**Università degli Studi di Salerno**

**Dipartimento di Informatica**

Immagine che contiene regina

Descrizione generata automaticamente

**Corso di Laurea in Informatica**

Fondamenti di Data Science e Machine Learning

Progetto: Driver Drowsiness Detection

|  |  |
| --- | --- |
| Relatori | Studenti |
| Prof. Giuseppe Polese  Dott.re Stefano Cirillo | Domenico Trotta 0522500810  Michele Castellaneta 0522500824 |

**Anno Accademico 2019/2020**

**Sommario**

[1. Abstract 3](#_Toc45811727)

[2. Introduzione 4](#_Toc45811728)

[3. Lavori presenti in letteratura 5](#_Toc45811729)

[4. Dataset 7](#_Toc45811730)

[4.1 Annotazioni e statistiche del dataset 9](#_Toc45811731)

[5. Modello preliminare 11](#_Toc45811732)

(Modello proposto al seguente link:

https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/)

[5.1 Caricamento dataset e Pre-processing 11](#_Toc45811733)

[5.2 Struttura 13](#_Toc45811734)

[5.3 Training modello 15](#_Toc45811735)

[5.4 Testing 16](#_Toc45811736)

[6. Modello proposto 18](#_Toc45811737)

(Modello sviluppato da noi con modifiche)

[6.1 Struttura 18](#_Toc45811738)

[6.2 Training 18](#_Toc45811739)

[6.3 Testing 18](#_Toc45811740)

[7. Confronto tra i due modelli 19](#_Toc45811741)

[8. Data Profiling 25](#_Toc45811742)

(Descrizione dei modelli sviluppati con i risultati ottenuti dalle seguenti fasi)

[8.1 FD 25](#_Toc45811743)

[8.2 RFD 25](#_Toc45811744)

[8.2.1 Extent 25](#_Toc45811745)

[8.2.2 Confronto 25](#_Toc45811746)

[8.3 Feature selection 25](#_Toc45811747)

[9. Confronto finale 26](#_Toc45811748)

[9.1 Confronto con lavori già svolti 26](#_Toc45811749)

[9.2 Summary dei modelli sviluppati. 26](#_Toc45811750)

[10. Descrizione demo 27](#_Toc45811751)

[10.1 Schermata principale 27](#_Toc45811752)

[10.2 Allarme visivo/sonoro 28](#_Toc45811753)

[10.3 Schermata impostazioni 29](#_Toc45811754)

[11. Conclusioni 30](#_Toc45811755)

[12. Riferimenti 31](#_Toc45811756)

# 1. Abstract

Struttura del paragrafo lavoro svolto:

# 2. 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*.

Lo scopo di tale progetto prevedeva la costruzione di un modello che fosse in grado di effettuare un’operazione di Drowsiness Detection System a partire da un dataset disponibile liberamente in rete. Una volta ottenuto tale modello questo è stato utilizzato per realizzare una demo che effettuasse la detection del soggetto inquadrato.

Prima di discutere del lavoro svolto e del funzionamento della demo verrà presentata una breve descrizione e analisi del dataset utilizzato e di alcuni dei lavori già svolti in letteratura in merito all’operazione di Drive Drowsiness Detection System.

# 3. 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 [1], 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 di 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 [2], 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 driver e consentire al sistema di supportare oggetti che riflettono gli infrarossi.

Singh e Kaur [3] 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 in 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 [4], 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 [5], nel 2013 hanno usato l’EOG[[1]](#footnote-1) 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.

A. Rahman et al [6], 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). Ciò indica che ha funzionato bene in condizioni ideali.

# 4. 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 grande dataset 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 [7] che 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 molte migliaia di regioni oculari da diverse immagini NIR; le immagini di input di esempio sono presentate in Figura 1.

Immagine che contiene persona, fotografia, uomo, facciata

Descrizione generata automaticamente

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 orientati[[2]](#footnote-2) 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 e 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.

Immagine che contiene persona, automobile, uomo, guardando

Descrizione generata automaticamente

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 tre diversi livelli di riflessione basati sulla dimensione delle area di riflesso in ciascuna immagine; nessun riflesso (la prima fila in Fig. 3), piccolo riflesso (Figura 3) e grande riflesso (la terza fila in Fig. 3).

Immagine che contiene fotografia, guardando, vista, acqua

Descrizione generata automaticamente

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

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

Immagine che contiene guardando

Descrizione generata automaticamente

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.

## 4.1 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 se l'immagine dell'occhio contiene gli occhiali sono fornite per ogni immagine (con e senza gli occhiali); 24.001 immagini con 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 - nessuna, 1 - piccola, 2 - grande]**: sono stati annotati tre stati di riflessione in base alla dimensione del riflesso (66.227 nessun riflesso, 5.962 piccolo riflesso e 12.709 grande riflesso);
* **condizioni di illuminazione [0 - pessime, 1 - buono]**: 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 x 480).

Di seguito sono riportati alcuni grafici che graficamente mostrano la distribuzione della caratteristica analizzata:

Immagine che contiene screenshot, disegnando

Descrizione generata automaticamente Immagine che contiene screenshot

Descrizione generata automaticamente

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

Per quanto concerne l’attributo eye state che rappresenterà la nostra etichetta, presenta la seguente distribuzione:

Immagine che contiene disegnando

Descrizione generata automaticamente

Figura 6 Distribuzione eye state.

