

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

**AffectiveDrive: sistema di Driver Assistance
basato sull'analisi di sensori inerziali
e tecniche di computer vision**

Relatore:

Chiar.mo Prof.

Marco Di Felice

Presentata da:

Luca D'Ambrosio

I Sessione

Anno Accademico 2017/2018

*"I "se" sono il marchio dei falliti!
Nella vita si diventa grandi "nonostante"."
Massimo Gramellini.*

Introduzione

I guidatori distratti sono la principale causa di uno sconcertante numero di incidenti stradali. I guidatori spesso sono condizionati a modificare il loro stile di guida a causa di stress, di distrazioni, sonnolenza, abuso di droghe o di alcool. In [18] vengono riportati i dati del NHTSA (National Highway Traffic Safety Administration) dove si nota che circa il 25% degli incidenti stradali siano stati causati da guidatori disattenti, in particolare, nel 2010 in America 3.092 persone sono state vittime di incidenti. In Europa secondo i dati OCSE (Organizzazione per la Cooperazione e lo Sviluppo Economico) del 2016, il 93% degli incidenti stradali deriva da comportamenti scorretti del guidatore. Precisamente in Italia, l'Istat (Istituto Nazionale della statistica) riporta che nel 2016 si sono verificati 175.791 incidenti stradali e afferma che tra le cause più frequenti vi sono la guida distratta, il mancato rispetto della precedenza e la velocità troppo elevata.

Negli ultimi anni, il rilevamento delle situazioni pericolose mentre si guida un veicolo per la prevenzione degli incidenti è un campo di ricerca molto popolare, in quanto questi studi cercano di incrementare la sicurezza stradale e del conducente. Per promuovere la sicurezza stradale e del conducente, i ricercatori hanno provato che il guidatore si comporta in maniera relativamente più sicura quando viene monitorato e può ricevere segnalazioni circa i comportamenti potenzialmente aggressivi e poco sicuri. Per il riconoscimento dello stile di guida sono state fatte ricerche con lo scopo di classificare e

riconoscere il tipo di guida e situazioni pericolose che vanno segnalate al conducente in modo da condurlo verso una guida sicura.

Tali soluzioni, conosciute con la sigla ADAS (Advanced driver-assistance systems), vengono integrate direttamente nelle auto, oppure possono essere implementate all'interno di applicazioni mobile. Quest'ultima soluzione, sviluppata e descritta in questa tesi di laurea, presenta molti vantaggi, tra cui quello di essere economica ed accessibile a tutti. Le applicazioni mobile immerse nel campo delle ADAS basano la loro attività su un'analisi dei dati sensoriali estratti durante la guida. La stessa analisi viene svolta attraverso delle tecniche di machine learning che, sulla base delle informazioni ottenute durante un processo di feature extraction, permettono di riconoscere la classe di appartenenza dei dati e di costruire schemi da utilizzare per classificazioni future.

In questa tesi l'applicazione sviluppata segue il pattern appena descritto, ed ha come obiettivo principale quello di riconoscere, attraverso l'analisi dei dati sensoriali, delle manovre di guida non sicure effettuate da un conducente. Inoltre, una volta individuate situazioni pericolose, l'applicazione utilizza il software development kit AffDex per migliorare il rilevamento delle prestazioni di guida. Quest'ultimo, offerto dall'azienda americana Affectiva, riconosce le emozioni e le espressioni prodotte dal volto del conducente, andando a capire se sia distratto oppure in uno stato di sonnolenza. Se uno di questi stati viene riscontrato, il guidatore verrà avvertito tramite un'allarme sonoro.

In particolare, nell'applicazione prodotta vengono estratti i dati sensoriali, rispettivamente di accelerometro, giroscopio e magnetometro. Vengono applicate delle tecniche di feature extraction per rappresentare i dati sensoriali; viene inoltre costruito un modello di predizione partendo da un dataset¹ di dati composto da rilevamenti di guida e infine

¹Il modello di predizione è stato costruito sulla base di un set di dati scaricato dalla rete, contenente una raccolta di misurazioni di dati sensoriali atte ad individuare diverse manovre svolte alla guida.

viene applicato l'algoritmo di classificazione RandomForest alle feature estratte per predire il comportamento del conducente. La parte di competenza di AffDex viene gestita attraverso delle soglie di valori, a causa della scarsa documentazione presente sull'argomento.

La tesi è composta da 5 capitoli ed è strutturata come segue: nel **Capitolo 1** viene fatta una panoramica generale sul mondo dei sistemi ADAS, descrivendo cosa sono e le tecniche utilizzate per la loro realizzazione. Inoltre, viene presentata Affectiva partendo dall'analisi degli studi sul riconoscimento delle emozioni/espressioni facciali fino ad arrivare alla descrizione nel dettaglio del software AffDex. Nel **Capitolo 2** si descrive la progettazione dell'applicazione, mentre nel **Capitolo 3** verranno descritti i dettagli implementativi. Nel **Capitolo 4** vengono mostrati i risultati ottenuti dall'applicazione dei diversi algoritmi di classificazione sul dataset. Infine, nel **Capitolo 5** saranno espone le conclusioni e i possibili sviluppi futuri dell'applicazione realizzata.

Indice

Introduzione	i
1 Stato dell'arte	1
1.1 ADAS	1
1.1.1 Sistemi ADAS: cosa sono?	2
1.1.2 Struttura generale dei sistemi ADAS su uno smarphone	5
1.1.3 Driving data: estrazione dei dati sensoriali	6
1.1.4 Sensori smartphone	6
1.1.5 Feature extraction	9
1.1.6 Algoritmi di classificazione	10
1.1.7 Fini e scopi delle app	11
1.1.8 Applicazioni ADAS sul mercato	13
1.2 Analisi e riconoscimento espressioni-emozioni facciali	16
1.2.1 Affective computing	17
1.3 Analisi delle espressioni: tra informatica e psicologia	18
1.3.1 Affectiva	27
2 Progettazione di un'applicazione ADAS	33
2.1 Architettura e scelte progettuali	34

2.1.1	Estrazione dei dati	35
2.1.2	Feature extraction	36
2.1.3	Creazione del modello di classificazione	37
2.1.4	Classificazione e valutazione delle situazioni pericolose	41
2.1.5	Modulo Affectiva	42
2.2	Funzionalità dell'applicazione	43
3	Implementazione	47
3.1	Strumenti	47
3.1.1	iOS	47
3.1.2	Swift	49
3.1.3	Python	52
3.1.4	Weka	54
3.2	Implementazione e codice	54
3.2.1	Principali classi dell'applicazione	55
3.2.2	Estrazione dei dati sensoriali e calcolo della magnitude	56
3.2.3	Feature Extraction	58
3.2.4	Creazione Dataset	60
3.2.5	Creazione modello di predizione	62
3.2.6	Affdex utilizzo e implementazione	65
4	Valutazioni e risultati finali	69
4.1	Dataset	69
4.2	Algoritmi di classificazione	70
4.3	Analisi dei Risultati	71
4.3.1	Analisi risultati sottogruppi di sensori	74

5 Conclusioni e sviluppi futuri**77****Bibliografia****80**

Elenco delle figure

1.1	ADAS on car	3
1.2	Struttura generale di un'applicazione ADAS	5
1.3	Fase training e testing in App	6
1.4	Assi giroscopio e accelerometro integrati nello smartphone	8
1.5	Applicazione CarSafe	14
1.6	Applicazione DriveSafe	15
1.7	Emozioni primarie, Ekman, Lie to me	20
1.8	Struttura funzionamento sistema Facial Expression Recognition, J. Kurami, R. Rajesh, KM. Pooja	21
1.9	Struttura di una rete neurale convoluzionale	23
1.10	Convoluzione in CNN	24
1.11	Pooling in CNN	25
1.12	Architettura CNN	26
1.13	Logo Affectiva	27
1.14	Affdex for Market Research	29
1.15	Affectiva Automotive AI	29
1.16	Struttura AffdexSDK	31

2.1	Schema applicazione tesi	34
2.2	Calcolo magnitude per i file del dataset iniziale	40
2.3	Esempio posizionamento telefono in auto	43
2.4	View iniziale applicazione	44
2.5	Schemata setting applicazione	44
2.6	View spiegazione applicazione	45
2.7	View principale dove viene mostrato l'andamento del comportamento durante la guida	45
2.8	View fotocamera, modulo Affdex	46
2.9	View grafico settimanale	46
3.1	Logo iOS	48
3.2	Logo Swift	49
3.3	Logo CocoaPods	50
3.4	Modello CoreML	51
3.5	Logo Python	52
3.6	Logo libreria Pandas	53
3.7	Logo libreria Scikit-learn	53
3.8	Risultati cross-validation modello di classificazione	64
4.1	Funzionamento algoritmo RandomForest	70
4.2	Grafico che mette a confronto l'accuracy e il tempo di costruzione per ogni algoritmo di classificazione	72
4.3	Confusion matrix dell'algoritmo Random Forest e Bayesian Network	73
4.4	Accuracy nel dettaglio dell'algoritmo Random Forest	74

Elenco delle tabelle

1.1	Tabella che mostra la relazione tra espressioni facciali e i fattori predittivi delle emozioni	32
2.1	Tabella che mostra gli eventi di guida rilevati e salvati nel dataset	38
4.1	Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset	72
4.2	Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati del giroscopio	74
4.3	Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati dell'accelerometro Lineare	74
4.4	Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati dell'accelerometro	75
4.5	Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati del magnetometro	75

Capitolo 1

Stato dell'arte

Prima di presentare il progetto che ha portato alla realizzazione dell'applicazione sviluppata durante il periodo di tesi e spiegarne il funzionamento è necessario mostrare lo stato dell'arte relativo alle tecnologie utilizzate, al fine di giustificare le scelte implementative.

In questo capitolo vengono introdotti i concetti e le tecniche che si trovano alla base dei sistemi ADAS, spiegando cosa sono e quale è il loro contributo nel garantire la sicurezza alla guida. Inoltre si definisce l'insieme delle tecniche utilizzate per il riconoscimento delle espressioni-emozioni umane, ponendo l'attenzione sul caso Affectiva.

1.1 ADAS

Negli ultimi anni il continuo progresso tecnologico ha portato le aziende automobilistiche a dotare le automobili di sistemi elettronici di assistenza alla guida sviluppati per tutelare al massimo l'incolumità del guidatore e del passeggero. Questi ausili elettronici vengono indicati con l'acronimo **ADAS**, cioè Advanced Driver Assistance Systems.

1.1.1 Sistemi ADAS: cosa sono?

Con il termine **ADAS** (Advanced Driver Assistance Systems) [1] ci si riferisce a quei sistemi di assistenza alla guida che nei veicoli supportano il conducente in determinate situazioni e contribuiscono generalmente a incrementare i livelli di sicurezza e il comfort alla guida. Le caratteristiche di sicurezza sono progettate per evitare collisioni e incidenti e offrono tecnologie che allertano il conducente su potenziali problemi.

Ci sono due principali direzioni di ricerca in questo argomento:

1. Sviluppo e produzione di soluzioni hardware e software complesse che vengono integrate nei veicoli
2. Sviluppo di applicazioni mobile che tramite l'utilizzo di sensori riescono a comprendere situazioni pericolose.

Le soluzioni del primo gruppo sono molto costose e accessibili solo ai veicoli di alto-livello. Tali proposte offrono diversi tipi di caratteristiche di sicurezza, tra i quali ricordiamo:

- **Sistemi di mantenimento della corsia:** il sistema ADAS di mantenimento della carreggiata più avanzato è il Lane Keeping System che, oltre ad avvisare il conducente del superamento della linea di corsia, riporta automaticamente il veicolo nella carreggiata;
- **Sistema di avviso in caso di collisione:** grazie alla presenza di una videocamera o di un radar posto nella zona anteriore, il sistema di sicurezza ADAS riconosce le situazioni di pericolo ed avvisa il guidatore con un segnale acustico;
- **Sistema di visione notturna (NVS):** è la capacità di vedere in condizioni di scarsa illuminazione;

- **Park Assistance System (PAS):** grazie alla presenza di sensori di parcheggio, aiuta l'autista nello svolgere la manovra;
- **Sistema di riconoscimento del semaforo (TLRS)**
- **Sistemi di navigazione e mappa supportati.**

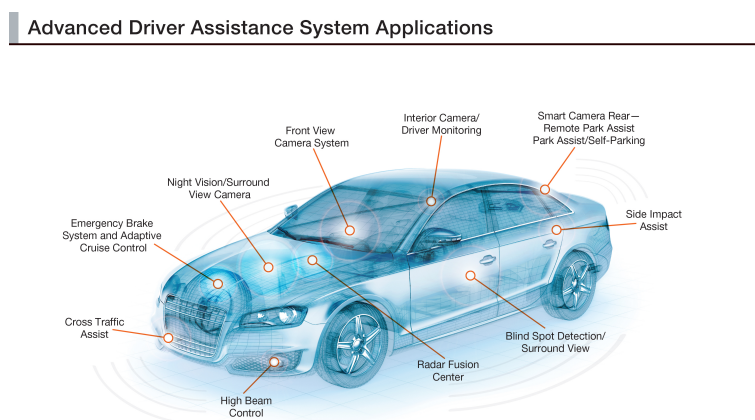


Figura 1.1: Esempio di caratteristiche di sicurezza ADAS su un'auto.

Così come descritto da A. Smirnov e I. Lashkov in [2], al momento sono disponibili sul mercato molti sistemi di assistenza alla guida; molte case automobilistiche sono dotate di soluzioni implementate nei veicoli. Per esempio, alcune di esse vengono offerte da:

- **Mobileye** [3]: è un'azienda che sviluppa sistemi avanzati di assistenza alla guida che forniscono avvertimenti per la prevenzione e la mitigazione delle collisioni. La tecnologia dell'azienda si basa sull'utilizzo di sistemi di visione ottica con algoritmi di rilevamento del movimento eseguiti su un acceleratore hardware personalizzato: il chip EyeQ

- **TRW** [4]: è un'azienda leader nel mercato delle ADAS. I loro sistemi sono basati su fotocamere e radar ed hanno lo scopo di controllare l'auto da tutti i lati in modo da evitare collisioni
- **Autoliv Inc.** [5]: ha sviluppato una telecamera per BMW che migliora la consapevolezza dei guidatori e in tal modo riduce gli incidenti
- **Subaru** [6]: gli ingegneri della casa automobilistica hanno sviluppato una tecnologia ADAS chiamata EyeSight. Tale sistema impiega due telecamere per catturare immagini. Grazie ad esse, EyeSight riconosce forma, velocità e distanza di veicoli. Quando rileva un potenziale pericolo il sistema avverte il guidatore e, quando necessario, provvede anche a frenare in modo da evitare incidenti.

Al contrario, le soluzioni del secondo gruppo ¹ sono gratuite o più economiche e sono indirizzate verso i guidatori che non hanno ancora accesso alle caratteristiche di sicurezza prima esposte.

Tali soluzioni suppongono l'utilizzo di uno smartphone per scaricare un'applicazione e utilizzarla. L'approccio basato sullo sviluppo di tali applicazioni è considerato una buona alternativa alle classiche implementazioni dei sistemi ADAS, in quanto meno costoso e soprattutto perché lo smartphone è ormai un dispositivo largamente diffuso nella società attuale.

Le applicazioni di questo tipo basano il loro funzionamento sull'analisi di una fusione dei dati provenienti dai diversi sensori presenti nello smartphone ed hanno come obiettivo, o il riconoscimento dello stile di guida di un conducente,- classificandolo come aggressivo o normale -, oppure la detenzione di situazioni pericolose, come il **DD** (Drowsy Driving) o **ID** (Inattentive Driving).

¹Sistemi ADAS implementati in applicazioni smartphone.

Prima di spiegare il rilevamento di questi comportamenti è utile introdurre la struttura generale delle applicazioni che simulano il funzionamento di un sistema ADAS.

1.1.2 Struttura generale dei sistemi ADAS su uno smartphone

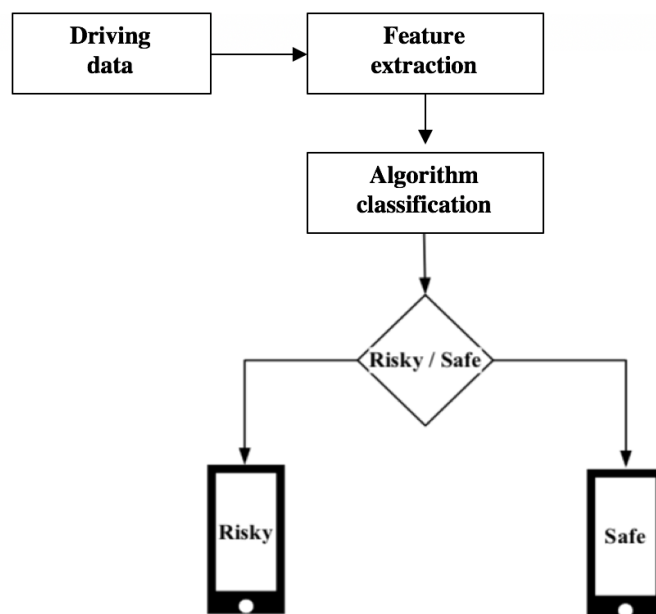


Figura 1.2: Struttura generale di un'applicazione ADAS

Come in Fig. 1.2, le applicazioni che svolgono il rilevamento del comportamento del conducente si compongono in diversi step: il primo step riguarda l'acquisizione dei dati grezzi e la pre-elaborazione di quest'ultimi per renderli puri attraverso la cosiddetta feature extraction dove i dati provenienti dai diversi sensori vengono raggruppati in finestre di tempo in modo tale da filtrare le informazioni rilevanti.

