate per analizzare e monitorare lo stato degli occhi del conducente in tempo reale e in condizioni di guida reali. Sulla base dei risultati presentati nelle sezioni precedenti, è possibile affermare che nella soluzione proposta nel documento il rilevamento degli occhi e il loro tracciamento risultano essere piuttosto robusti e accurati in condizioni di luce variabile, variazioni di illuminazione esterna e orientamenti del viso.

Inoltre, come proposto in altre lavori riguardanti l'argomento di ricerca, è possibile aggiungere alcune funzionalità di intelligenza artificiale per migliorare ulteriormente l'affidabilità del sistema. In particolare, il sistema dovrebbe essere in grado di apprendere i particolari schemi di blinking⁵ del conducente, espressioni del viso, schemi di guida (accelerazione, velocità di decelerazione, velocità di sterzata) e i segni fisiologici del conducente come la pressione sanguigna, la frequenza cardiaca e l'attività muscolare quando sta per addormentarsi. In questo modo il sistema potrebbe predire la sonnolenza e avvisare il conducente prima che si addormenti, salvandolo così da un incidente. Infine, un altro aspetto che è possibile migliorare potrebbe riguardare la demo real-time sviluppata e descritta nella sezione 9. del documento. In particolare, si ritiene utile l'aggiunta di uno slider che permetta di selezionare l'intervallo di tempo in secondi dopo la quale far partire il segnale acustico di avviso.

⁵ Il blinking è una chiusura rapida semi-autonoma della palpebra. La durata varia mediamente tra i 100-150 millisecondi secondo l'University College London (UCL) [14].

11. Riferimenti

- [1] Global status report on road safety 2018. Geneva: World Health Organization; 2018.
- [2] M. J. Flores, J. M. Armingol and A. Escalera," Real-time warning system for driver drowsiness detection Using Visual Information", Journal of Intelligent & Robotic Systems, Volume 59, Issue 2, pp 103-125, August 2010.
- [3] S. Vitabile, A. D. Paola and F. Sorbello, "A real-time non- intrusive FPGA-based drowsiness detection system", Journal of Ambient Intelligence and Humanized Computing, Volume 2, Issue 4, pp 251-262, December 2011.
- [4] M. Singh, G. Kaur," Drowsiness detection on eye blink Duration using algorithm", International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 4, April 2012.
- [5] C. W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de-Oca, Y. Cheng, M. Lin, L. Torresani and A. T. Campbell, "CarSafe App: Alerting Drowsy and Distracted Drivers using Dual Cameras on Smart phones", In MobiSys'13, June 25-28, 2013.
- [6] Sahayadhas, K. Sundaraj and M. Murugappan," Drowsiness detection during different times of day using multiple feature", Australasian Physical & Engineering Sciences in Medicine, Volume 36, Issue 2, pp 243-250, June 2013.
- [7] A. Rahman, M. Sirshar and A. Khan, "Real time drowsiness detection using eye blink monitoring," 2015 National Software Engineering Conference (NSEC), Rawalpindi, 2015, pp. 1-7, doi: 10.1109/NSEC.2015.7396336.
- [8] Fusek R. (2018) Pupil Localization Using Geodesic Distance. In: Bebis G. et al. (eds) Advances in Visual Computing. ISVC 2018. Lecture Notes in Computer Science, vol 11241.
- [9] <https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/>
- [10] Caruccio, Loredana & Deufemia, Vincenzo & Polese, Giuseppe. (2016). Relaxed Functional Dependencies - A Survey of Approaches. IEEE Transactions on Knowledge and Data Engineering. 28. 147-165. 10.1109/TKDE.2015.2472010.
- [11] L. Caruccio, V. Deufemia, F. Naumann and G. Polese, "Discovering Relaxed Functional Dependencies based on Multi-attribute Dominance," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2020.2967722.

[12] <https://opencv.org>

[13] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[14] <https://www.ucl.ac.uk/news/2005/jul/blink-and-you-miss-it>