Come si può osservare in 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. Tutti questi tipi di immagini in questo set di dati rendono più difficile il rilevamento di parti oculari, ma permettono di addestrare un modello che sia più robusto ad eventuali variazioni di posa e illuminazione.

# 5. Modello preliminare

Nel seguente paragrafo verrà descritto il lavoro svolto per la creazione di un primo modello (basato sulle CNN) che abbiamo realizzato basandoci su lavori già svolti [8]. L’obiettivo in questo caso è stato quello di ottenere una base sul quale effettuare delle analisi che verranno descritte nei paragrafi successivi.

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

## 5.1 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.

Immagine che contiene fotografia, guardando, sedendo, indossando

Descrizione generata automaticamenteImmagine che contiene interni, piastrellato, sedendo, fotografia

Descrizione generata automaticamente

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.

## 5.2 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 che abbiamo 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

Immagine che contiene sedendo, schermo, tavolo, cellulare

Descrizione generata automaticamente

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 nel paragrafo successivo.

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 Mean Absolute Error (MAE) come loss function. La MAE pesa tutti gli errori commessi sulla stessa scala lineare, per cui i valori outlier non avranno un peso maggiore rispetto ad altri.

## 5.3 Training modello

Per quanto riguarda il validation set si è utilizzato una percentuale del training set 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. Pertanto, abbiamo utilizzato il parametro *validation\_split* del metodo fit assegnandogli il valore 0.1.



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 dopo solo 14 epoche per evitare di andare in overfitting.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Epoche | Train loss | Train accuracy | Val loss |
| img\_14\_0.25\_3CL\_0.2\_basicModel | 14 |  |  |  |

Tabella 2 Training modelli in scala di grigi con 200 e 500 epoche.

I risultati nella *Tabella 2* indicano che entrambi i modelli hanno commesso un errore medio di circa 5 anni sulle immagini di validation.

## 5.4 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\_200\_0.15\_3CL\_0.2 | 200 | 5,518 | 0.113 |

Tabella 3 Risultati ottenuti sul test set per il modello da 200 epoche.

Come si può notare dai dati in *Tabella 3*, l’errore commesso dal modello sul test set è simile a quello commesso 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 10 Esempi di predictions.

Come si può notare dai risultati, il massimo errore ottenuto dal modello è stato di due anni, nel caso dell’immagine in basso a destra.

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.

Immagine che contiene maglietta

Descrizione generata automaticamente

Figura 11 Prediction su esempio reale

In questo caso il modello ha avuto un errore pari a circa quattro anni, che è comunque inferiore all’errore medio ottenuto sul validation e sul test set (cinque anni)

# 6. Modello proposto

## 6.1 Struttura

## 6.2 Training

## 6.3 Testing

# 7. Confronto tra i due modelli

In tale paragrafo si discuterà delle valutazioni fatte sul modello illustrato nel paragrafo precedente, quello di 200 epoche che considera solo le immagini (in scala di grigi) e nessun’altra feature. Le valutazioni fatte per esso sono state effettuate anche per tutti gli altri modelli creati. I risultati sono riportati nel paragrafo 6 (Altri modelli).

La prima operazione effettuata è stata il caricamento del modello tramite la funzione **load\_model** che restituisce il modello compilato. Tale operazione è possibile soltanto se si è provveduto al salvataggio del modello dopo averne effettuato il training. Pertanto, per ogni modello si è provveduto al suo salvataggio per poterne effettuare le successive valutazioni. Il formato utilizzato per il salvataggio del modello è il formato HDF5[[3]](#footnote-3).

In *Figura 12* è riportata la struttura del modello che stiamo considerando. Come si può notare, il modello accetta in input immagini con dimensione (80, 80, 1) e restituisce in output un solo valore, ovvero l’età predetta. Inoltre, una rappresentazione come quella in figura permette una comprensione più rapida dei vari passi (convoluzione, max pooling, …) che la rete effettua e come la dimensione degli input viene modificata passo dopo passo.