Successivamente, vengono utilizzati algoritmi di machine learning (algoritmi di classificazione) per generare un modello di riconoscimento delle operazioni svolte alla guida a partire dal set di features precedentemente estratte. Tale modello di apprendimento

preventivamente addestrato sarà utilizzato in seguito per generare una previsione del comportamento del guidatore in base ai dati raccolti durante la sessione di guida.

In altre parole possiamo dire che tali applicazioni si compongono di due fasi rilevanti: training e testing/evaluation.

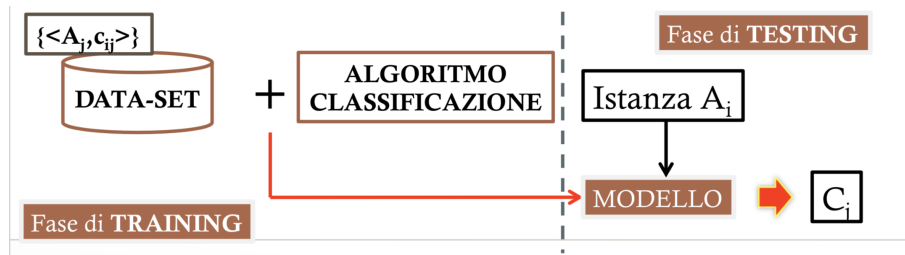


Figura 1.3: Fase di testing e training in un'applicazione basata su tecniche di data mining

Il training riguarda l'attività di estrazione, pre-elaborazione dei dati sensoriali e la loro collocazione in un training-set dopo l'applicazione di tecniche di feature extraction. Nella fase di testing vengono valutati i dati risultanti dalla fase precedente attraverso un modello d'apprendimento che classifica il comportamento del guidatore.

1.1.3 Driving data: estrazione dei dati sensoriali

Questo paragrafo ha lo scopo di descrivere brevemente i diversi tipi di sensori che sono presenti negli smartphone e che sono utilizzati per analizzare il comportamento del guidatore.

1.1.4 Sensori smartphone

Il sensore è un dispositivo di input che misura una quantità fisica in diretta interazione con il sistema misurato [7]. I suoi valori grezzi vengono trasformati per avere informazioni sull'ambiente e sugli utenti. Dato il loro prezzo contenuto vengono integrati negli smartphone moderni. Alcuni dei sensori usati per la raccolta di dati sono:

- Accelerometro a 3 assi
- Giroscopio a 3 assi
- Magnetometro
- Sensore di prossimità
- Fotocamera
- GPS
- Microfono.

In quest'ambito di ricerca spesso i dati di più sensori vengono aggregati per avere dei migliori risultati per riconoscere lo stile di guida, ad esempio nel sistema MIROAD (A Mobile-Sensor-Platform for Intelligent Recognition Of Aggressive Driving) [8] vengono usati: la fotocamera, l'accelerometro, il giroscopio, il magnetometro e il GPS per il riconoscimento dello stile di guida. In questa ricerca di tesi sono stati usati i seguenti sensori: accelerometro, giroscopio, magnetometro e fotocamera che vengono introdotti e spiegati da N. Jalra e D. Bansal in [9] e nelle sezioni successive.

Accelerometro

L'accelerometro è un dispositivo elettromeccanico che misura le forze d'accelerazione che subisce il dispositivo sui tre assi (x, y, z) in m/s^2 . Queste forze possono essere statiche (Asse z), come la forza di gravità, oppure dinamiche (Assi x, y) causate dagli spostamenti e/o dalle vibrazioni dello smartphone.

Giroscopio

Il giroscopio è un sensore che rileva la posizione corrente del dispositivo o i cambi di orientamento di quest'ultimo sui tre assi (x , y , z). Viene espresso in rad/s.

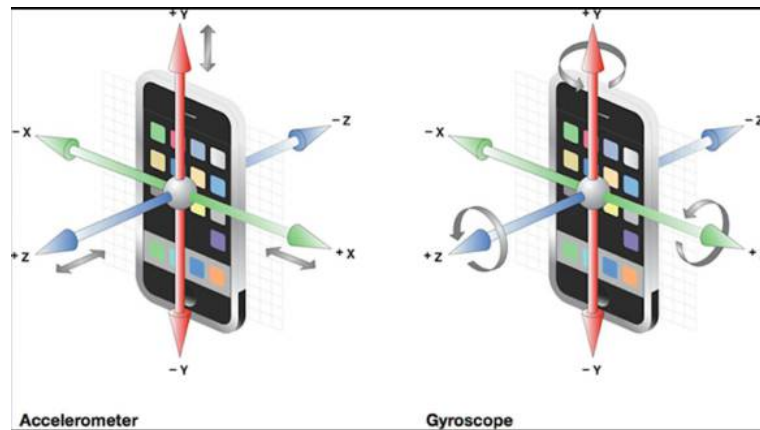


Figura 1.4: Direzione degli assi di giroscopio e accelerometro integrati nello smartphone

Magnetometro

Il magnetometro permette di misurare le onde magnetiche presenti nell'ambiente sugli assi (x , y , z) ed è capace di percepire la direzione verso il quale è rivolto il dispositivo. Esso riesce a "trasformare" lo smartphone in una vera e propria bussola capace di determinare l'inclinazione del campo magnetico.

Fotocamera

La fotocamera è un sensore che viene utilizzato per catturare delle immagini e stream video e gioca un ruolo importante per rilevare i comportamenti umani.

I dati risultanti da questi sensori sono utili per la fase di training e testing. La prima fase implica che tali dati, rilevati da una sessione di guida, siano salvati in un dataset. Nella

seconda fase i dati verranno confrontati con il modello di apprendimento per identificare se il comportamento di guida è sicuro oppure no.

1.1.5 Feature extraction

Una volta raccolti i data grezzi dai sensori si passa alla seconda fase di progettazione detta "feature extraction".

La feature extraction o estrazione delle caratteristiche è il processo di trasformazione degli input in un insieme di caratteristiche. Considerato il fatto che i dati provenienti dai diversi sensori sono numerosi per l'esecuzione di un algoritmo e che esiste il sospetto di ridondanza, questi dati attraverso la feature extraction verranno convertiti in una rappresentazione ridotta di un insieme di features accurate.

Tale processo viene anche applicato perché una singola rilevazione in uno specifico istante temporale non fornisce adeguate informazioni per descrivere la manovra svolta da un guidatore. Per tale motivo i dati devono essere salvati su delle strutture dati e su di queste applicare le tecniche di feature extraction, in modo tale che le attività vengano riconosciute basandosi su una finestra temporale piuttosto che su una singola rilevazione. Esistono diversi metodi di feature extraction ed in questa tesi si è utilizzato un metodo dove, considerando un finestra temporale di 5 secondi, per ogni sessione vengono estratti 4 valori per sensore: $\min(s,k)$, $\max(s,k)$, $\text{avg}(s,k)$ e $\text{std}(s,k)$ i quali rappresentano rispettivamente il minimo, il massimo, il valore medio e la varianza della magnitude² dei sensori nella sessione k.

²Grandezza che serve a condensare i dati degli assi di un sensore in un unico numero dato dalla formula $M = \sqrt{x^2 + y^2 + z^2}$ dove x, y e z sono i dati dei 3 assi del sensore analizzato.

1.1.6 Algoritmi di classificazione

Estrate le features, i nuovi dati accurati vengono inviati ad un classificatore che, seguendo algoritmi di machine learning, cerca di riconoscere all'interno dei dati i pattern necessari per il riconoscimento di comportamenti alla guida.

Gli algoritmi di classificazione [11] hanno lo scopo di determinare se gli attributi di una certa istanza appartengono o meno ad una classe. Negli algoritmi di classificazione i dati in input, - chiamati anche training set -, consistono in records ciascuno dei quali avente attributi o caratteristiche multiple. Inoltre ogni record è etichettato con un attributo di classe. L'obiettivo della classificazione è quello di analizzare i dati in input e sviluppare un'accurata descrizione o un modello per ogni classe usando le caratteristiche presenti nei dati.

Gli algoritmi di classificazione portano all'identificazione di schemi o insiemi di caratteristiche che definiscono la classe cui appartiene un dato record.

In generale, partendo dall'utilizzo di insiemi esistenti e già classificati, si cerca di definire alcune regolarità che caratterizzano le varie classi. Le descrizioni di tali classi vengono usate per classificare i records di cui non si conosce la classe di appartenenza, o per sviluppare una migliore conoscenza di ogni classe nel dataset. Esistono diversi algoritmi di classificazione ed in questa tesi di laurea sono stati utilizzati:

- **Alberi di decisione:** vengono utilizzati per classificare le istanze di grandi quantità di dati. Un albero di decisione descrive una struttura ad albero dove i nodi foglia rappresentano le classificazioni e le ramificazioni l'insieme delle proprietà che portano a quelle classificazioni. Di conseguenza ogni nodo interno risulta essere una macro-classe costituita dall'unione delle classi associate ai suoi nodi figli. Una sua evoluzione è la tecnica **Random forest** [12], ovvero, un classificatore d'insieme

che è composto da molti alberi di decisione e dà in uscita la classe che corrisponde all'output delle classi degli alberi presi individualmente.

- **Metodi Bayesiani:** I classificatori bayesiani sono classificatori statistici, possono prevedere la probabilità che una tupla appartenga a una particolare classe. La classificazione bayesiana si basa sul teorema di Bayes. **Bayesian Network (BN)** e **Naïve Bayes (NB)** sono i principali rappresentanti di questa classe di classificatori. Il classificatore NB (Naïve Bayes) [13] utilizza una tecnica statistica con la quale si cerca di stimare la probabilità di un'istanza di appartenere ad una certa classe. Formalmente le reti Bayesiane (BN, Bayesian network) [14] sono grafi diretti aciclici i cui nodi rappresentano variabili casuali in senso Bayesiano: possono essere quantità osservabili, variabili latenti, parametri sconosciuti o ipotesi. Gli archi rappresentano condizioni di dipendenza; i nodi che non sono connessi rappresentano variabili che sono condizionalmente indipendenti tra di loro. Ad ogni nodo è associata una funzione di probabilità che prende in input un particolare insieme di valori per le variabili del nodo genitore e restituisce la probabilità della variabile rappresentata dal nodo. Per esempio una rete Bayesiana potrebbe rappresentare la relazione probabilistica esistente tra i sintomi e le malattie. Dati i sintomi, la rete può essere usata per calcolare la probabilità della presenza di diverse malattie.
- **SMO** [15]: è un algoritmo per risolvere il problema di programmazione quadratica (QP) che si verifica durante l'addestramento di Support Vector Machines (SVM).

1.1.7 Fini e scopi delle app

I sistemi ADAS integrati all'interno di applicazioni mobile si possono distinguere in base al tipo di rilevazione delle situazioni pericolose di guida. In particolare, si possono

identificare differenti casi d'uso per le applicazioni di questo tipo:

- **Drowsy driving** (Sonnolenza): i conducenti stanchi sono soggetti a episodi di "micro-sonno", della durata variabile tra i 5 e i 10 secondi, durante i quali il cervello si addormenta. Gli individui sono spesso inconsapevoli di questi episodi e pensano di essere stati svegli per tutto il tempo o di aver perso momentaneamente la concentrazione. Di conseguenza i conducenti che soffrono di "micro-sonno" sono ad alto rischio di incidente.
- **Inattentive driving** (Distrazione): Mantenere il contatto visivo con la strada è fondamentale per una guida sicura. Esiste un ampio insieme di comportamenti distratti che possono anche sfociare in una guida disattenta, come conversazioni con i passeggeri, controllo delle gomme, sms, musica, osservazione e lettura delle mappe. Tali esempi vengo raggruppati nel concetto di disattenzione del guidatore.
- **Classificazione comportamento di guida**: molte applicazioni rilevano il comportamento di guida del conducente classificandolo in aggressivo o normale.

I primi due casi d'uso rilevano situazioni pericolose tramite l'utilizzo della telecamera, nel primo caso, andando a controllare la chiusura degli occhi o la frequenza di sbadigli in un determinato arco temporale e nel secondo caso, controllando la posizione della testa, facendo attenzione che quest'ultima sia rivolta verso la strada. Il sistema di interazione uomo-smartphone di A. Smirnov, I. Lashkov e A. Kashevnik [10] rileva le due situazioni prima citate attraverso l'uso della telecamera dello smartphone. Tale sistema mira a rilevare le situazioni pericolose mentre si guida un veicolo, in particolare, a tener traccia della posizione della testa e della chiusura degli occhi per rilevare se il guidatore sta bene oppure si è addormentato o è distratto.

L'ultimo caso trattato viene svolto attraverso l'utilizzo e l'analisi dei dati sensoriali prodotti dallo smartphone durante una sessione guida.

1.1.8 Applicazioni ADAS sul mercato

Oltre ad elencare i sistemi ADAS integrati nelle auto, A. Smirnov e I. Lashkov in [2], hanno descritto le applicazioni mobile che implementano un sistema avanzato di assistenza alla guida. Infatti, all'interno dei rispettivi store (Android ed Apple) è possibile trovare applicazioni ADAS. Le applicazioni più interessanti verranno descritte di seguito.

CarSafe

CarSafe [16] è un applicazione di guida sicura, per dispositivi Android, che rileva e avvisa i guidatori nel caso di comportamenti o condizioni di guida pericolose. Tale applicazione utilizza algoritmi di computer vision e machine learning per monitorare e rilevare se il conducente è stanco o distratto, utilizzando la fotocamera frontale. Allo stesso tempo vengono rilevate le condizioni della strada utilizzando la fotocamera posteriore. CarSafe utilizza un algoritmo context-aware che commuta le due telecamere mentre elabora i dati in tempo reale con l'obiettivo di ridurre al minimo la perdita di eventi sia all'interno (ad esempio, sonnolenza alla guida) che al di fuori dell'auto (ad esempio, mancata distanza di sicurezza). CarSafe utilizza altri sensori incorporati sul telefono (cioè sensori inerziali) per generare suggerimenti relativi a potenziali pericoli. I risultati di CarSafe sono promettenti in quanto può dedurre un insieme comune di comportamenti di guida pericolosi e condizioni stradali con una precisione complessiva dell'83%.

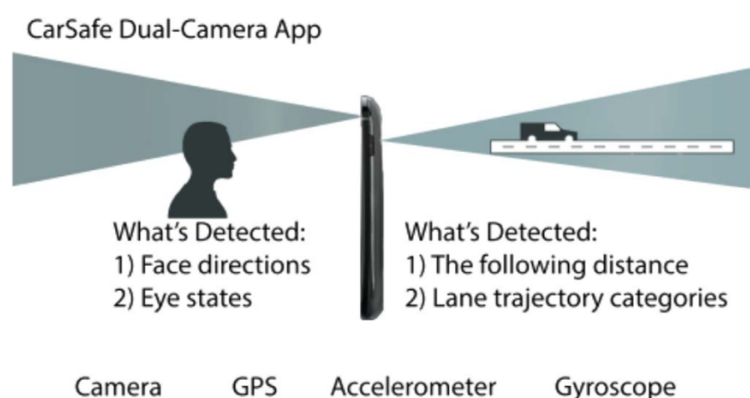


Figura 1.5: CarSafe utilizza due telecamere, una per monitorare lo stato del conducente, l'altra per controllare le situazioni stradali

iOnRoad

iOnRoad [17] è un'applicazione disponibile sia per la piattaforma iOS che per Android e fornisce un insieme di funzioni di assistenza alla guida che includono l'avvertimento di collisione e "black box". L'applicazione utilizza GPS, giroscopio e telecamera per monitorare la posizione del veicolo sulla strada.

DriveSafe

DriveSafe [18] è un'applicazione di guida sicura per iPhone che rileva comportamenti di guida disattenti e fornisce un feedback al conducente determinando la guida e avvertendolo nel caso il suo comportamento non sia sicuro. L'applicazione utilizza algoritmi di computer vision e tecniche di riconoscimento del modello per valutare se il guidatore è in stato di sonnolenza oppure distratto, usando la telecamera posteriore, il microfono, sensori inerziali e il GPS. Il sistema viene attivato quando la velocità del veicolo supera i 50 km/h perché progettato come assistente di guida per strada³ e non per uso urbano.