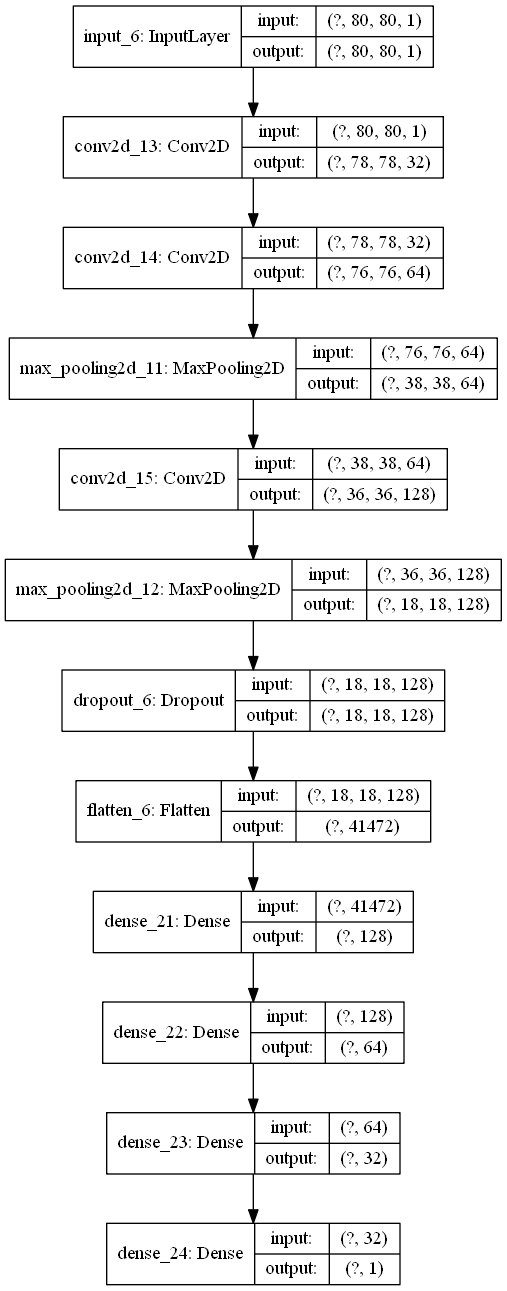


Figura 12 Struttura del modello caricato.

A questo punto si può procedere con il caricamento dell’history di training di tale modello. Questa permette di recuperare alcune informazioni riguardanti il modello come: 'accuracy', 'loss', 'val\_accuracy' e 'val\_loss'.

Anche in questo caso è fondamentale provvedere al salvataggio dell’history di un modello dopo averne effettuato il training se si desidera farne uso in un secondo momento: tale operazione è stata effettuata per tutti i modelli creati.

Di seguito sono mostrati due grafici che raffigurano l’andamento della loss e dell’accuracy durante le varie epoche.

Immagine che contiene luce, scuro, sedendo, illuminato

Descrizione generata automaticamente Immagine che contiene fumo, arrivando, treno, aria

Descrizione generata automaticamente

Figura 13 A sinistra vi è il grafico che descrive l'andamento della loss, mentre a destra abbiamo l'accuracy.

Osservando i due grafici mostrati in *Figura 13* si può notare come il modello vada in overfitting, in quanto al crescere del numero di epoche la *loss*, ma anche *l’accuracy* tendano rispettivamente a diminuire ed aumentare, mentre la *val\_loss* e *val\_accuracy*, ovvero la loss e l’accuracy registrata sul validation set tendano a stabilizzarsi. Pertanto, osservando l’andamento delle due curve mostrate nei grafici si può concludere che con un numero molto più basso di epoche avremmo ottenuto lo stesso risultato ottenuto con 200 epoche. Infatti, analizzando un modello addestrato con 500 epoche il risultato non è migliorato.

Per cercare di eliminare o quantomeno ridurre l'overfitting si sono valutati diversi approcci come:

* cercare di livellare le istanze, cioè le immagini per ogni fascia d’età;
* trasferire circa 4000 immagini dal test set al training set;
* creare modelli con diverse configurazioni di dropout e di layer convoluzionali;
* ridurre e aumentare la dimensione delle immagini in input;
* creare modelli che considerassero anche altre feature (sesso ed etnia) oltre alle sole immagini.

Per ognuno degli approcci sopra elencati si sono addestrati dei modelli ad hoc, ma nessuno di essi ha portato ad una riduzione dell’overfitting e ad un miglioramento delle performance generali sulle prediction.

Dopo aver osservato come il modello si comporta sul training set all’aumentare del numero di epoche, viene analizzato come quest’ultimo si comporta con le immagini del test set.

Immagine che contiene oggetto, spillo, molti, sedendo

Descrizione generata automaticamente

Figura 14 Età predette vs Età reali su 200 immagini di test.

Il grafico mostrato di seguito mette in relazione le prediction fatte dal modello con le età reali. Per rendere più agevole la lettura del grafico sono state considerate le prediction su soltanto 200 immagini delle oltre 4000 del test set.

Nel grafico in Figura 14 sull’asse delle y sono riportate l’età, mentre sull’asse delle x sono riportate le immagini analizzate. Attraverso tale grafico, si può notare come in molti casi l’età predetta vada a sovrapporsi perfettamente con l’età reale, mentre in molti altri risulta non coincidere affatto.

Inoltre, per avere una visione degli errori commessi su tutte le immagini è stato realizzato il grafico in *Figura 15*. Per errore si intende la differenza in valore assoluto tra l’età reale e l’età predetta.

Immagine che contiene disegnando

Descrizione generata automaticamente

Figura 15 Errore (in anni) sulle previsioni fatte su tutte le immagini di test.

Le classi di distribuzione degli errori sono state strutturate per indicare la quantità di errore commessa dal modello (espressa in anni). Ad esempio, per circa 1500 immagini di test si è registrato un errore inferiore ai 2 anni.

Sull’asse delle x della *Figura 15* sono mostrate le classi di distribuzione degli errori, mentre sull’asse delle y sono riportate le relative occorrenze.

Si può notare come in alcuni casi (25 immagini) l’errore risulta essere molto alto, addirittura superiore ai 30 anni. Per comprendere il motivo per cui su tali immagini la rete avesse commesso un errore così alto, queste sono state analizzate singolarmente, per verificare che non presentassero anomalie rispetto alle altre immagini del test set.

Di seguito è riportata l’immagine sulla quale la rete ha commesso l’errore più alto.



**Nome immagine:** 5\_1\_0\_20170104202352685.jpg.chip.jpg

**Età reale:** 5

**Età predetta:** 54.228626

**Errore (in anni):** 49.2286262512207

In questo caso, il modello ha associato a tale immagine un’età di circa 54 anni, mentre l’etichetta dell’immagine è pari a 5 anni. Chiaramente, immagini di questo tipo (cioè, immagini etichettate in modo errato dai creatori del dataset) se utilizzate nel training set possono influenzare le prestazioni del modello.

Purtroppo, analizzando manualmente le immagini con etichetta da 1 a 5 anni abbiamo riscontrato che l’immagine riportata non è un caso isolato.

Di seguito sono riportate alcune delle immagini trovate per la quale l’etichetta non coincide con l’età reale del volto. Sotto ad ogni immagine è riportata la rispettiva etichetta.

Immagine che contiene persona, abbigliamento, tuta, fotografia

Descrizione generata automaticamenteImmagine che contiene uomo, mammifero, animale, vecchio

Descrizione generata automaticamenteImmagine che contiene uomo, mammifero, persona, guardando

Descrizione generata automaticamenteImmagine che contiene persona, sorridente, donna, interni

Descrizione generata automaticamente

*1 anno*

*1 anno*

*5 anni*

*5 anni*

L’errore medio commesso dal modello su 200 immagini è di **5.66** anni, mentre calcolando

l’errore medio sull’intero test set (4739 immagini) si ha un errore pari a **5.51**.

Chiaramente, la media risulta essere influenzata da outlier come quelli analizzati sopra e

pertanto, si è deciso di calcolare l’errore mediano, poiché la mediana non risulta essere influenzata da valori estremamente alti o bassi.

L’errore mediano su tutte le immagini di test è pari a **3.64** anni. La mediana risulta essere più bassa della media, che risente degli outlier.

Infine, di seguito è riportata una tabella che riassume le valutazioni fatte per i seguenti modelli:

* **img\_200\_0.15\_3CL\_0.2**: modello analizzato precedentemente, ovvero un modello che considera soltanto le immagini (in scala di grigi).
* **RGBimg\_200\_0.15\_3CL\_0.2**: a differenza del precedente, questo modello prende in input immagini a colori.
* **RGBsex\_200\_0.15\_3CL\_0.2:** modello che considera non solo le immagini (a colori), ma anche la feature sesso.
* **RGBall\_200\_0.15\_3CL\_0.2:** modello che oltre alle immagini (a colori) considera anche tutte le altre feature (sesso ed etnia).

La tabella inoltre, contiene informazioni su:

* **Train loss:** loss media del training set;
* **Train accuracy:** accuracy media del training set;
* **Val loss:** loss media del validation set;
* **Test accuracy:** accuracy media del test set;
* **Test loss:** loss media del test set;
* **Median error:** errore mediano del test set.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nome modello | Train loss | Train accuracy | Val loss | Test accuracy | Test loss | Median error |
| img\_200\_0.15\_3CL\_0.2 | 1.807 | 0.331 | 5.613 | 0.113 | 5.518 | 3.640 |
| RGBimg\_200\_0.15\_3CL\_0.2 | 1.954 | 0.295 | 5.685 | 0.106 | 5.567 | 3.700 |
| RGBall\_200\_0.15\_3CL\_0.2 | 1.901 | 0.293 | 5.683 | 0.101 | 5.646 | 3.779 |
| RGBsex\_200\_0.15\_3CL\_0.2 | 1.896 | 0.303 | 5.675 | 0.107 | 5.520 | 3.728 |

Tabella 4 Riepilogo dei risultati.

Da notare che numero il numero di epoche, la percentuale di dropout e il numero di layer convoluzionali sono identici per tutti e quattro i modelli considerati.

Inoltre, tutti i modelli sopra elencati sono stati utilizzati nella realizzazione della demo: per

ulteriori approfondimenti si rimanda al paragrafo 7 (Descrizione demo).

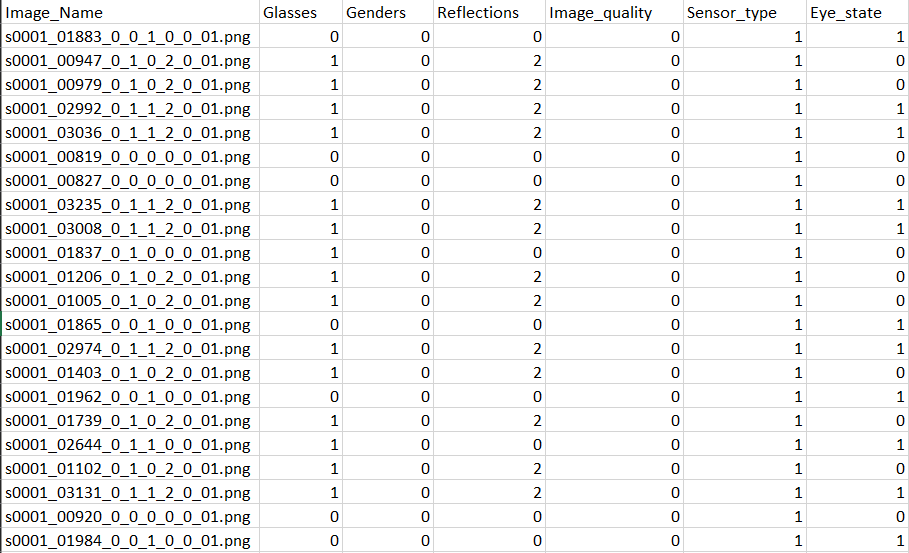
# 8. Data Profiling

Successivamente alla creazione del modello si è provveduto ad effettuare diverse operazioni di data profiling come l'estrazione di dipendenze funzionali, dipendenze funzionali rilassate(riferimento) e features selection. Operazioni 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