³Strada inteso come autostrada, superstrada, comunque lo studio si basa su rilevazioni di viaggi lunghi.

Inoltre, con la telecamera posteriore e l'applicazione di algoritmi quali, Canny algorithm e Hough transform algorithm, viene monitorato il mantenimento della corsia.



Figura 1.6: Applicazione DriveSafe in funzione

Nell'ambito di un applicazione ADAS, particolare attenzione va posta al consumo della batteria e alla violazione della privacy. Il consumo di batteria è causato principalmente dall'utilizzo di fotocamera e GPS che richiedono uno sforzo importante al telefono, il quale deve essere collegato ad un alimentatore durante sessioni di guida lunghe ⁴. Al contrario Eren, H.; Makinist, S.; Akin, E.; Yilmaz in [19], propongono un sistema che utilizzi solo l'accelerometro, il giroscopio e il magnetometro ottenendo dei risultati utili per analizzare statisticamente il comportamento del guidatore. Considerando il fatto che non vengono utilizzati sensori esterni riesce ad essere un sistema semplice, leggero e non invasivo.

La violazione della privacy attraverso un applicazione ADAS è dovuta all'utilizzo di fotocamera, GPS e microfono. Un caso eclatante è [10] di A. Smirnov, I. Lashkov e A.

⁴Vedi ad esempio [18], [16]

Kashevnik, dove viene proposto un sistema di consigli contenuti nel Cloud, dove lo smartphone dopo aver rilevato con la telecamera una situazione di sonnolenza o distrazione controlla se è disponibile la connessione internet, se presente, invia suggerimenti del tipo fermarsi al bar più vicino, fare un "riposino" in un motel, in base alla sua posizione.

Il sistema proposto in questa tesi cerca di preservare il consumo energetico e la violazione dei dati personali, utilizzando solo la fotocamera solo dopo il rilevamento da parte dei sensori inerziali di manovre di guida non sicure.

1.2 Analisi e riconoscimento espressioni-emozioni facciali

La nostra faccia è un eccellente organo di comunicazione emotiva. I primi studi sull'espressioni facciali risalgono al XIX secolo da parte di C. Darwin con la pubblicazione de "L'espressione delle emozioni nell'uomo e negli altri animali" (1872) nel quale si dimostra come le espressioni facciali fra uomo e animale siano universali ed innate, in quanto frutto dell'evoluzione. Sulla base di tali studi, nel 1971 Ekman e Friesen individuarono come l'espressione delle emozioni avviene tramite l'attivazione di determinati muscoli facciali che costituiscono, nel loro lavoro, le cosiddette Action Unit.

Successivamente, a partire dagli anni Novanta tali studi sono stati applicati nel campo informatico con l'obiettivo di realizzare sistemi automatici robusti in grado di esprimere una "tecnologia emotivamente intelligente" e, conseguentemente, di adattare il proprio comportamento, in base alle emozioni rilevate[20].

In questa sezione verranno introdotti i concetti che stanno alla base del riconoscimento espressioni-emozioni facciali, in particolare si adotterà un approccio top-down, partendo dall'affective computing fino ad arrivare a vedere AffDex sdk.

1.2.1 Affective computing

Dalla nascita dei primi computer, l'Information Technology è sempre stata considerata come connessa a quelle correnti psicologiche di stampo prettamente cognitivistico, in cui il processamento e l'elaborazione dell'informazione diviene elemento fondante l'analogia tra mente e computer.

Una data critica in questo scenario è il 1997, anno in cui Rosalind Picard dal Massachusetts Information Technology Lab (MIT) pubblica il libro "Affective computing" [22] lanciando una sfida: fino a che punto e secondo quali modalità l'affettività e l'emotività possono entrare a far parte del mondo della tecnologia informatica?

A partire dalla fine degli anni novanta sono state condotte, e sono tutt'ora in fase di progettazione, ricerche che hanno portato alla realizzazione di tecniche computerizzate in grado di riconoscere, esprimere, modellare, comunicare informazioni emotive; in questo senso la finalità idealistica è avvicinarsi a un'interazione uomo-macchina, in cui i due partner interattivi si comprendano e si adattino reciprocamente l'uno l'altro in maniera flessibile.

Nasce così un'area innovativa dell'intelligenza artificiale, chiamata **affective computing** che propone di realizzare calcolatori in grado di riconoscere, interpretare, elaborare ed esprimere emozioni. E' un campo interdisciplinare che abbraccia l'informatica, la psicologia e le scienze cognitive.

Taranpreet Singh Saini, Dr. MangeshBedekar, Saniya Zahoor in [23], individuano due aree di ricerca dell'affective computing:

Rilevazione e riconoscimento delle emozioni

La rilevazione delle informazioni emotive inizia con sensori passivi che acquisiscono dati sullo stato o sul comportamento fisico dell'utente senza interpretare l'input. I dati

raccolti sono analoghi ai segnali che gli umani usano per percepire le emozioni negli altri esseri umani. Ad esempio, una videocamera potrebbe catturare espressioni facciali, postura del corpo e gesti, mentre un microfono potrebbe catturare la voce. Altri sensori rilevano segnali emotivi misurando direttamente i dati fisiologici come la temperatura della pelle. L'insieme di questi diversi metodi valutativi è una condizione ideale per un accurato e preciso riconoscimento dello stato affettivo del soggetto. Riconoscere le informazioni emotive richiede l'estrazione di modelli significativi dai dati raccolti. Questo processo viene fatto attraverso vari metodi come optical flow, Hidden Markov Models, o elaborazione di reti neurali. Queste metodologie possono essere combinate per fornire un stima dello stato emotivo del soggetto.

Emozioni nelle macchine

Tale area di ricerca si riferisce alla progettazione di dispositivi che siano in grado di simulare le emozioni in maniera convincente e di esibirle. Un approccio più pratico, basato sulle attuali capacità tecnologiche, è la simulazione delle emozioni attraverso la conversazione al fine di arricchire e facilitare l'interattività tra uomo e macchina [21].

1.3 Analisi delle espressioni: tra informatica e psicologia

Come detto precedentemente, l'affective computing può essere considerato come una terra di incontro tra informatica, architetture cognitive e patterns di espressione di riconoscimento di informazioni emotivo-affettive. Nelle sezioni successive saranno descritti gli aspetti principali che si trovano alla base del riconoscimento delle emozioni sia dal punto di vista psicologico che dal punto di vista informatico.

Psicologia

Le aziende immerse nel mondo dell'affective computing fanno tutto affidamento sul lavoro di Paul Ekman [24], uno psicologo statunitense che, a partire dagli anni '60, ha raccolto un convincente corpo di prove dove afferma l'esistenza di almeno sei emozioni universali espresse dal volto in modo identico tra tutti gli esseri umani, indipendentemente da genere, età o educazione culturale. Ekman lavorò per decodificare queste espressioni, suddividendole in combinazioni di quarantasei movimenti individuali, chiamate "unità d'azione". Da questo lavoro, compilò insieme a Friesen, nel 1978 il **Facial Action Coding System**, o **FACS**: una tassonomia di cinquecento pagine sui movimenti facciali che comprende una descrizione sistematica in parole, fotografie e filmati di come misurare i movimenti facciali in termini anatomici, scomponendoli in singole unità di movimento (Action Unit). Tale sistema è stato utilizzato per decenni da accademici professionisti e ufficiali di polizia interessati alla comprensione dell'inganno o di bugie. Ekman ha dimostrato che, contrariamente alla convinzione precedente di alcuni antropologi, le espressioni facciali e le emozioni non sono determinate dalla cultura di un posto o dalle tradizioni, ma sono universali ed uguali per tutto il mondo, indicando il fatto che sono di origine biologica. Nel 1987 egli ha condotto un esperimento per stabilire quanto influisse il contesto sociale, testando un gruppo di americani e uno di giapponesi, rilevando come i giapponesi apparissero meno espressivi solo in compagnia, mentre da soli esprimevano le stesse emozioni primarie. Nel 1972, seguendo una tribù isolata dal mondo in Papua Nuova Guinea, ha redatto le espressioni di "base" universali, utilizzate tutt'ora dai software per il riconoscimento delle emozioni:

- Neutro
- Rabbia

- Disgusto
- Tristezza
- Gioia
- Paura
- Sorpresa



Figura 1.7: Esempi espressioni primarie realizzate nella serie tv Lie To Me, basata sul lavoro di P. Ekman

Ognuna di queste emozioni può essere riconosciuta nel proprio viso ed in quello di tutte le persone, a prescindere da etnia, cultura, genere, religione e, con un opportuno allenamento, è possibile riconoscerle e capire la reale emozione che una persona prova, persino se quella persona siamo noi stessi. Grazie ad Ekman oggi esistono computer capaci di distinguere un sorriso di circostanza dalla gioia spontanea; sfruttando la loro attenzione instancabile e la memoria illimitata sono capaci di registrare anche espressioni che non sappiamo nemmeno di avere, perché troppo brevi per essere registrate dall'occhio umano e per la nostra capacità di "leggere" gli altri.

Informatica

Come accennato prima, un metodo di comunicazione non verbale da cui capire l'umore o lo stato d'animo di una persona è l'espressione della faccia. Il riconoscimento dell'espressione facciale (FER) in maniera automatica è diventato un interessante e stimolante area di ricerca per la computer vision e viene usato nell'implementazione di social robot, nei videogiochi e in applicazioni di neuromarketing. J. Kurami, R. Rajesh, KM. Pooja in [25] descrivono il funzionamento di un sistema FER (Facial expression recognition). Tale sistema si compone attraverso 5 passi, come mostra la figura seguente.

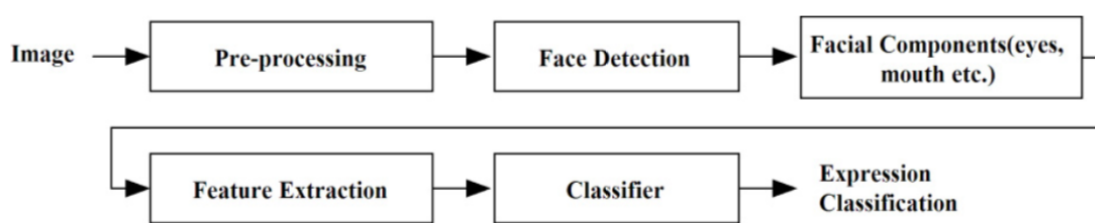


Figura 1.8: Diagramma a blocchi di classificazione dell'espressione facciale

Il miglioramento dei dati viene eseguito nella fase di pre-elaborazione, prendendo in input una singola immagine o una sequenza di immagini⁵ e fornendo al volto ulteriori elaborazioni. Il rilevamento dei componenti del viso rileva il ROI⁶ per occhi, naso, guance, bocca, sopracciglia, orecchio, fronte-testa, ecc. La fase di features extraction riguarda l'estrazione delle informazioni rilevanti dalle ROI. Le tecniche di estrazione delle feature più popolari sono: Gabor filters, Local Binary Patterns (LBP), Principal Component Analysis (PCA), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA), Local Gradient Code (LGC), Local Directional Pattern (LDP). La fase di "Clas-

⁵Una serie temporale di immagini da un'espressione neutra a un'espressione significativa.

⁶Etichettatura di forma rettangolare attorno all'area del viso presa in questione.

sifier” classifica le caratteristiche nelle rispettive classi di espressioni facciali basate sui metodi di classificazione, alcuni dei più popolari sono SVM (Support Vector Machine) e NN (Nearest Neighbor).

I cambiamenti dell’espressione facciale possono essere basati sia sulla creazione di piccole rughe sul volto oppure su deformazioni maggiori del viso. Tra le tecniche di features extraction vi sono: Geometric based e Appearance based, Action Unit di un gruppo di muscoli o Non-AU, Local Versus Holistic. I metodi che si basano su idee geometriche, controllano la posizione e lo spostamento della testa o del componente facciale che è preso in considerazione. I metodi Action Unit si basano sulle idee introdotte nel FACS da Ekman.

In [26], viene descritto un sistema per il riconoscimento delle emozioni facciali che sfrutta le potenzialità di una rete neurale convoluzionale.

Tale studio afferma come metodi di riconoscimento delle emozioni possono essere suddivisi in due gruppi principali: il primo gruppo lavora su immagini statiche e il secondo lavora su sequenze di immagini dinamiche. Negli approcci statici, le informazioni temporanee non vengono considerate e usano solo le informazioni correnti dell’immagine, mentre negli approcci dinamici vengono utilizzate le informazioni temporali per riconoscere le emozioni espresse nelle sequenze di fotogrammi.

Il riconoscimento automatico delle espressioni delle emozioni comprende tre passaggi: acquisizione dell’immagine del volto, estrazione delle caratteristiche e riconoscimento dell’espressione delle emozioni facciali. Come detto precedentemente, i processi di feature extraction usati per il riconoscimento delle emozioni si possono dividere in due approcci: metodi basati su feature geometriche e metodi basati sull’aspetto. Nei primi metodi vengono considerati posizione e forma delle parti del viso, mentre nei secondi metodi vengono considerate regioni particolari o tutto il viso.

Reti Neurali convoluzionale

Convolutional Neural Networks

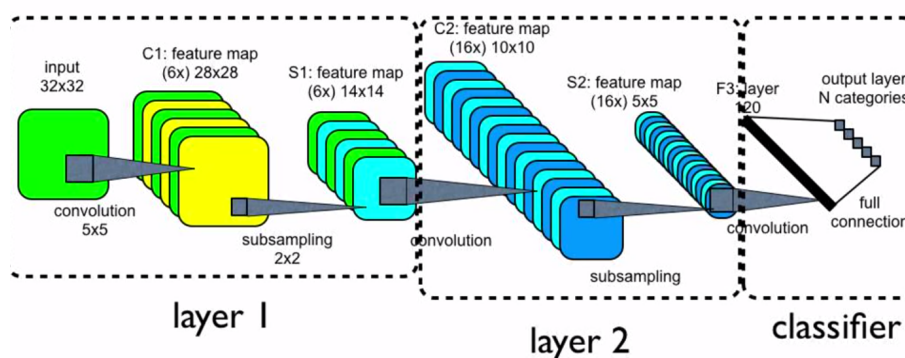


Figura 1.9: Struttura e processi di base di una rete neurale convoluzionale

Nel machine learning, una rete neurale convoluzionale (CNN o ConvNet dall'inglese convolutional neural network) è un tipo di rete neurale artificiale feed-forward in cui il pattern di connettività tra i suoi neuroni è ispirato dall'organizzazione della corteccia visiva animale, i cui neuroni individuali sono disposti in maniera tale da rispondere alle regioni di sovrapposizione che tassellano il campo visivo [27].

Le reti neurali convoluzionali sono reti specializzate nel processamento di dati che presentano una struttura a griglia e si sono dimostrate molto efficaci in aree quali il riconoscimento e la classificazione delle immagini. Una CNN ha un'architettura basata su una gerarchia di livelli: Convolutional layer, Sub-sampling layers, Rectified linear unit (ReLU), Fully connected layer, Output layer and Softmax layer.

Dall'immagine prima introdotta si può capire la struttura e i processi di base di una rete neurale convoluzionale, ovvero: la convoluzione, il pooling (subsampling), la normaliz-

zazione, la strato di connessione totale e lo strato di apprendimento.

- **Convoluzione:** è una delle più importanti operazioni di image processing attraverso la quale si applicano filtri digitali. Un filtro digitale è fatto scorrere sulle diverse posizioni di input (immagine analizzata); per ogni posizione viene generato un valore di output, eseguendo il prodotto scalare tra la maschera e la porzione dell'input coperta. In parole più semplici possiamo dire che lo strato di neuroni che si occupa della convoluzione divide l'immagine in vari frammenti sovrapposti, che sono in seguito analizzati per individuare le particolarità che lo caratterizzano, trasferendo l'informazione allo strato seguente sotto forma di una "feature map" contenente le relazioni tra neuroni e particolarità.

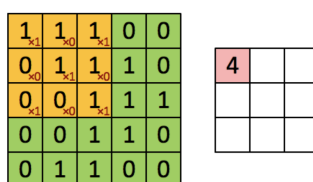


Figura 1.10: Processo di convoluzione

- **Pooling o Sub Sampling:** Un livello di pooling esegue un'aggregazione delle informazioni nel volume di input, generando feature map di dimensione inferiore. Obiettivo è conferire invarianza rispetto a semplici trasformazioni dell'input mantenendo al tempo stesso le informazioni significative ai fini della discriminazione dei pattern. Il pooling è un'operazione che prende in ingresso una serie di input e li riduce ad un singolo valore, un po' come l'operazione di subsampling effettuata dal supersampling anti-aliasing che isola quattro sotto pixel e li riduce ad uno solo effettuando una media. Così facendo, si possono semplificare alcune caratteristiche della nostra immagine per meglio adattarle al calcolo tramite una convolutional

neural network. Processo molto utilizzato per il pooling è il cosiddetto max-pooling, mostrato nell'immagine seguente dove per ciascuna delle regioni rappresentate dal filtro si prende il massimo di quella regione e si crea una nuova matrice di output in cui ogni elemento è il massimo di una regione nell'input originale.