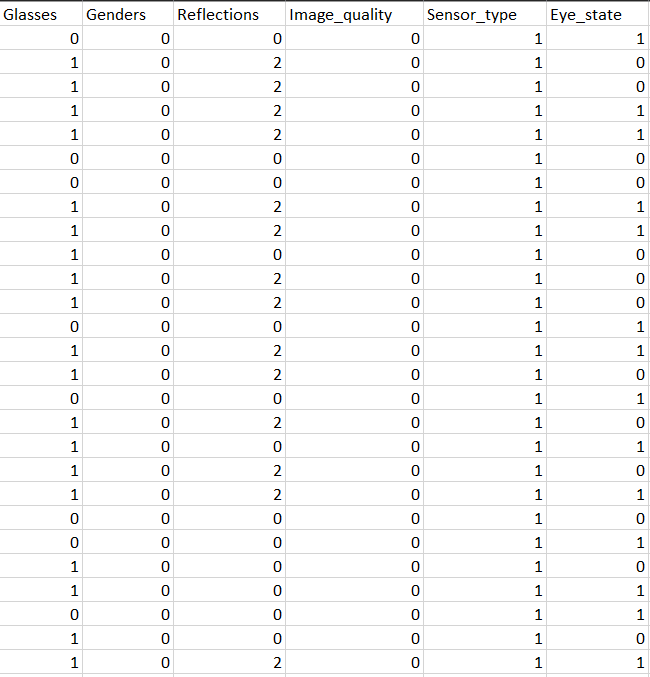
Il secondo file csv comprendeva tutte le features del dataset tranne l'attributo " Eye\_state", che rappresenta la feature target del dataset.



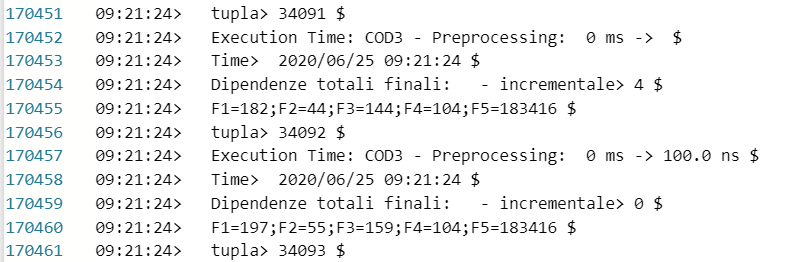


## 8.1 FD

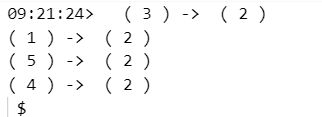
La prima operazione è stata quella di ricercare l'esistenza di Dipendenze Funzionali all'interno del dataset. Tale operazione è stata effettuata tramite il tool NOME TOOL. Il file di input utilizzato per l'algoritmo è stato quello che non prevedeva il target attribute, modificato in modo da eliminare l'attributo Image\_Name, non rilevante allo scopo.



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.



Le dipendenze in questione erano:



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 |

Per cui, le corrispondenze trovate erano:

Glasses →Genders

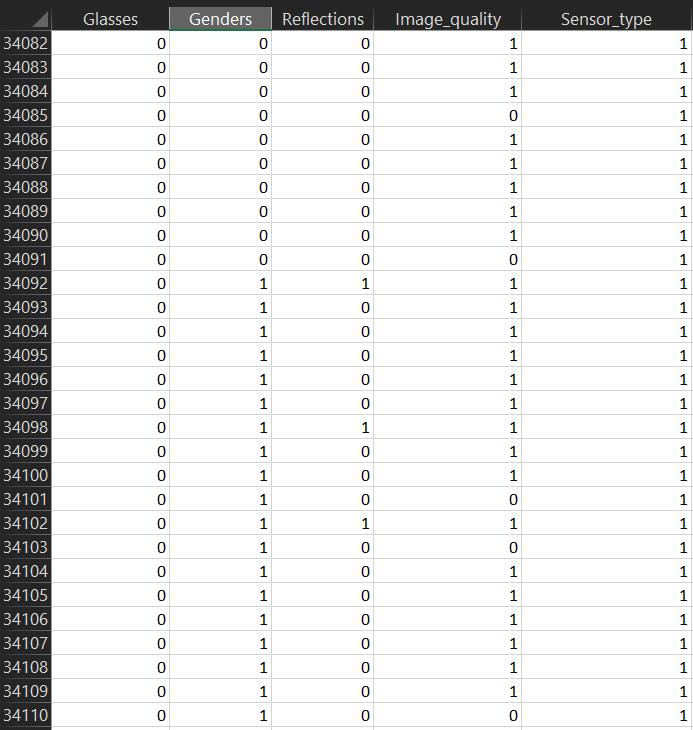
Reflections → Genders

Image\_quality → Genders

Sensor\_type → Genders

Eye\_state → Genders

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



Come si può analizzare dalla figura, 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.