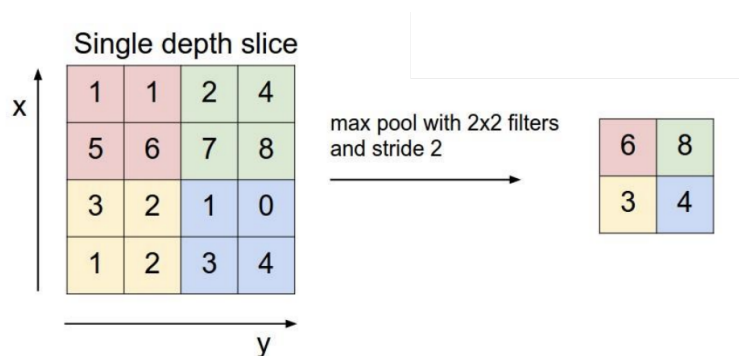


Figura 1.11: Processo di max pooling

- Normalizzazione:** è utilizzata per evitare le anomalie in seguito a vari passaggi negli strati della rete neurale. Le funzioni più utilizzate sono la Rectified Linear Units (ReLU), la tangente iperbolica e la funzione sigma, anche se la ReLU è la migliore poiché più rapida. ReLU sta per Rectified Linear Unit ed è un'operazione non lineare. Lo scopo di questo livello è quello di introdurre la non linearità in un sistema che fondamentalmente è stato in grado di calcolare operazioni lineari durante i livelli convoluzione. Questo livello aumenta le proprietà non lineari del modello e della rete globale senza influenzare i campi ricettivi del livello convoluzionale.
- Classificazione e apprendimento:** Lo strato di connessione totale (full connection layer) è l'ultimo strato nascosto della rete neurale, nel quale tutti gli input dei vari neuroni sono messi insieme, permettendo alle particolarità di essere identificate. Infine, lo strato finale per tutte le reti neurali è quello di apprendimento (loss

layer) che permette al sistema di modificare i valori associati ai neuroni sulla base della correttezza dei risultati emessi. Il metodo di allenamento più utilizzato è la backpropagation.

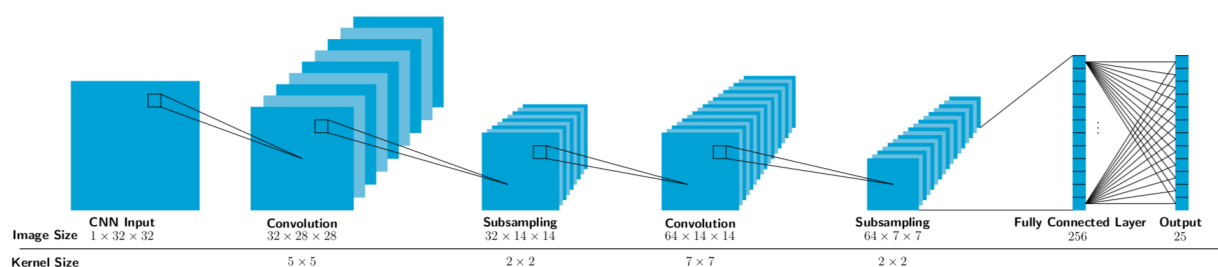


Figura 1.12: Architettura CNN proposta in [26]

Nel metodo proposto dagli autori di [26], al fine di classificare sette stati emotivi di base, viene utilizzato il framework psicologico precedentemente introdotto, Facial Action Coding System (FACS) per migliorare la precisione di rilevamento. L'architettura CNN proposta per il rilevamento delle AU include quattro parti: si prendono in input delle immagini e si processano attraverso dei filtri, dopo ogni livello convoluzionale, vengono seguite le funzioni di normalizzazione (ReLU), viene applicato un processo di max pooling e infine per la classificazione delle unità d'azione si considera il livello di softmax⁷ con 25 neuroni che indicano numeri di unità d'azione completamente connessi allo strato precedente.

Tali sistemi vengono addestrati e testati su dataset che comprendono immagini di espressioni emotive. Uno dei più importanti dataset è l'Extended Cohn-Kanade (CK+) [35], utilizzato da molte aziende nel settore dell'affective computing.

⁷Funzione che genera direttamente la probabilità per la classe di riferimento

1.3.1 Affectiva



Figura 1.13: Logo Affectiva

Tra la miriade di start up nel settore dell'affective computing, una delle più promettenti è **Affectiva** [31], fondata nel 2009 dalla scienziata egiziana Rana El Kaliouby che attraverso AffDex⁸ riesce a riconoscere ed interpretare 21 espressioni facciali umane, sulla base dei video raccolti in 75 paesi diversi. La nuova tecnologia sviluppata è in grado di leggere le espressioni facciali e abbinarle alle emozioni corrispondenti, le sei emozioni principali, indicando la valenza (quanto è positiva o negativa l'esperienza che una persona sta vivendo) e l'engagement (quanto una persona è espressiva). Per capire le emozioni, si scansionano le espressioni facciali, mappandole in punti e costruendo modelli dettagliati di come ciascun volto può reagire a un diverso stimolo.

La tecnologia che si trova alla base del sistema Affectiva si basa sull'apprendimento profondo (in inglese deep learning), campo di ricerca dell'apprendimento automatico e dell'intelligenza artificiale che si basa su diversi livelli di rappresentazione, corrispondenti a gerarchie di caratteristiche di fattori o concetti, dove i concetti di alto livello sono definiti sulla base di quelli di basso [32]. Tali algoritmi consentono alle aziende di intelligenza artificiale come Affectiva di modellare problemi più complessi con maggiore accuratezza

⁸AffDex si basa sugli studi dello psicologo Paul Ekman, in particolare, sul suo sistema Facial Action Coding System (FACS).

rispetto alle altre tecniche di machine learning.

Inoltre, il deep learning risolve una varietà di problemi (classificazione, segmentazione, modellazione temporale) e consente l'apprendimento end-to-end di una o più attività complesse congiuntamente. L'indirizzo delle attività svolte da Affectiva include rilevamento e tracciamento dei volti, rilevamento delle attività vocali e classificazione delle emozioni da viso e voce.

Per risolvere questi diversi compiti l'azienda utilizza una suite d'architetture di deep learning:

- Reti neurali convoluzionali (CNN)
- Reti neurali ricorrenti (RNN).

Il sistema viene utilizzato principalmente in due campi d'azione:

- **AffDex for Market Research** [29]: è una soluzione basata su cloud che utilizza AffDex per il riconoscimento delle emozioni. Tale soluzione offre una visione approfondita delle risposte emotive dei consumatori non filtrate e imparziali ai contenuti digitali. Tutti gli strumenti di cui si ha bisogno sono la connettività Internet e una webcam standard. Mentre gli spettatori guardano uno spot, Affdex for Market Research misura le espressioni facciali e le emozioni momento per momento. I risultati vengono aggregati e visualizzati in una dashboard facile da usare.

AffDex for Market Research viene usato da marchi, inserzionisti e ricercatori di mercato in quanto si è visto come le emozioni influenzano il comportamento dei consumatori.

- **Affectiva Automotive AI** [30]: usando telecamere e microfoni in cabina, Affectiva Automotive AI analizza espressioni facciali e vocali per identificare espressioni,

emozioni e reazioni delle persone in un veicolo. Tale soluzione si propone di monitorare, attraverso l'analisi del viso e della voce, sia lo stato del conducente che quello dei passeggeri.

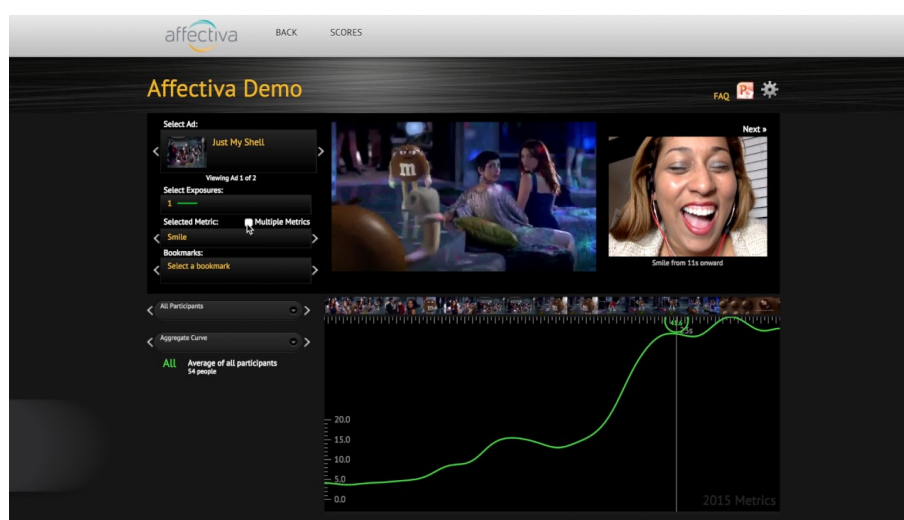


Figura 1.14: Funzionamento del software AffDex for Market Research

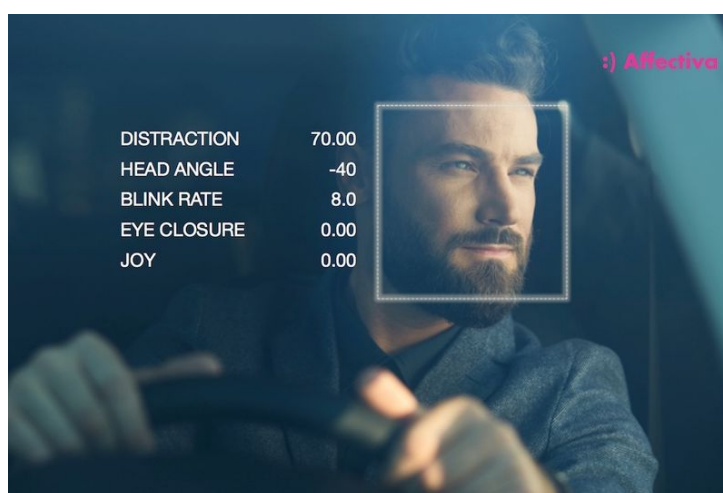


Figura 1.15: Affectiva Automotive AI

Affectiva [31] utilizza una curva ROC (Receiver Operating Characteristic) per segnalare la precisione del proprio sistema di rilevamento delle emozioni/espressioni, in quanto viene considerato il modo più generalizzato per misurare la precisione di un rilevatore. I valori della curva ROC vanno da 0 a 1 e più il valore è vicino ad 1 più il classificatore è preciso. Molte espressioni facciali, come il sorriso, il solco delle sopracciglia, il sopracciglio della fronte interna, il sollevamento delle sopracciglia e la ruga del naso hanno un punteggio ROC superiore a 0,9. Alcune espressioni facciali più sfumate, che sono molto più difficili da identificare anche per gli esseri umani, includono il movimento delle labbra e la chiusura degli occhi. Questi hanno un punteggio ROC di oltre 0,8.

AffDex

In questa sezione viene presentato AffDex software development kit (SDK) [33]. L'SDK è progettato per analizzare le espressioni facciali spontanee che le persone mostrano nelle loro interazioni quotidiane, esso basa il suo funzionamento su algoritmi di computer vision e machine learning. Gli algoritmi di computer vision identificano i punti chiave sul viso, ad esempio gli angoli delle sopracciglia, la punta del naso, gli angoli della bocca. Gli algoritmi di apprendimento automatico (classificatori) analizzano i pixel in quelle regioni per classificare le espressioni facciali. L'utilizzo dell'SDK è disponibile sulle principali piattaforme mobile e desktop.

AffDex basa il suo funzionamento su quattro passaggi: rilevamento facciale e dei punti di riferimento (ROI) nel volto, features extraction dei dati facciali, classificazione delle azioni facciali e modellizzazione delle espressioni. La Fig. 1.16 mostra tali passaggi.

Il rilevamento dei volti viene eseguito utilizzando l'algoritmo di rilevamento dei volti Viola-Jones [34]. Il rilevamento dei punti di riferimento viene applicato a ciascun riquadro di delimitazione facciale e ai 34 punti di riferimento identificati. Se il livello di

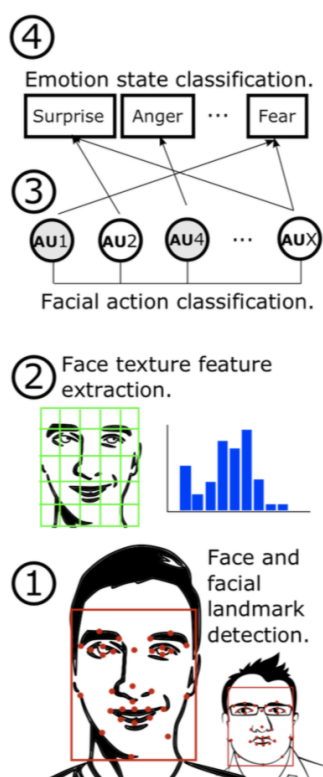


Figura 1.16: Struttura operazioni Affdex SDK per il rilevamento delle emozioni

confidenza del rilevamento dei punto di riferimento è inferiore a una soglia, il riquadro di delimitazione viene ignorato.

Le emozioni (Rabbia, Disgusto, Paura, Gioia, Tristezza, Sorpresa e Disprezzo) sono basate su combinazioni di azioni facciali, il sistema usa le espressioni facciali osservate come input per calcolare la probabilità di un'emozione. La mappatura delle emozioni si basa, come detto precedentemente, sulle mappature EMFACS sviluppate da Friesen ed Ekman. La Tab. 1.1 mostra la relazione tra le espressioni facciali e i fattori predittivi delle emozioni.

Le espressioni/emozioni vengono classificate attraverso un classificatore SVM (Support Vector Machine) che fornisce un punteggio che parte da 0 (assente) fino ad arrivare a

100 (presente) per ogni azione facciale.

Il software è disponibile all'indirizzo <http://www.affectiva.com/sdk>⁹.

Emotion	Increase Likelihood	Decrease Likelihood
Joy	Smile	Brow Raise Brow Furrow
Anger	Brow furrow Lid Tighten Eye Widen Chin Raise Mouth Open Lip Suck	Inner Brow Raise Brow Raise Smile
Disgust	Nose Wrinkle Upper Lip Raise	Lip Suck Smile
Surprise	Inner Brow Raise Brow Raise Eye Widen Jaw Drop	Brow Furrow
Fear	Inner Brow Raise Brow Furrow Eye Widen Lip Stretch	Brow Raise Lip Corner Depressor Jaw Drop Smile
Sadness	Inner Brow Raise Brow Furrow Lip Corner Depressor	Brow Raise Eye Widen Lip Press Mouth Open Lip Suck Smile
Contempt	Brow Furrow Smirk	Smile

Tabella 1.1: Tabella che mostra la relazione tra espressioni facciali e i fattori predittivi delle emozioni

Affdex SDK è stato utilizzato in questa tesi di laurea per valutare l'attenzione e la sonnolenza di un guidatore, come meglio descritto nel prossimo capitolo.

⁹Gli utenti, una volta effettuata la registrazione come sviluppatori, possono utilizzare l'sdk offerto dall'azienda, in base alla loro piattaforma che desiderano utilizzare.

Capitolo 2

Progettazione di un'applicazione ADAS

L'obiettivo di questa tesi è stato quello di costruire un'applicazione mobile che riconosca gli eventi di guida pericolosi e una volta rilevati, controlli, tramite il software Affdex prima introdotto, le condizioni del conducente avvisandolo nel caso in cui venga riscontrata una situazione potenziale di distrazione o sonnolenza.

Basandomi su quanto proposto da alcune delle applicazioni ADAS che sono state elencate nello stato dell'arte, si è deciso di realizzare un sistema che tramite il riconoscimento dello stile di guida attraverso i sensori inerziali attivi la fotocamera solo nel caso in cui si sia riscontrata una situazione di guida pericolosa attraverso una tecnica di sliding window, cercando così di minimizzare il consumo di batteria e di rispettare la privacy del conducente.

L'applicazione si occuperà di raccogliere i dati dai sensori e di convertirli in una rappresentazione ridotta di un insieme di caratteristiche attraverso una tecnica di feature extraction. Queste feature accurate saranno inviate al modello di classificazione che avrà il compito di predire i comportamenti di guida, classificandoli in sicuri e non sicuri.

Il modello di classificazione è stato creato sulla base di un dataset di dati scaricato da internet [35], utilizzato in [36]. Si è preferito adottare questa soluzione in quanto non si aveva a disposizione un'automobile e perché il rilevamento di dati di guida presuppone la prova di manovre pericolose e azzardate, e quindi più adatte a piloti esperti. Anche nei documenti consultati [18], [16], [8] si è adottata questa soluzione per garantire l'incolumità del guidatore.

In questo capitolo viene trattato il funzionamento del sistema e la sua progettazione specificando la sua architettura e le componenti utilizzate.

2.1 Architettura e scelte progettuali

In questa sezione vengono presentati l'architettura dell'applicazione e scelte metodologiche che sono state adottate. L'applicazione segue lo schema presentato nella Fig. 2.1.

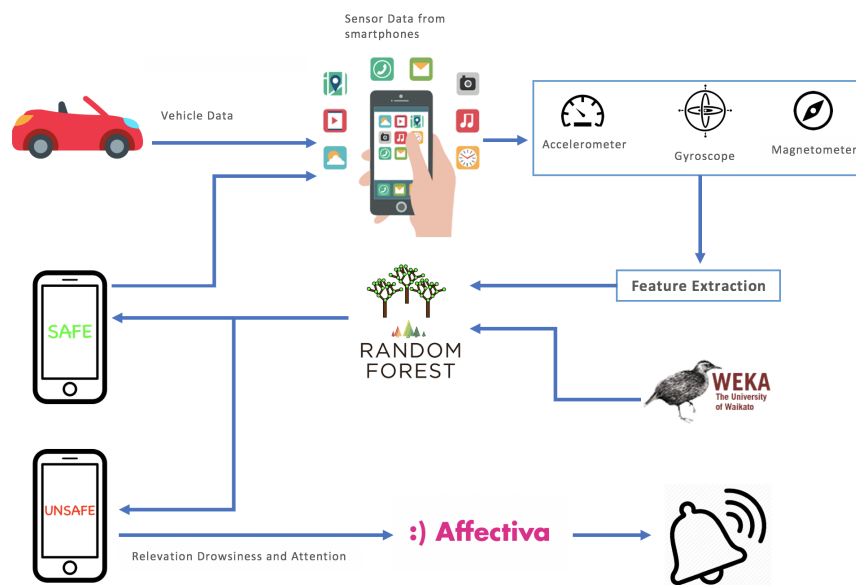


Figura 2.1: Schema applicazione AffectiveDrive

La metodologia utilizzata per sviluppare un applicazione ADAS prevede di utilizzare una serie di strumenti,- hardware e software -, che hanno permesso la realizzazione del sistema attraverso varie fasi di studio, dalla raccolta dei dati, fino alla classificazione per predire il comportamento del guidatore.

L'applicazione è stata sviluppata in iOS in quanto in possesso di uno smartphone Apple ed era impossibile sfruttare un simulatore (Android/iOS) dato che non si possono estrarre i dati sensoriali da un simulatore.

Successivamente, verranno descritte nel dettaglio le diverse operazioni che hanno permesso la realizzazione dell'applicazione mobile.

2.1.1 Estrazione dei dati

La prima fase per sviluppare un applicazione ADAS è quella di realizzare un'applicazione iOS in grado di campionare i dati sensoriali con frequenza fissa. I sensori inerziali utilizzati nell'applicazione sono stati scelti in base ai sensori presenti nel dataset prima citato e sono: accelerometro, giroscopio e magnetometro. L'accelerometro misura le variazioni di accelerazione subite dal device. Esso sfrutta le variazioni di accelerazione per determinare se ci si sta spostando o meno, in quale direzione ci si sta spostando e altre informazioni riguardanti la posizione e il movimento del dispositivo mobile o di chi lo indossa e lo sta utilizzando. Il giroscopio insieme all'accelerometro è uno dei sensori principali degli smartphone ed è utilizzato dai dispositivi mobili per individuare ogni tipo di movimento. Infine il magnetometro misura l'intensità e la direzione di un campo magnetico.

I dati grezzi del sensore sono fondamentalmente composti da valori basati sui 3 assi (asse x, asse y, asse z). Tuttavia, tali dati non vengono direttamente inviati al classificatore. Prima di raggrupparli in serie temporali, applicando così le tecniche di feature extraction,

essi vengono concentrati in un unico valore detto **magnitude**.

Magnitude

La **magnitude** è una grandezza che serve a condensare i dati di un sensore in un unico valore e viene calcolata tramite formula $Magnitude = \sqrt{x^2 + y^2 + z^2}$ dove x, y e z sono assi del sensore preso in esame. Tale valore viene calcolato all'interno della fase di estrazione e trattamento dei dati grezzi e tale operazione viene effettuata in quanto essa, a differenza del set di 3 valori, non dipende dall'orientamento del dispositivo al momento del rilevamento. Per tale motivo non c'è bisogno di imporre al conducente l'obbligo di posizionare lo smartphone in una specifica direzione. Questo ci permette di avere una stima dei sensori del dispositivo in termini assoluti.

2.1.2 Feature extraction

Come detto nel capitolo precedente, i dati ottenuti dai sensori sono dati raw (cioè ottenuti direttamente dalla fonte) e quindi non utilizzabili allo stato attuale. Per poter sfruttare tali dati è necessario effettuare un passaggio di feature extraction per sintetizzarne le caratteristiche e permetterne il confronto con altri dati simili. Nel nostro caso specifico, ogni volta che viene calcolata la magnitude di uno specifico sensore, tale valore viene salvato in una struttura dati apposita, dove con un intervallo di registrazione pari a 5 secondi vengono estratte le features utili per essere confrontate con il modello di classificazione. Per ogni struttura relativa a un sensore vengono create sequenze di dati frutto della divisione dei dati prima ottenuti in sessioni di 5 secondi. Per ogni sessione vengono estratti le seguenti features:

- **Minimo**: valore minimo registrato dal sensore in esame durante una determinata sequenza

- **Massimo:** valore massimo registrato dal sensore in esame durante una determinata sequenza
- **Media:** media dei dati registrati dal sensore in esame durante una determinata sequenza
- **Varianza:** varianza dei dati registrati dal sensore durante una determinata sequenza.

2.1.3 Creazione del modello di classificazione

In questa sezione verranno presentate le tecniche di costruzione del modello di classificazione partendo da un set di dati fornito da J. J. Ferreira et al. in [36].

Dataset

Come detto nelle sezioni precedenti si è preferito non costruire un dataset di dati basato sulle rilevazioni del mio smartphone, ma utilizzare un set di dati trovato in rete. Il dataset in questione è disponibile presso l'indirizzo <https://github.com/jair-jr/driverBehaviorDataset>.

Il set di dati è una raccolta di misurazioni effettuata con un'applicazione Android per registrare i dati sensoriali dello smartphone, come accelerometro, accelerazione lineare (priva della forza di gravità), magnetometro e giroscopio durante una sessione di guida. L'esperimento si è basato sui dati raccolti in 4 viaggi in auto di circa 13 minuti ciascuno. Il lavoro ha visto coinvolti due piloti, con più di 15 anni di esperienza di guida che hanno eseguito specifiche manovre, e uno smartphone ¹ che è stato fissato sul parabrezza dell'auto per mezzo di un supporto evitando degli spostamenti di quest'ultimo durante

¹Motorola XT1058 con versione Android 5.1

la raccolta dei dati provenienti dai diversi sensori. Lo scopo di [36] era quello di stabilire una serie di eventi di guida che rappresentassero le solite manovre del mondo reale come frenate, accelerazioni, svolte e cambi di corsia. La tabella seguente mostra i 7 tipi di eventi di guida rilevati in questo lavoro e il loro numero di campioni raccolti.

Driving Event Type	Number of samples
Aggressive breaking	12
Aggressive acceleration	12
Aggressive left turn	11
Aggressive right turn	11
Aggressive left lane change	4
Aggressive right lane change	5
Non-aggressive event	14
Total	69

Tabella 2.1: Tabella che mostra gli eventi di guida rilevati e salvati nel dataset

Il file dataset scaricato risulta essere composto da quattro cartelle², ognuna di esse rappresenta un viaggio contenente i seguenti file:

1. **aceleracaoLinear_terra.csv**: File con accelerazione lineare su 3 assi.
2. **acelerometro_terra.csv**: File con l'accelerazione su 3 assi.
3. **campoMagnetico_terra.csv**: File con le letture del magnetometro su 3 assi.
4. **giroscopio_terra.csv**: File con le letture del giroscopio su 3 assi.
5. **groundTruth.csv**: File con i timestamp di inizio e fine per ciascun evento.
6. **viagem.json**: Informazioni sullo smartphone utilizzato.

I file .csv relativi ai sensori ³ sono composti dalle misurazioni dei 3 assi (x, y e z) del sensore preso in analisi, dal timestamp del momento in cui viene preso il record di

²Le cartelle sono nominate 16, 17, 20, 21

³cfr file1, 2, 3, 4 prima elencati

dati e dall'uptimeNanos, ovvero, il tempo in nanosecondi che segnala l'inizio dell'evento rilevato.

Il file groundTruth.csv è utile per etichettare i dati delle rilevazioni ed è composto da i seguenti campi:

- **evento**: Tipo di manovra svolta.
- **inizio**: Inizio dell'evento in secondi.
- **fm**: Fine dell'evento in secondi

Per il lavoro di tesi, si è scelto di utilizzare il file 17, contenente le misurazioni dei sensori che rilevano delle manovre relative a frenate brusche e accelerazioni violente.

I file si presentano come descritto precedentemente, ovvero, composti dal valore dei sensori sui 3 assi (x, y e z), timestamp e uptimeNanos, e quindi non hanno al loro interno un valore unico, come la magnitude, che possa condensare i dati relativi agli assi del sensore in un unico numero. Per questa ragione ho costruito, partendo dai file presenti nella cartella 17, un file .csv con i seguenti attributi:

- Magnitude Accelerometer Linear
- Magnitude Accelerometer Terra
- Magnitude Gyroscope
- Magnitude Magnetometer
- Behavior.

I primi quattro attributi sono stati ottenuti calcolando per ogni record del file preso in esame la magnitude relativa ad un preciso timestamp. Ad esempio nella figura seguente,

per il file `aceleracaoLinear_terra.csv` (come per tutti gli altri) si è calcolato per ogni riga la magnitude prendendo in input i valori degli assi. I valori ottenuti dai quattro file sono stati riportati in un nuovo file `.csv`, al quale è stata aggiunta la colonna "Behavior" che rappresenta l'etichetta da attribuire ad ogni record. Tale etichettature viene attribuita ad ogni records del file confrontando i timestamp del file ottenuto con i dati riportati nel file `groundTruth.csv`.

Dataset finale

magnitudeAccelerometerLinear	magnitudeAccelerometerTerra	magnitudeGyroscope	magnitudeMagnetometer	safe/unsafe
0.06598445643622408	9.85222547044493	0.057486282604537316	17.551820425492572	True
0.19479193178474097	9.63463408016614	0.029549717615580156	17.607815637599305	True
0.3946016172737377	9.515997419153672	0.042830322998900786	17.734588055896616	True
0.2327495365410524	9.673433251695617	0.02346867584585791	17.73458914316593	True
0.06361028517325605	9.845074235110818	0.04024299525216674	17.689818710759404	True

Magnitude = $\sqrt{x^2 + y^2 + z^2}$

acelerometro_terra

timestamp	uptimeNanos	x	y	z
14/05/2016 11:17:09	12893233616460	-0.04759973571752418	0.004669870245576746	9.852109377080389
14/05/2016 11:17:09	12893242986214	-0.06699311435252042	0.06098352149071218	9.634208155911663
14/05/2016 11:17:09	12893253210538	-0.2001309613255451	0.17045380619558603	9.512365635298703
14/05/2016 11:17:09	12893272865658	-0.18531473840756973	-0.03981773343299366	9.67157607998012

Figura 2.2: Passaggio dal file iniziale al dataset finale, con il calcolo della magnitude

Modello di classificazione

Una volta completate le operazioni per la creazione del training set, quest'ultimo viene analizzato attraverso il software **Weka**[45] dove vengono applicati dei metodi di apprendimento automatico (learning methods) al dataset in esame al fine di analizzarne

i risultati. È possibile quindi attraverso questi metodi, avere una previsione dei nuovi comportamenti dei dati. Gli algoritmi di machine learning applicati sul dataset di dati sono:

- ZeroR
- Random Forest
- Bayesian Network
- Naive Bayes
- SMO

L'output dell'elaborazione degli algoritmi di classificazione di Weka prevede un'analisi dei dati attraverso Confusion Matrix e indici di precisione come precision e accuracy. Dall'analisi dei dati abbiamo evidenziato che l'algoritmo RandomForest produce i risultati migliori in termini di accuratezza e quindi su tale algoritmo si è deciso di costruire un modello di predizione nonostante richieda un tempo abbastanza lungo per costruire tali modelli. I risultati dell'analisi sono descritti nel Capitolo 4.

2.1.4 Classificazione e valutazione delle situazioni pericolose

Siamo giunti quasi alle fasi finali della progettazione dell'applicazione, infatti, dopo aver costruito il modello di classificazione e averlo integrato all'interno dell'app è ora possibile predire il comportamento di guida di un conducente tramite i dati prodotti dallo smartphone durante una sessione di guida.

Questi dati vengono inviati a una funzione che tramite l'implementazione dell'algoritmo di machine learning, classifica i dati producendo come output True o False⁴.

⁴True e False sono le etichette utilizzate per etichettare i record del dataset, True indica un comportamento sicuro, mentre False indica un comportamento pericoloso

Valuzione della situazione di pericolo

La valutazione della situazione di pericolo avviene tramite un metodo di sliding window, ovvero, il comportamento viene considerato pericoloso nel momento in cui si hanno tre⁵ valori False consecutivi nella struttura dati all'interno del quale vengono salvati i comportamenti predetti dal modello.

Una volta riconosciuta la situazione pericolosa la gestione passa il controllo al modulo Affectiva.

2.1.5 Modulo Affectiva

Il modulo Affectiva ha il ruolo di rilevare lo stato di sonnolenza e distrazione del guidatore durante la sessione di guida. E' stata adottata questa soluzione, cioè monitorare queste situazioni solo dopo aver rilevato un comportamento non sicuro con i dati dei sensori inerziali, per non sovraccaricare la batteria e non costringere il guidatore ad avere lo smartphone collegato ad un'alimentatore durante la sessione di guida.

Tramite il software Affdex, il sistema ogni qualvolta rileva un volto riceve per ogni situazione tracciata un valore compreso tra 0 e 100. I casi d'uso presi in azione sono il rilevamento della chiusura degli occhi, l'apertura della bocca e l'attenzione, quest'ultima intesa come la continuità del conducente di avere la testa rivolta verso la strada. Non avendo trovato molto nella letteratura dello stato dell'arte, si è deciso di avvisare, tramite un'allarme sonoro, il conducente ogni qualvolta il valore prodotto da AffDex superi una soglia prefissata. Tale soglia può essere modificata dall'utente.

⁵Il valore 3, è un valore di default. L'utente può impostare una soglia che varia da 0 a 5. Tali passaggi verranno spiegati di seguito

2.2 Funzionalità dell'applicazione

L'applicazione sviluppata si occupa della rilevazione del comportamento di guida del guidatore classificandolo come "sicuro" "pericoloso"; inoltre, dopo aver rilevato un atteggiamento pericoloso viene attivato il funzionamento della fotocamera, che permette, tramite AffDex sdk, di rilevare delle possibili situazioni di sonnolenza e distrazione alla guida.

L'applicazione per sfruttare al meglio le proprie funzionalità presuppone che lo smartphone prima della sessione di guida venga posizionato su un supporto attaccato al parabrezza o allo specchio retrovisore in modo che sia possibile rilevare il volto - vedi la figura seguente.

L'applicazione implementa diverse funzionalità che saranno descritte di seguito.



Figura 2.3: Esempio di posizionamento del telefono nell'auto

Guida dell'app

1. La pressione del bottone "Star Trip" indica al sistema di cominciare a tracciare i sensori. L'utente può cliccando il bottone impostazioni o quello di spiegazione dell'app.



Figura 2.4: View iniziale applicazione

2. L'utente premendo il bottone impostazioni può decidere di iniziare a tracciare il proprio viaggio con le soglie⁶ di default oppure cambiarle secondo le sue esigenze.

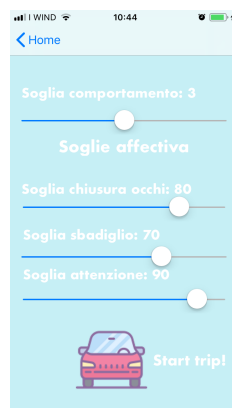


Figura 2.5: Schemata setting applicazione

⁶Le soglie sono state impostate con tali valori dopo aver svolto varie prove ed aver analizzato i diversi risultati riportati dal sistema

3. L'utente può leggere una guida dell'applicazione cliccando il bottone "?"



Figura 2.6: View spiegazione applicazione

4. Durante la fase di rilevazione del comportamento dell'utente, egli/ella visualizza sullo schermo la classe predetta dal sistema.