## 8.2 RFD

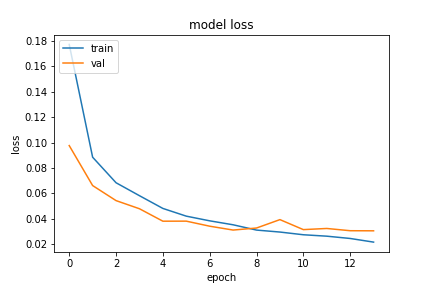
Recentemente si è posto un interesse sulle cosiddette Relaxed Functional Dependencies(METTERE RIFERIMENTO AD ARTICOLO SULLE RFD DI CARUCCIO), vale a dire delle dipendenze funzionali inesatte che rilassano su uno o più vincoli delle dipendenze funzionali canoniche. Considerando che non sono state rilevate dipendenze funzionali canoniche sul dataset, si è provveduto a ricercare RFD che rilassano sull'extent e sul confronto tramite l'algoritmo DOMINO (METTERE RIFERIMENTO).

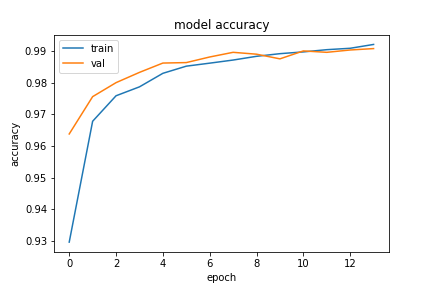
### 8.2.1 Extent

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 dipendenza funzionale approssimata (Genders Reflections Sensor\_type) ->  ( 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 affinchè una dipendenza funzionale X → Y valga. Nello specifico caso ciò significa che affinchè 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 nel 6.

In figura NUMERO sono mostrati i grafici riportanti l'andamento della loss e dell'accuracy al crescere del numero di epoche:





I risultati del modello sono mostrati nella tabella NUMERO:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Training** | | **Validation** | | **Test** | |
| Nome | **Mean epoch training time** | **Loss** | **Accuracy** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| all-glasses\_14\_0.25\_  3CL\_0.2\_  basicModel\_24 | 10.926 | 0.051 | 0.980 | 0.043 | 0.984 | 0.032 | 0.989 |

### 

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Training** | | **Validation** | | **Test** | |
| Nome | **Mean epoch training time** | **Loss** | **Accuracy** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| all-glasses\_14\_0.25\_  3CL\_0.2\_  basicModel\_24 | 10.926 | 0.051 | 0.980 | 0.043 | 0.984 | 0.032 | 0.989 |
| all\_14\_0.25\_3CL\_0.2\_basicModel\_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 |

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

### 8.2.2 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 o uguale ad uno e la differenza massima tra i valori di Y sia minore o uguale ad 1.

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

SOGLIA 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

SOGLIA 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

SOGLIA 15

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 cc ( CHE SONO?) dal tool sono le medesime:

0->0

1->0

0->1

E cioè:

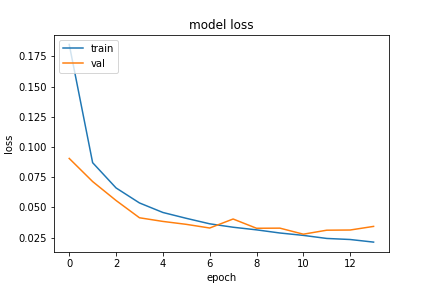
Glasses → Glasses

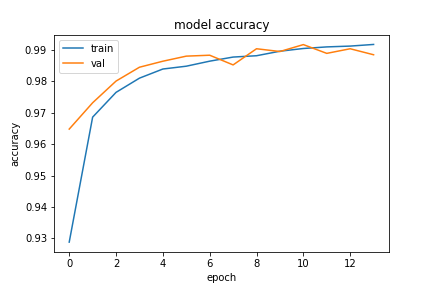
Genders → Glasses

Glasses → Genders

Poichè la prima dipendenza è banale, ed il modello senza la feature Glasses è stato descritto nell' 8.2.1, 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 NUMERO sono mostrati i grafici riportanti l'andamento della loss e dell'accuracy al crescere del numero di epoche:





I risultati del modello sono mostrati nella tabella NUMERO:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Training** | | **Validation** | | **Test** | |
| Nome | **Mean epoch training time** | **Loss** | **Accuracy** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| all-genders\_14\_0.25\_3CL\_0.2\_basicModel\_24 | 10.995 | 0.050 | 0.981 | 0.042 | 0.985 | 0.040 | 0.987 |

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

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Training** | | **Validation** | | **Test** | |
| Nome | **Mean epoch training time** | **Loss** | **Accuracy** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| all-genders\_14\_0.25\_3CL\_0.2\_basicModel\_24 | 10.995 | 0.050 | 0.981 | 0.042 | 0.985 | 0.040 | 0.987 |
| all\_14\_0.25\_3CL\_0.2\_basicModel\_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 |

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

## 8.3 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:

· Ridurre l'Overfitting: Meno features → Minore probabilità di effettuare decisioni basate su noisy data.