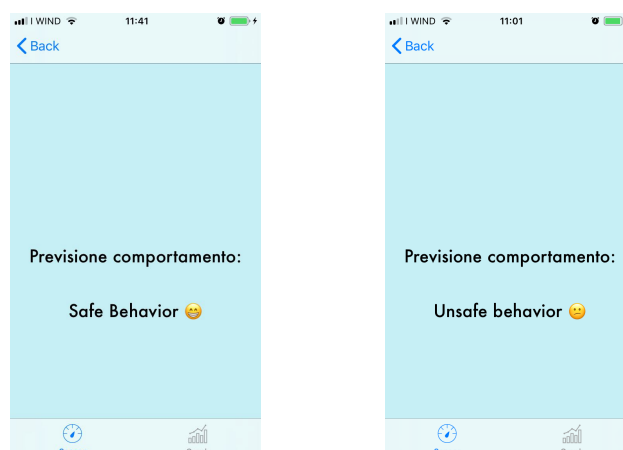


Figura 2.7: View principale dove viene mostrato l'andamento del comportamento durante la guida

5. Rilevata la situazione pericolosa si attiva la fotocamera che traccia il volto dell'utente e lo avverte nel caso i dati rilevati superino le soglie impostate nel sistema.

Con il bottone "Stop" si arresta il rilevamento del volto e si ritorna all'analisi dei dati sensoriali

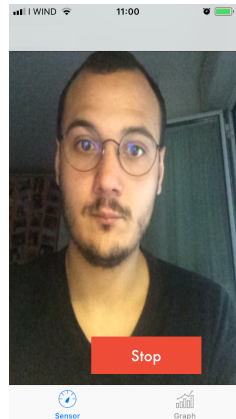


Figura 2.8: View rilevazione sonnolenza/distrazione

6. L'utente può controllare l'andamento del comportamento di guida settimanale grazie ad un grafico che rappresenta i comportamenti sicuri e non rilevati nella settimana.

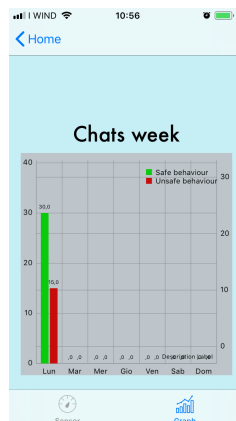


Figura 2.9: View applicazione dove visualizzare l'andamento di guida settimanale grazie ad un grafico

Capitolo 3

Implementazione

In questo capitolo vengono illustrate le scelte implementative che hanno portato alla realizzazione dell'applicazione spiegando nel dettaglio software, linguaggi, librerie e algoritmi utilizzati.

3.1 Strumenti

L'applicazione è stata sviluppata sulla piattaforma iOS e implementata su un iPhone 5s, utilizzando come ambiente di sviluppo xCode 9.3 e il linguaggio Swift. Per la costruzione del dataset e la creazione del modello di classificazione è stato utilizzato il linguaggio Python con delle librerie open-source di quest'ultimo, Scikit-learn, Pandas, coremltools. Per la valutazione di accuracy e precision lo strumento utilizzato è stato Weka.

3.1.1 iOS

iOS [37] è un sistema operativo sviluppato da Apple per iPhone, iPod touch e iPad. Esso è una derivazione di UNIX e usa un microkernel XNU Mach basato su Darwin OS.

iOS ha quattro livelli di astrazione: il Core OS layer, il Core Services layer, il Media layer e il Cocoa Touch layer.

Dopo la presentazione dell'AppStore ¹ ed il rilascio dell'SDK ufficiale per la realizzazione di applicazioni, il sistema operativo ha immediatamente suscitato molto interesse da parte di singoli sviluppatori e grandi software-house. Le applicazioni per iOS hanno subito un incremento esponenziale e una diffusione grandissima, dovuta principalmente alla grande popolarità e soprattutto alle enormi vendite dei dispositivi Apple.

Il 17 ottobre 2007, in una lettera aperta scritta nel blog How News di Apple, Steve Jobs ha annunciato che un SDK (software development kit) sarebbe stato disponibile agli sviluppatori di terze parti in febbraio 2008. L'SDK permette agli sviluppatori di creare applicazioni e testarle in un simulatore. L'ambiente di sviluppo (IDE) per iOS SDK è Xcode, esso contiene una suite di strumenti utili allo sviluppo di software per i sistemi macOS, iOS, watchOS e tvOS.



Figura 3.1: Logo iOS

¹Il negozio virtuale nel quale acquistare le applicazioni per i dispositivi Apple

3.1.2 Swift

Swift [38] è il linguaggio di programmazione per sistemi iOS e OS X presentato da Apple durante il WWDC 2014. E' un linguaggio di tipo object-oriented progettato per coesistere con Objective-C², ma rendendo la scrittura del codice più semplice. Swift è stato progettato per essere più resistente agli errori nel codice.

Il progetto nasce nel 2010, portato avanti da un team guidato da Chris Lattner. Si ispira agli altri linguaggi di programmazione più diffusi, come ObjectiveC, Rust, Haskell, Ruby, Python ecc.. La presentazione ufficiale avviene nel 2014, quando viene rilasciata al pubblico la prima app sviluppata in questo linguaggio. Swift è un linguaggio di tipo compilato. Grazie al compilatore LLVM è possibile creare programmi che contengono parti di codice in diversi linguaggi (C, Objective-C, C++ e Swift). Esso ha infatti il vantaggio di avere una sintassi e un funzionamento notevolmente più semplici e simili ad altri linguaggi più comuni.



Figura 3.2: Logo Swift

Ora verrà mostrato un esempio del classico programma che scrive in output "Hello, world!".

```
import UIKit
print("Hello, world!")
```

La funzione "import" importa nel programma determinati metodi o classi che permettono a quest'ultimo di implementare determinati comandi. "UIKit" è un framework

²Linguaggio utilizzato tipicamente per gli stessi scopi

sviluppato da Apple che fornisce le funzioni fondamentali per gestire la GUI e gli input dell'utente. In questo caso il codice compila anche senza importare UIKit (è stato importato solo come esempio) in quanto l'unica funzione utilizzata è stata la funzione "print" che fa parte del core del linguaggio.

La funzione "print" permette di mostrare a schermo ciò che viene scritto tra le virgolette dentro le parentesi tonde. In Swift non è necessario inserire il punto e virgola (;) alla fine di ogni istruzione.

CocoaPods



Figura 3.3: Logo CocoaPods

Molto spesso certe interfacce o funzionalità più complesse richiedono giorni per essere sviluppate e magari anche competenze che non possediamo. Una soluzione a questo problema viene rappresentata da CocoaPods.

Cocoapods [39] è un portale, o sarebbe più realistico definirlo un aggregatore, di risorse gratuite sviluppate da altri sviluppatori. Le risorse sono rese disponibili alla community per poterle integrare nelle nostre future applicazioni.

Le risorse o come vengono generalmente chiamate **Pods**, spaziano dalla grafica alle funzionalità dall'interfaccia utente fino a complesse animazioni per le nostre app. Per esempio si può integrare agilmente nell'applicazione: un'animazione, lo stile di un bottone che ci piace, un particolare tipo di menu, una fotogallery e così via.

In quest'applicazione sono state integrate le risorse che hanno permesso di sfruttare le

funzionalità del software AffDex, la creazione e la gestione del grafico tramite la libreria Charts.

CoreML

Core ML è stato utilizzato per integrare un modello di apprendimento automatico addestrato³ nell'app sviluppata per questo lavoro.

Core ML [40] è un framework annunciato da Apple nel corso della Worldwide Developers Conference WWDC17. Si tratta di un framework per il machine learning che viene utilizzato dalle app Apple come fotocamera, Siri e QuickType ma è utilizzabile anche dagli sviluppatori di terze parti.

Core ML consente di integrare vari meccanismi di machine learning nelle app. Supporta oltre 30 tipi di livelli, metodologie di apprendimento quali le macchine a vettori di supporto (SVM, dall'inglese Support Vector Machines) o l'apprendimento ensemble e una serie di metodi d'insieme che usano modelli multipli per ottenere una migliore prestazione predittiva. La tecnologia Core ML è sfruttabile da applicazioni che si occupano di Vision (individuare testi, volti, punti chiave, codici a barre, tracciare oggetti, ecc.) e per identificare in profondità testi tenendo conto di lemmi, token, parti di parlato, entità.



Figura 3.4: Modello CoreML

³Un modello addestrato è il risultato dell'applicazione di un algoritmo di apprendimento automatico a una serie di dati di addestramento. Il modello effettua previsioni basate su nuovi dati di input.

3.1.3 Python

Python [41] è un linguaggio di programmazione ad alto livello, orientato agli oggetti, adatto, tra gli altri usi, per sviluppare applicazioni distribuite, scripting, computazione numerica e system testing. Esso è un linguaggio multiparadigma, che ha tra i principali obiettivi dinamicità, semplicità e flessibilità. Supporta il paradigma object oriented, la programmazione strutturata e molte caratteristiche di programmazione funzionale e riflessione. Le caratteristiche più immediatamente riconoscibili di Python sono le variabili non tipizzate e l'uso dell'indentazione per la definizione delle specifiche. Altre caratteristiche distintive sono l'overloading di operatori e funzioni tramite delegation, la presenza di un ricco assortimento di tipi e funzioni di base e librerie standard, sintassi avanzate quali slicing e list comprehension.

Python all'interno di questo lavoro di tesi è stato utilizzato per due scopi differenti: il primo è stato quello di trasformare i dataset in un unico set di dati finale che contenga le magnitude dei diversi sensori analizzati e un attributo che rappresenti la classe. Il secondo è stato creare un modello di predizione da poter caricare all'interno dell'applicazione. Queste operazioni sono state rese possibili grazie all'utilizzo delle librerie Scikit-learn, Pandas e coremltools.



Figura 3.5: Logo Python

Pandas

Pandas [42] è una libreria software scritta per il linguaggio di programmazione Python per la manipolazione e l'analisi dei dati. In particolare, offre strutture dati e operazioni

per manipolare tabelle numeriche e serie temporali.



Figura 3.6: Logo libreria Pandas

Pandas viene utilizzato in questo lavoro per leggere i dati di un file .csv.

Scikit-learn

Scikit-learn [43] è una libreria open source di Python che fornisce molteplici strumenti per il machine learning, in particolare contiene numerosi algoritmi di classificazione, regressione e clustering (raggruppamento), macchine a vettori di supporto, regressione logistica, e molto altro.



Figura 3.7: Logo libreria Scikit-learn

Tale libreria è stata utilizzata per costruire il modello di classificazione.

Coremltools

Coremltools [44] è un pacchetto python per la creazione, l'esame e il test di modelli di classificazione in formato .mlmodel⁴. Essa è stata usata per convertire il modello di classificazione creato dalla libreria Scikit-learn in un formato accettabile per CoreML.

⁴Formato richiesto da CoreML[40]

Tali librerie sono state usate in maniera congiunta per creare un modello di classificazione, partendo da un dataset di dati in formato .csv, per poi convertirlo in formato .mlmodel.

3.1.4 Weka

WEKA [45] (Waikato Environment for Knowledge Analysis) è un ambiente open-source interamente scritto in Java che offre diverse funzionalità⁵ e una semplice interfaccia per attività di data mining. In questo lavoro di tesi il software è stato utilizzato applicando dei metodi di apprendimento automatici (learning methods) ad un set di dati (dataset), e analizzando i risultati.

3.2 Implementazione e codice

L'applicazione sviluppata è stata chiamata AffectiveDrive e la versione minima supportata corrisponde all'aggiornamento iOS 11.0, tale che quest'ultima sia compatibile solo con dispositivi che supportano una versione 11.0 o superiore. Non è stato possibile allargare il raggio di applicabilità a tutti i dispositivi, in quanto, il modulo CoreML è supportato solo dai dispositivi che dispongono dall'aggiornamento 11.0 o successivi. In tale sezione verranno descritti gli algoritmi usati per l'implementazione dell'applicazione. In particolare, verrà analizzato il codice utilizzato per realizzare le operazioni introdotte nel capitolo precedente, andando a ripercorre i passi che hanno portato alla creazione dell'app.

⁵WEKA consiste in una collezione di algoritmi di machine learning

3.2.1 Principali classi dell'applicazione

Prima di spiegare il codice, è utile elencare e descrivere le classi principali che compongono l'applicazione:

- **AppDelegate:** è il file che governa l'app e viene generato in automatico con la creazione del progetto. Con questo file possiamo gestire determinate situazioni come: l'uscita dall'app, il rientro nell'applicazione e la gestione delle attività che possono essere eseguite in background
- **ViewController:** classe che gestisce la View e potenzialmente tutti i suoi elementi all'interno. Essa si occupa di estrarre i dati sensoriali, calcolare la magnitudine e predire il comportamento del guidatore
- **moduleAffectiva:** classe che si occupa del rilevamento del volto e di di situazioni di sonnolenza e distrazione. Tale classe utilizza l'sdk Affectiva prima descritto
- **moduleCharts:** classe che realizza il grafico dei comportamenti di guida settimanali. Il grafico è stato creato grazie all'installazione nel Podfile della libreria Charts[46]
- **moduleImpostazioni:** la classe che permette all'utente di modificare i valori delle soglie per il rilevamento di situazioni pericolose
- **gestioneDB:** in questa classe vengono gestite le operazioni che dialogano con CoreData, utilizzato solo per la creazione del grafico
- **HomeController:** classe che gestisce la grafica della prima View (schermata iniziale)

- **behaviorModel**: costruzione del modello per la gestione dei dati del grafico.

Inoltre, l'applicazione contiene il modello `modelWeka.mlmodel` (modello di classificazione costruito con l'ausilio di Python e integrato nell'applicazione) e un Data Model dal nome `Project_AffSens.xcdatamodeld` un modello di dati CoreData. Il Core Data è uno strumento intermedio o ponte che fa da tramite tra la l'applicazione iOS e la memorizzazione dei dati in memoria. Inoltre ha diversi strumenti che vanno al di là del semplice salvataggio delle informazioni dell'applicazione. Esso è stato creato da Apple per la memorizzazione dei dati in memoria e per essere quanto più vicino ai concetti di programmazione.

3.2.2 Estrazione dei dati sensoriali e calcolo della magnitudine

Il primo passo per la costruzione dell'applicazione è il campionamento dei sensori. L'estrazione dei dati comincia con l'avvio della funzione `motionManager.startSensorUpdate`. Una volta che sono stati ottenuti i valori degli assi del sensore viene calcolata la magnitudine e i valori ottenuti vengono salvati in tre diversi array rispettivamente per i tre sensori analizzati. Tali operazioni sono contenute nella classe `ViewController.swift` e il codice utilizzato viene descritto di seguito.

```
import CoreMotion
import UIKit
class ViewController: UIViewController {
    let motionManager = CMMotionManager()
}
```

La funzione "import CoreMotion" importa le funzioni fondamentali per gestire i dati relativi al movimento dello smartphone. Esso comprende la gestione di accelerometro, giroscopio, contapassi, magnetometro e barometro.

Nella classe ViewController viene creata una costante che ci permetterà di usare le funzioni del framework CoreMotion. La dimostrazione di quanto appena scritto viene mostrata nel codice seguente.

```
//Avvia gli aggiornamenti dell'accelerometro
    motionManager.startAccelerometerUpdates()
//Intervallo, in secondi, per fornire aggiornamenti dell accelerometro al
    gestore di blocchi
motionManager.accelerometerUpdateInterval = 0.1
    //Avvia gli aggiornamenti accelerometro su una coda di operazioni
    motionManager.startAccelerometerUpdates(to: OperationQueue.current!)
{ (data, error ) in
    if let accelerometerData = data
    {
        //Dati accelerometro Asse-X
        self.AxisX_varAcc = accelerometerData.acceleration.x

        //Dati accelerometro Asse-Y
        self.AxisY_varAcc = accelerometerData.acceleration.y

        //Dati accelerometro Asse-Z
        self.AxisZ_varAcc = accelerometerData.acceleration.z

        //Calcolo magnitudine
        self.accelerometerMagnitude_var = sqrt(self.AxisX_varAcc *
self.AxisX_varAcc + self.AxisY_varAcc * self.AxisY_varAcc +
self.AxisZ_varAcc * self.AxisZ_varAcc)

        self.accelerometerMagnitude_array.append(self.accelerometerMagnitude_var)

        print("Accelerometer Magnitude: "
self.accelerometerMagnitude_array)
    } }
}
```

Nel dettaglio il codice mostra l'estrazione dei valore degli assi dell'accelerometro e il calcolo della Magnitude. Il primo passo è avviare gli aggiornamenti dell'accelerometro tramite la funzione `startAccelerometerUpdates()`. Una volta fatto ciò viene impostato il tempo di aggiornamento del valore dell'accelerometro (0,1 s). L'ultimo passaggio è la creazione della funzione `startAccelerometerUpdates(to queue: OperationQueue)` dove viene avviato l'accelerometro su una coda di operazioni. Eseguiti tali step vengono estratti uno alla volta i valori degli assi, calcolata la magnitude, e quest'ultimo valore viene aggiunto in un array. Tale processo sarà ripetuto per il giroscopio e il magnetometro.