· Migliorare l'Accuracy

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

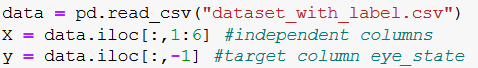
1. Selezione Univariata

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:



### 8.3.1 Selezione Univariata

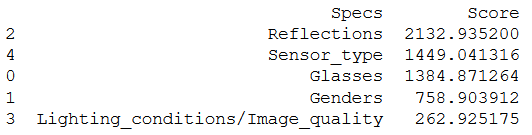
I test statistici sono uno dei modi utilizzati per selezionare le features che hanno la più alta correlazione con la variabile di output.

La classe SelectKBest della libreria di Scikit-learn consente di effettuare un insieme di test statistici per selezionare uno specifico numero di features.

Di seguito viene mostrato il codice per restituire le 5 features del dataset insieme ai relativi punteggi, utilizzando il test statistico del chi-squared (chi²).



Il risultato di tali operazioni è mostrato nella FIGURA NUMERO.



Analizzando la FIGURA NUMERO, si può notare che la feature più correlata all'attributo target risulta essere "Reflections".

### 8.3.2 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.

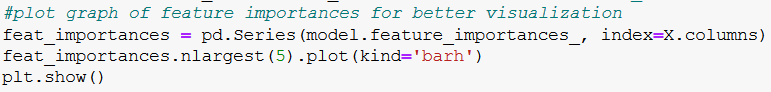
Tale proprietà è una classe propria dei classificatori ad alberi. Per usarla, quindi, ci sarà la necessità di usare un'istanza di questo tipo di classificatori. Nello snippet di codice seguente viene mostrato come sia possibile estrarre i punteggi per le 5 feature del dataset.



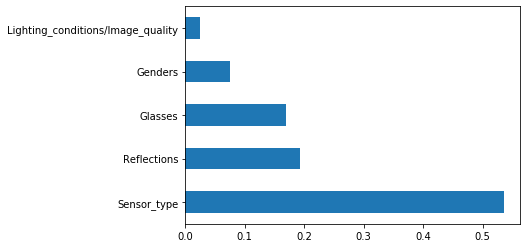
Stampando il contenuto della variabile *model.feature\_importances\_* si ottiene una lista di valori indicante l'importanza di ognuna delle feature del dataset.



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 mostra , oltre al valore delle feature, anche la loro denominazione.



Eseguendo il codice si ottiene il seguente grafico:



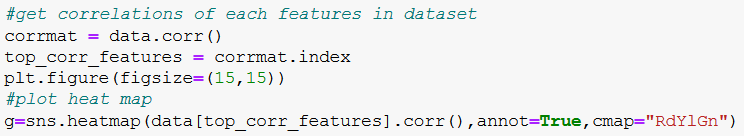
Dal grafico in FIGURA NUMERO, è possibile osservare che la feature che ha riportato il maggiore punteggio è stata "Sensor\_type", differentemente a quanto accaduto con il metodo della selezione univariata descritto nell' 8.3.1. Le altre features del dataset presentano dei punteggi di importanza molto minori di quello ottenuto da "Sensor\_type", addirittura inferiori alla metà di quest'ultimo.

### 8.3.3 Matrice di Correlazione con Heatmap

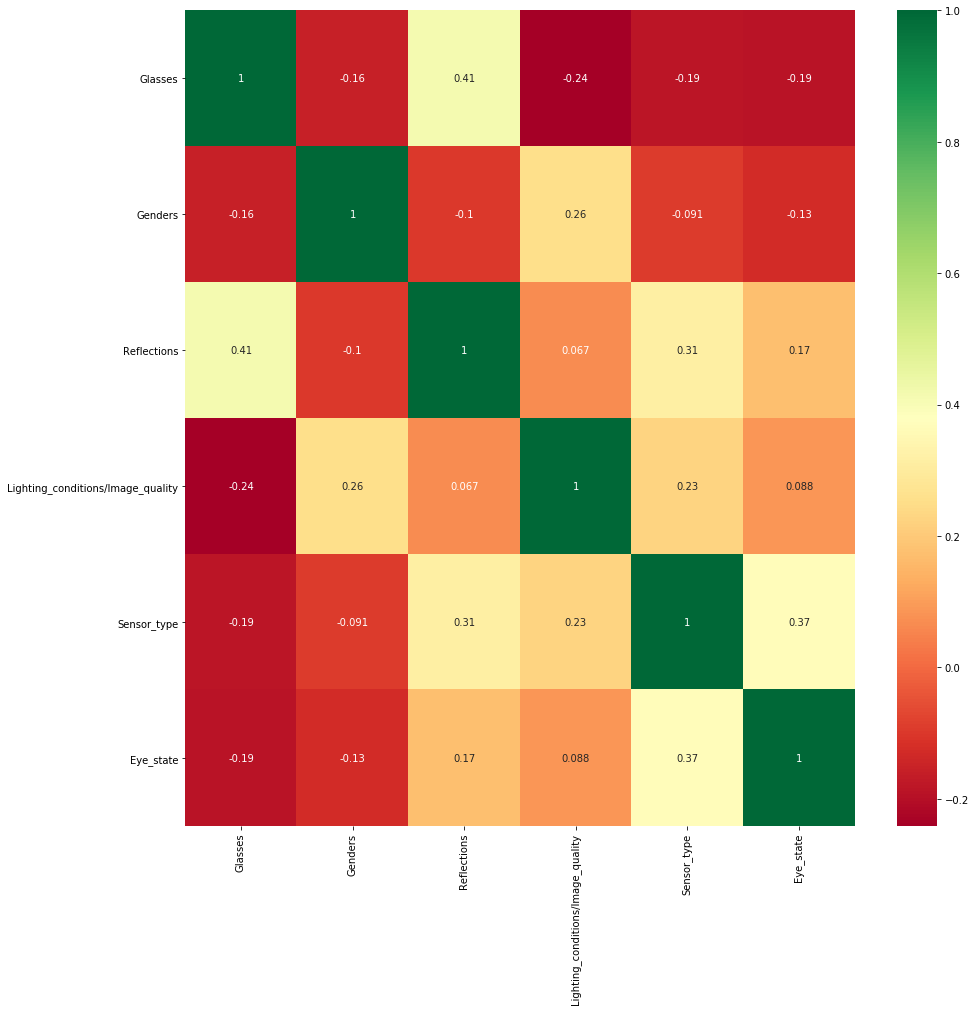
Un ulteriore metodo per la feature selection è rappresentato dalla matrice di correlazione. Essa mostra come le features sono correlate rispetto alle altre o all'attributo target. La correlazione può essere positiva o negativa. Nel primo caso l'incremento del valore della feature provoca l'incremento di quello dell'attributo target, mentre nel secondo l'incremento del valore della feature provoca il decremento del valore dell'attributo target.