3.2.3 Feature Extraction

Per ogni array di magnitude viene applicata la tecnica di feature extraction. Con un intervallo temporale di 5 secondi (time window) vengono estratte per ciascun sensore s durante una sequenza k le seguenti features accurate: $\text{minimo}(s, k)$, $\text{massimo}(s, k)$, $\text{valore medio}(s, k)$ e infine la $\text{varianza}(s, k)$. Il codice seguente mostrerà le funzioni che sono state costruite nella classe `ViewController.swift` per calcolare le feature.

```
//Funzione per il calcolo del minimo e del massimo
func calcoloMinMax(array: [Double]) -> (min: Double, max: Double) {
    var minimo = array[0] //setto minimo e massimo uguali al primo valore
    dell'array
    var massimo = array[0]

    for valore in array {
        if valore < minimo {
            minimo = valore
        } else if valore > massimo {
            massimo = valore
        }
    }
}
```

```
        return (minimo, massimo)
    }
//Funzione per il calcolo della media
    func calcoloMedia(array: [Double]) -> (Double) {
        var sommaValoriMedia = 0.0
        var valoreMedia = 0.0
        let lunghezzaArray = Double(array.count)

        //Media
        for valore in array {
            sommaValoriMedia = sommaValoriMedia + valore;
        }
        valoreMedia = sommaValoriMedia/lunghezzaArray
        return (valoreMedia)
    }
//Funzione atta a calcolare la varianza
    func calcoloVarianza(array: [Double]) -> Double {
        var doppio = [Double]()
        var varianza : Double
        var quadratoMedia : Double
        for valoreDoppio in array {
            doppio.append(valoreDoppio*valoreDoppio)
        }
        let media = self.calcoloMedia(array: array)
        let mediaQuadrati = self.calcoloMedia(array: doppio)
        quadratoMedia = media*media
        varianza = mediaQuadrati - quadratoMedia
        return varianza
    }
```

Tali funzioni vengono richiamare dalla funzione update⁶ che viene attivata da un timer con un intervallo costante di 5 secondi.

⁶Funzione che passa l'array della magnitude di ogni sensore per il calcolo delle feature

```
timer = Timer.scheduledTimer(timeInterval: 5.0, target: self, selector:  
    #selector(update), userInfo: nil, repeats: true)}
```

3.2.4 Creazione Dataset

La creazione del dataset è stata un'operazione "esterna" all'implementazione dell'app effettuata con il linguaggio Python. Essa ha come scopo quello di trasformare i set di dati dei vari sensori in un unico dataset che rispondesse alle esigenze della mia applicazione.

```
import csv  
import math  
  
magnitudeAccelerometerLinear = []  
magnitudeAccelerometerTerra = []  
magnitudeGyroscope = []  
magnitudeMagnetometer = []  
i = 0  
  
with  
    open('/Users/driverBehaviorDataset-master/data/17/aceleracaoLinear_terra.csv',  
        'r') as csv_file:  
        csv_reader = csv.DictReader(csv_file)  
        for line in csv_reader:  
  
            magnitudeAccelerometerLinear.append(math.sqrt(math.pow(float(line['x']),  
2)+ math.pow(float(line['y']), 2) + math.pow(float(line['z']), 2)))
```

La prima parte del codice consiste nell'importare i pacchetti csv e math: il primo consente di aggiungere al codice le classi che permettono la lettura e la scrittura di un file .csv, il secondo fornisce l'accesso alle funzioni matematiche. Tramite with open letto il primo file

”aceleracaoLinear_terra.csv” e attraverso la classe DictReader viene creato un oggetto che funziona come un normale lettore ma mappa le informazioni lette in un dict⁷ le cui chiavi sono date dal parametro fieldnames che può opzionale come in questo caso. Infine per ogni riga del file viene calcolata la magnitude che viene aggiunta in un array. Queste operazioni vengono ripetute per tutti i file da analizzare creando ogni volta un nuovo lettore.

L’ultima parte del codice consiste nella scrittura delle magnitude in un nuovo file .csv.

```
with open('new_names.csv', 'w') as new_file:
    fieldname = ['magnitudeAccelerometerLinear',
                'magnitudeAccelerometerTerra', 'magnitudeGyroscope',
                'magnitudeMagnetometer', 'safe/unsafe']
    csv_writer = csv.DictWriter(new_file, fieldnames=fieldname,
                                delimiter='\t')
    csv_writer.writeheader()
    while i < len(magnitudeAccelerometerTerra):
        magAccTerra = magnitudeAccelerometerTerra[i]
        magAccLinear = magnitudeAccelerometerLinear[i]
        magGyro = magnitudeGyroscope[i]
        magMgn = magnitudeMagnetometer[i]
        csv_writer.writerow({'magnitudeAccelerometerLinear':
magAccLinear, 'magnitudeAccelerometerTerra': magAccTerra,
'magnitudeGyroscope': magGyro, 'magnitudeMagnetometer': magMgn,
'safe/unsafe': True})
        i = i + 1
```

Il funzionamento è analogo a quello prima descritto del lettore. Viene creato un nuovo file .csv dal nome ”new_names.csv” al quale vengono assegnate le chiavi (fieldnames) per il dizionario. Quest’ultimo viene creato tramite la classe DictWriter. Successivamente

⁷Struttura dizionario

viene fatta un'iterazione della lunghezza pari al numero di righe del file letto, che assegna per ogni variabile rappresentante la magnitudine di un sensore il valore dell'array di magnitudine corrispondente. Ogni variabile sarà scritta nel file nella colonna d'appartenenza con il metodo `writerow({'nomeColonna': valoreDaAssegnare})`.

3.2.5 Creazione modello di predizione

Creato il dataset finale e avendo applicato su quest'ultimo gli algoritmi di classificazione tramite il software Weka, si è arrivati alla conclusione, dopo aver confrontato confusion matrix e accuracy, che l'algoritmo che meglio rispondeva alle esigenze del sistema è il Random Forest. Ora ci focalizziamo sulla creazione del modello e nell'integrazione di quest'ultimo nell'app.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn import metrics
import coremltools
# Load data
# Names of the columns
names = ['magnitudeAccelerometerLinear', 'magnitudeAccelerometerTerra',
         'magnitudeGyroscope', 'magnitudeMagnetometer', 'safe/unsafe']
# Read the CSV file
data = pd.read_csv('Cartel4_2_no-header_.csv', names=names, header=None)
```

La prima parte del codice è caratterizzata dall'importazione delle funzionalità offerte dalle librerie prima descritte. Pandas viene utilizzata per caricare il modello, Scikit-learn importa il modello che useremo per fare previsioni (RandomForest) e alcune funzionalità per la convalida incrociata e infine coremltools viene utilizzata per la conversione di un

modello da scikit-learn in `mlmodel`. Pandas ci offre la possibilità di leggere i dati da CSV, SQL, JSON e altri formati, questa operazione viene effettuata attraverso il metodo `read_csv('percorsoFile', 'nomeColonne', 'headers8=Yes/None')`.

Nella seconda parte vengono separati i dati di input dalla classe da predire e poiché il set di dati non ha valori mancanti si procede con la creazione del modello. Successivamente, adattiamo il modello con i dati (X, y) . Infine, convertiamo il modello nel formato Core ML specificando i nomi delle colonne di input che utilizzeremo nel codice Swift.

```
# Train a model
X = data[['magnitudeAccelerometerLinear', 'magnitudeAccelerometerTerra',
         'magnitudeGyroscope', 'magnitudeMagnetometer']]
y = data['safe/unsafe'].astype(str)
# Create the model
model = RandomForestClassifier()
# Fit the data
model.fit(X, y)
# Convert model to Core ML
coreml_model = coremltools.converters.sklearn.convert(model,
              input_features=['magnitudeAccelerometerLinear', 'magnitudeAccelerometerTerra',
                              'magnitudeGyroscope', 'magnitudeMagnetometer'])
# Save Core ML Model
coreml_model.save('modelWeka.mlmodel')
print('Core ML Model saved')
```

Un'ulteriore operazione è stata quella di calcolare l'accuratezza del modello usando la validazione incrociata (con 10 pieghe).

```
# Evaluate the model with cross validation
scores = cross_val_score(model, X, y, cv=10)
print('Scores: {}'.format(scores))
```

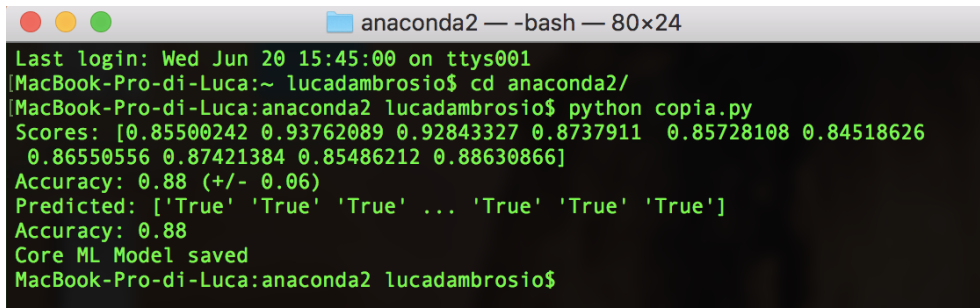
⁸Indica la presenza o meno dell'intestazione nel file

```

print('Accuracy: {0:0.2f} (+/- {1:0.2f})').format(scores.mean(), scores.std()
    * 2)
predicted = cross_val_predict(model, X, y, cv=10)
print('Predicted: {}'.format(predicted))
accuracy_score = metrics.accuracy_score(y, predicted)
print('Accuracy: {0:0.2f}').format(accuracy_score)

```

Tale codice ha generato i seguenti valori di accuracy:



```

Last login: Wed Jun 20 15:45:00 on ttys001
MacBook-Pro-di-Luca:~ lucadambrosio$ cd anaconda2/
MacBook-Pro-di-Luca:anaconda2 lucadambrosio$ python copia.py
Scores: [0.85500242 0.93762089 0.92843327 0.8737911 0.85728108 0.84518626
0.86550556 0.87421384 0.85486212 0.88630866]
Accuracy: 0.88 (+/- 0.06)
Predicted: ['True' 'True' 'True' ... 'True' 'True' 'True']
Accuracy: 0.88
Core ML Model saved
MacBook-Pro-di-Luca:anaconda2 lucadambrosio$

```

Figura 3.8: Risultati cross-validation modello di classificazione

Dopo aver generato il modello di classificazione, esso viene integrato nel codice Swift nella classe ViewController.

```

import CoreML
class ViewController: UIViewController {
    let model = modelWeka()
}

```

Per predire le manovre durante la guida nella classe ViewController viene implementata la funzione predictBehaviour all'interno della quale si applica il metodo "prediction" che viene utilizzato per rilevare il comportamento in base ai dati prodotti dallo smartphone.

```

//Funzione di predizione comportamento guidatore
func predictBehaviour() {
    while i < magnitudeMagnetometer.count

```

```
{
    mgnAccelerometer = magnitudeAccelerometer[i]
    mgnGyroscope = magnitudeGyroscope[i]
    mgnMagnetometer = magnitudeMagnetometer[i]

    if let prediction = try?
model.prediction(magnitudeAccelerometerLinear: Double(mgnAccelerometer),
magnitudeAccelerometerTerra: Double(mgnAccelerometer),
magnitudeGyroscope: Double(mgnGyroscope), magnitudeMagnetometer:
Double(mgnMagnetometer))
    {
        print("Descrizione", prediction.classLabel.description)
        print(prediction.classProbability[(prediction.classLabel)])
        arrayPredict.append(prediction.classLabel)
        print(arrayPredict)
    }
}
}
```

3.2.6 Affdex utilizzo e implementazione

Dopo aver rilevato la situazione pericolosa con i dati sensoriali, la classe ViewController passa la gestione alla classe moduleAffectiva che si occupa del riconoscimento, tramite il rilevamento del volto, di sonnolenza o distrazione.

Per utilizzare Emotion sdk bisogna aver installato CocoaPods sul Mac e creare un file chiamato "Podfile" nella directory del progetto. Il Podfile è un file di testo semplice che descrive le dipendenze del framework e della libreria contenuta nel progetto. In questo file vengono scritte le righe di codice che vengono fornite da Affectiva. Fatto ciò si procede con l'installazione del file pod che caricherà e configurerà il framework AffDex SDK

da utilizzare nel progetto.

Il codice seguente mostra come, dopo aver importato i moduli Affectiva (Affdex), viene effettuata la dichiarazione di un rilevatore del viso (detector), la sua creazione, configurando il rilevamento di espressioni e emozioni e infine il suo avvio.

```
import Affdex
class moduleAffectiva: UIViewController, AFDXDetectorDelegate {
    var detector : AFDXDetector? = nil
    override func viewDidLoad() {
        super.viewDidLoad()
        // create the detector
        detector = AFDXDetector(delegate:self, using:AFDX_CAMERA_FRONT,
maximumFaces:1)
        detector?.setDetectAllEmotions(true)
        detector?.setDetectAllAppearances(true)
        detector?.setDetectAllExpressions(true)
        detector!.start()
    }
}
```

Dopo aver creato un rilevatore del volto con la funzione detector vengono catturati i dati che potrebbero rilevare una situazione di sonnolenza o di distrazione. Tali dati vengono rappresentati dai parametri: chiusura degli occhi, apertura della bocca e attenzione. Tali dati vengono espressi tramite un numero Float compreso tra 0 (assente) e 100 (presente).

```
func detector(_ detector: AFDXDetector?, hasResults faces:
NSMutableDictionary?, for image: UIImage?, atTime time: TimeInterval) {
    if faces == nil {
        imageView?.image = image
    } else {
        let a = faces?.allValues
        if (a?.count ?? 0) > 0 {
```