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 crearla tramite la libreria Seaborn.



Eseguendo le operazioni, si ottiene il risultato in FIGURA NUMERO



Analizzando l'immagine in FIGURA NUMERO, 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".

# 9. Confronto finale

## 9.1 Confronto con lavori già svolti

## 9.2 Summary dei modelli sviluppati.

In questo paragrafo è riportata una tabella riepilogativa delle diverse configurazioni di modelli creati e dei risultati ottenuti su ciascuno di essi. La seguente tabella è stata estratta dal file *Sommario modelli creati.xlsx.*

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, ….

Immagine che contiene screenshot, largo

Descrizione generata automaticamente

Figura 16 Tabella riassuntiva dei modelli creati.

Osservando la tabella, si può notare che il miglior risultato sul test size è dato dal modello **img\_200\_0.15\_3CL\_0.2**.

# 10. 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 e nel caso fossero chiusi viene lanciato un allarme sonoro e visivo. Quindi, tramite delle immagini verrà mostrata l’interfaccia della demo, mentre per la visione del codice si rimanda al file Live\_Drowsiness\_Detection.py.

## 10.1 Age prediction con una sola persona

Immagine che contiene persona, uomo, schermo, interni

Descrizione generata automaticamente

**4**

**2**

**1**

**3**

Figura 17 Prediction con un singolo soggetto rilevato.

**1**

Tasto per accedere alle impostazioni della demo descritte nel paragrafo 7.3.

**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.

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

## 10.2 Allarme visivo/sonoro

Immagine che contiene fotografia, schermo, sedendo, uomo

Descrizione generata automaticamente

Figura 18 Allarme sonoro/visivo.

Come accennato precedentemente, dopo un intervallo temporale predefinito l’utente viene avvisato tramite un allarme sia visivo che sonoro.

## 10.3 Schermata impostazioni

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

**Immagine che contiene screenshot

Descrizione generata automaticamente**

**1**

**2**

Figura 19 Impostazioni.

La schermata mostrata in Figura 19 offre le seguente opzione:

**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.

# 11. Conclusioni

In questo documento è stato presentato l’approccio seguito per la realizzazione di un modello che fosse in grado di effettuare il prediction dell’età di un soggetto rilevato.

Partendo dai lavori già svolti, si è provveduto alla realizzazione di diversi modelli con l’obiettivo di trovare quello che fornisse le prestazioni migliori. Per i risultati ottenuti, questi sono stati, seppur non di molto, migliori di quelli dei lavori già svolti su tale dataset. Pertanto, considerando che il risultato è fortemente dipendente dal dataset e che per la creazione di quest’ultimo non è stata posta una particolare attenzione da parte dei creatori, il risultato registrato viene considerato piuttosto soddisfacente.

# 12. Riferimenti

**[1]** 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.

**[2]** 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.

**[3]** 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.

**[4]** 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.

**[5]** 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.

**[6]** 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.

**[7]** 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.

**[8]** https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/

**[9]**

**[10]** <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>

1. Segnale elettrooculogramma che viene registrato per individuare i movimenti degli occhi utili nella stadiazione del [sonno](https://it.wikipedia.org/wiki/Sonno). La registrazione si basa sulla differenza di potenziale esistente tra [cornea](https://it.wikipedia.org/wiki/Cornea) (positiva) e [retina](https://it.wikipedia.org/wiki/Retina) (negativa) e viene effettuata mediante [elettrod](https://it.wikipedia.org/wiki/Elettrodi)i. [↑](#footnote-ref-1)
2. è un descrittore di caratteristiche usate in computer vision ed in elaborazione delle immagini per il riconoscimento di oggetti. [↑](#footnote-ref-2)
3. Hierarchical Data Format versione 5 (HDF5), è un formato di file open source che supporta dati di grandi dimensioni, complessi ed eterogenei. [↑](#footnote-ref-3)