```
        let face = a?[0] as? AFDXFace
        let eye: CGFloat? = face?.expressions.eyeClosure
        let attention: CGFloat? = face?.expressions.attention
        let mouth: CGFloat? = face?.expressions.mouthOpen
    }
}
}
```

La fase di implementazione è stata caratterizzata da varie operazioni. Quelle essenziali, ovvero, quelle che garantiscono un rilevamento di stati di guida pericolosi, sono state descritte in precedenza.

Capitolo 4

Valutazioni e risultati finali

In questo capitolo verranno analizzati i risultati prodotti dal software Weka tramite l'applicazione dei vari algoritmi di classificazione sul dataset. Inoltre, è stata svolta un'analisi sul riconoscimento di sottogruppi di sensori per scoprire quale combinazione di questi ultimi sia in grado di garantire un accuracy maggiore. Prima di procedere con l'analisi dei risultati è utile descrivere gli algoritmi di classificazione utilizzati e il dataset su cui verranno applicati.

4.1 Dataset

Il dataset costruito come descritto nel paragrafo 3.2.4 è un dataset composto da 20675 records, le colonne sono rappresentate dai valori della magnitude dei sensori presi in esame in questa tesi di laurea e l'ultima colonna rappresenta la classe da predire. Come descritto nel sito dal quale è stato scaricato il dataset, ogni file corrisponde alla registrazione di dati sensoriali per un tempo di 13 minuti.

4.2 Algoritmi di classificazione

Gli algoritmi di classificazione, come detto in precedenza, sono quelli che hanno lo scopo di determinare se gli attributi di una certa istanza appartengono o meno ad una classe. In particolare, grazie al software Weka, in questo lavoro di tesi sono stati utilizzati i seguenti:

- **Random Forest:** è un classificatore basato su un approccio insiemistico che è composto da molti alberi di decisione e dà in uscita la classe che corrisponde al risultato delle classi degli alberi presi individualmente.

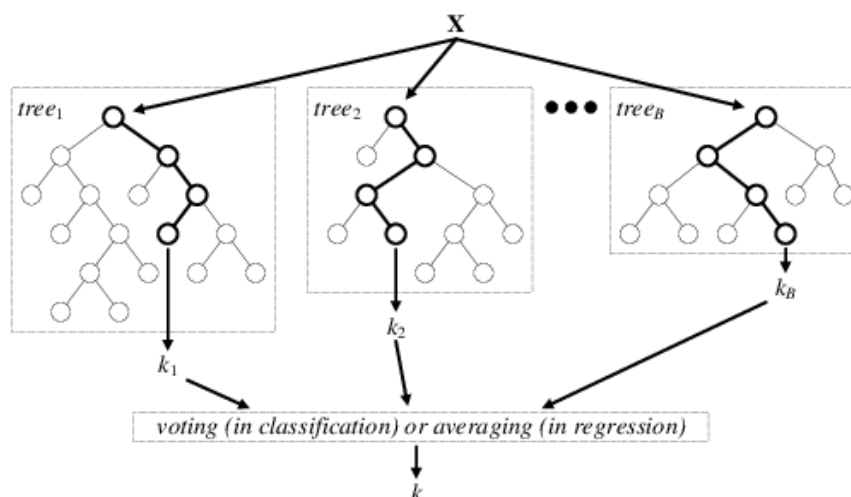


Figura 4.1: Funzionamento algoritmo RandomForest

- **Naive Bayes:** è una tecnica statistica con la quale si determina la probabilità di un elemento di appartenere a una certa classe. La tecnica si basa sul teorema di Bayes, dove si definisce la probabilità condizionata di un evento rispetto ad un altro.

- **Bayesian Network:** è un modello grafico probabilistico che rappresenta un insieme di variabili stocastiche con le loro dipendenze condizionali attraverso l'uso di un grafo aciclico diretto.
- **ZeroR:** prevede semplicemente la categoria di maggioranza (classe)
- **SMO:** risolve il problema di programmazione quadratica (QP) che si verifica durante l'addestramento di Support Vector Machines (SVM).

4.3 Analisi dei Risultati

In Weka per la valutazione dei classificatori vengono utilizzate: **Confusion Matrix** e **accuracy**.

La Confusion Matrix restituisce una rappresentazione dell'accuratezza di classificazione statistica [47]. Le colonne della matrice rappresentano le istanze che sono state classificate come appartenenti a quella classe. Le righe della matrice di confusione rappresentano le reali istanze che appartengono a quella classe. La matrice indica quindi il numero di casi che sono stati classificati correttamente e il numero di casi classificati in modo scorretto.

L'accuracy rappresenta l'accuratezza del modello e viene definita come il rapporto delle istanze classificate correttamente sul numero totale di istanze di testing considerate. I dati per calcolare l'accuracy vengono direttamente presi dalla Confusion Matrix della quale si vuole calcolare l'accuratezza.

In questo lavoro di tesi sono stati valutati inizialmente l'accuracy, il tempo di costruzione del modello e la confusion matrix del dataset completo, ovvero, non escludendo dalla classificazione nessun sensore. I risultati ottenuti vengono descritti nella seguente tabella.

Algoritmo di classificazione	Accuracy	Time
Bayesian Network	90.4474%	0.05 s
Naive Bayes	90.1669%	0.05 s
Random Forest	91.8162%	7.15 s
ZeroR	89.5333%	0.01 s
SMO	89.5333%	0.62 s

Tabella 4.1: Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset

Da tale tabella si evince che nonostante un tempo di costruzione maggiore l'algoritmo Random Forest ha un'accuratezza di riconoscimento più alta rispetto a quella degli altri algoritmi.

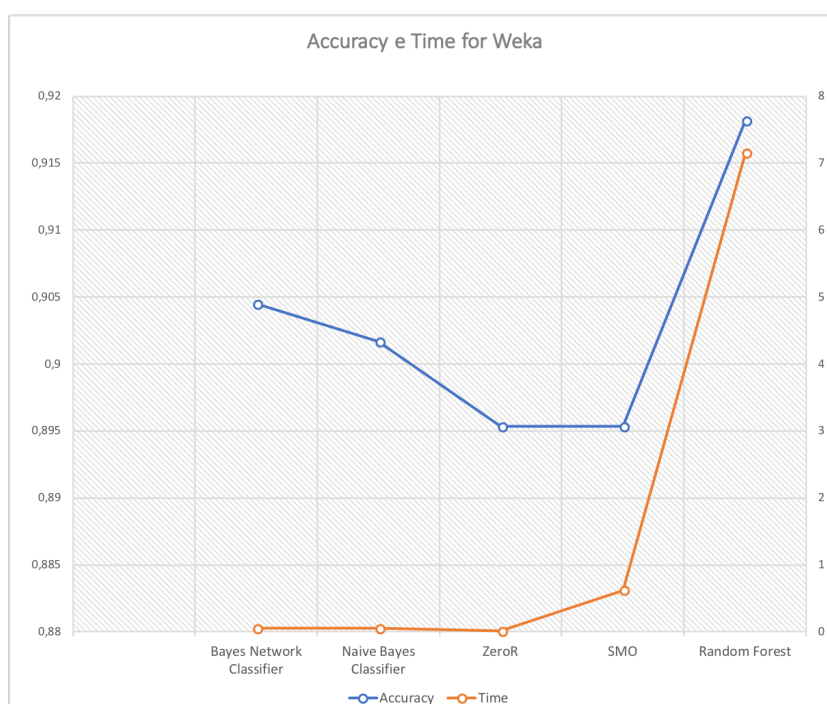


Figura 4.2: Grafico che mette a confronto l'accuracy e il tempo di costruzione per ogni algoritmo di classificazione

Tale risultato può essere anche visto attraverso il confronto delle diverse confusion matrix prodotte. Nelle figure successive vengono messe a confronto le confusion matrix tra gli

algoritmi che hanno un accuracy maggiore.

```

=== Confusion Matrix ===
      a      b  <-- classified as
18255  256 |      a = True
1436   728 |      b = False

=== Confusion Matrix ===
      a      b  <-- classified as
17951  560 |      a = True
1415   749 |      b = False

```

Figura 4.3: Confusion matrix dell'algoritmo Random Forest e Bayesian Network

Il numero di istanze classificate correttamente è dato dalla somma della diagonale, mentre le altre istanze sono classificate in modo errato. Ad esempio abbiamo 256 istanze della classe a erroneamente classificate come appartenenti alla classe b e 1436 istanze di b sono erroneamente classificate come appartenenti alla classe a.

Altri parametri di valutazione che vengono forniti dalla cross-validation permettono di valutare nel dettaglio l'accuracy della classe, tra questi vi sono:

- **True Positive rate (TP rate) o Recall:** è la frazione di esempi classificati come appartenenti alla classe x, fra tutti quelli che sono realmente della classe x. Nella confusion matrix è l'elemento diagonale diviso per la somma degli elementi della riga
- **False Positive rate (FP rate):** è la frazione di esempi classificati come appartenenti alla classe x, ma che in realtà appartengono a un'altra classe, fra tutti quelli che non appartengono alla classe x. Nella matrice di confusione viene dato dalla somma della colonna meno l'elemento diagonale diviso per la somma delle righe nelle altre classi.
- **Precision:** è la frazione di esempi realmente di classe x fra tutti quelli classificati come x. Nella confusion matrix è l'elemento diagonale diviso per la somma delle colonne rilevanti.

- **F-Measure:** è una misura che riassume precision e recall e viene calcolato con la seguente formula $F - Measure = 2 * Precision * Recall / (Precision + Recall)$.

```

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                0,986   0,664   0,927     0,986   0,956     0,464   0,846   0,973   True
                0,336   0,014   0,740     0,336   0,463     0,464   0,846   0,552   False
Weighted Avg.   0,918   0,596   0,907     0,918   0,904     0,464   0,846   0,929

```

Figura 4.4: Accuracy nel dettaglio dell'algoritmo Random Forest

4.3.1 Analisi risultati sottogruppi di sensori

In questa sezione vengono descritte le tabelle che mostrano i risultati ottenuti dagli algoritmi di classificazione sul dataset, analizzando, in questo caso, solo i sottogruppi di sensori presenti nel dataset.

Algoritmo di classificazione	Accuracy	Time
Bayesian Network	90.4667%	0.04 s
Naive Bayes	90.2733%	0.02 s
Random Forest	91.6759%	5.04 s

Tabella 4.2: Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati del giroscopio

Algoritmo di classificazione	Accuracy	Time
Bayesian Network	90.3071%	0.06 s
Naive Bayes	89.9734%	0.02 s
Random Forest	90.6505%	5.15 s

Tabella 4.3: Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati dell'accelerometro Lineare

Algoritmo di classificazione	Accuracy	Time
Bayesian Network	90.578%	0.06 s
Naive Bayes	90.4716%	0.02 s
Random Forest	91.3761%	4.91 s

Tabella 4.4: Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati dell'accelerometro

Algoritmo di classificazione	Accuracy	Time
Bayesian Network	90.3507%	0.05 s
Naive Bayes	90.2056%	0.02 s
Random Forest	90.8779%	6.25 s

Tabella 4.5: Tabella dei risultati di accuracy e tempo di costruzione degli algoritmi di classificazione applicati sul dataset con l'esclusione dei dati del magnetometro

I risultati prodotti dall'applicazione degli algoritmi di classificazione sui sottogruppi di sensori, evidenzia come, nonostante l'abbassamento del tempo di costruzione del modello¹, l'algoritmo più accurato resta sempre il RandomForest. Inoltre, si può notare come con l'esclusione dall'analisi dei dati del giroscopio il modello sia più accurato rispetto agli altri sottogruppi, ma non rispetto al dataset iniziale.

Per tali motivi si è deciso di costruire il modello di classificazione partendo dal dataset iniziale e non da sottogruppi di sensori.

¹Dovuto all'assenza di un sensore, quindi meno calcoli

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi di laurea è stata sviluppata un'applicazione che implementa i concetti ADAS ed è in grado di riconoscere il comportamento alla guida di un conducente attraverso l'analisi dei sensori inerziali e il riconoscimento delle espressioni/emozioni facciali. Questo lavoro oltre ad arricchire il campo delle applicazioni mobile nel settore dell'ADAS cerca di proporsi come applicazione innovativa nel settore in quanto non utilizza il GPS e cerca di limitare l'utilizzo della fotocamera, tutelando la privacy del guidatore e minimizzando il consumo d'energia dello smartphone durante la sessione di guida.

I risultati ottenuti con Weka hanno mostrato come il classificatore Random Forest è in grado di produrre modelli molto precisi, con un'accuratezza superiore al 90%. Ottimi risultati sono stati ottenuti anche grazie all'utilizzo dell'sdk Emotien offerto da Affectiva per quanto riguarda la rilevazione delle espressioni/emozioni facciali. Tale precisione è stata valutata attraverso una fase di testing personale.

Unica pecca del lavoro è stata l'impossibilità di creare il dataset di dati per la costruzione del modello di apprendimento con le rilevazioni effettuate dal mio smartphone e la mancanza di una fase di testing. Questi problemi, come spiegato nei capitoli precedenti, sono riconducibili entrambi alla mancanza di un'automobile e per l'assenza di guidatori

esperti in grado di svolgere particolari e pericolose manovre di guida.

Possibili sviluppi futuri possono riguardare proprio la risoluzione dei problemi prima esposti, attraverso un campionamento dei sensori inerziali dello smartphone durante una sessione di guida con lo scopo di costruire un dataset partendo dalle rilevazioni prodotte dal mio dispositivo e cercando, inoltre, di identificare un insieme più vasto di attività di guida pericolose.

Altre estensioni del progetto riguardano la possibilità di coinvolgere nell'attività di training, precisamente nell'estrazione dei dati sensoriali, un raggio più ampio di sensori dello smartphone,- ad esempio, può essere considerando l'utilizzo del microfono, che tramite la registrazione di rumori potrebbe identificare delle manovre di guida azzardate.

Per preservare al meglio il consumo della batteria, un possibile sviluppo potrebbe essere quello di attivare la detenzione dello stile di guida dopo che il veicolo supera una determinata velocità, prendendo spunto da [18]. Inoltre un'altra svolta potrebbe essere l'integrazione delle mappe e l'attivazione di queste solo quando il guidatore ne ha bisogno, tutelando così la sua privacy.

Un altro aspetto interessante riguarda la possibilità di sviluppare l'applicazione su un dispositivo Android, andando così a confrontare l'accuratezza tra i due sistemi operativi nel campo dell'applicazione sviluppata.

Infine, credo che tali sistemi saranno via via sostituiti da auto sempre più intelligenti che andranno ad essere dei sostituti eccellenti del conducente, basti pensare alle autovetture autonome progettate da Uber e Tesla, o ai sistemi ADAS implementati in auto di medio livello come Hyundai. Nonostante ciò, l'applicazione sviluppata, o in generale le applicazioni mobile che rientrano nel campo delle ADAS vanno a interessare un'ampia fascia di consumatori che posseggono un'auto priva di sistemi avanzati di assistenza alla guida ed hanno lo scopo d'assistere il conducente durante la guida cercando di rilevare attraverso

i dati dei sensori e le espressioni facciali situazioni di guida pericolose, utili per prevenire incidenti e garantire un maggiore comfort alla guida.

Bibliografia

- [1] https://en.wikipedia.org/wiki/Advanced_driver-assistance_systems
- [2] Smirnov A., Lashkov I.: State-of-the-art analysis of available advanced driver assistance systems. in 17th Conference of the Open Innovations Association FRUCT, 2015
- [3] <https://www.mobileye.com/>
- [4] <https://www.trwaftermarket.com/it/>
- [5] <https://www.autoliv.com/>
- [6] <https://www.subaru.it/tecnologia/eyesight>
- [7] <https://en.wikipedia.org/wiki/Sensor>
- [8] D. A. Johnson, M. M. Trivedi, Driving style recognition using a smartphone as a sensor platform in Intelligent Transportation Systems (ITSC), 2011
- [9] Nidhi Kalra, Divya Bansal, Analyzing Driver Behavior using Smartphone Sensors: A Survey in International Journal of Electronic and Electrical Engineering, Chandigarh, India, 2014

- [10] Smirnov A., Lashkov I., Kashevnik A., Human-Smartphone Interaction for Dangerous Situation Detection and Recommendation Generation While Driving in Springer International Publishing Switzerland, ITMO University, St. Petersburg, Russia, 2016
- [11] Vincenzo Antonio Manganaro, Tecniche di DM: Alberi di decisione ed algoritmi di classificazione, Palermo, Italia
- [12] https://en.wikipedia.org/wiki/Random_forest
- [13] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- [14] https://en.wikipedia.org/wiki/Bayesian_network
- [15] https://en.wikipedia.org/wiki/Sequential_minimal_optimization
- [16] C.-W. You et al., "CarSafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones" in Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services, 2013
- [17] iOnRoad. <http://www.ionroad.com/>
- [18] Luis M. Bergasa, Daniel Almería, Javier Almazàn, J. Javier Yebes, Roberto Arroyo, DriveSafe: an App for Alerting Inattentive Drivers and Scoring Driving Behaviors, on IEEE Intelligent Vehicles Symposium (IV), Dearborn, Michigan, USA, 2014
- [19] Eren, H.; Makinist, S.; Akin, E.; Yilmaz, A. Estimating driving behavior by a smartphone. Proceedings on IEEE Intelligent Vehicles Symposium (IV), 2012
- [20] Michele Ruta, Floriano Scioscia, Annarita Cinquepalmi, Silvia Cipriani, Eugenio Di Sciascio, Dai biosegnali agli stati emotivi: un approccio semantico, Dipartimento di Ingegneria Elettrica e dell'Informazione (DEI), Politecnico di Bari, Bari, Italia

- [21] https://en.wikipedia.org/wiki/Affective_computing
- [22] Picard R.W., *Affective computing*, MIT press, Cambridge, 1997
- [23] Taranpreet Singh Saini, Dr. MangeshBedekar, Saniya Zahoor, *Analysing Human Feelings by Affective Computing Survey*, in MAEER's MIT, Kothrud, Pune, India, 2017
- [24] https://en.wikipedia.org/wiki/Paul_Ekman
- [25] J. Kurami, R. Rajesh, KM. Pooja, in *Second International Symposium on Computer Vision and the Internet (VisionNet'15)*, India, 2015
- [26] Mostafa Mohammadpour, Seyyed Mohammad. R Hashemi, Hossein Khaliliardali, Mohammad. M AlyanNezhadi, *Facial Emotion Recognition using Deep Convolutional Networks*, in *4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)* Iran University of Science and Technology, Tehran, Iran, 2017
- [27] https://en.wikipedia.org/wiki/Convolutional_neural_network
- [28] P.Lucey, J.F.Cohn, T.Kanade, J.Saragih, Z.Ambadar, and I. Matthews, *The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression*, In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE Computer Society Conference on, pages 94-101, IEEE, 2010
- [29] <https://www.affectiva.com/product/affdex-for-market-research/>
- [30] <https://www.affectiva.com/product/affectiva-automotive-ai/>
- [31] Daniel McDuff, Rana El Kaliouby, Thibaud Senechal, May Amr, Jeffrey F. Cohn, Rosalind Picard, *Affectiva-MIT Facial Expression Dataset (AM-FED): Naturalistic*

and Spontaneous Facial Expressions Collected In-the-Wild, in IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013

[32] https://en.wikipedia.org/wiki/Deep_learning

[33] Daniel McDuff, Rana El Kaliouby, May Amr et al., AFFDEX SDK: A Cross-Platform Real-Time Multi-Face Expression Recognition Toolkit, San Jose, CA, USA, 2016

[34] Viola, P. and Jones, M. 2001. "Rapid object detection using a boosted cascade of simple features." PROC CVPR IEEE, 2001

[35] <https://github.com/jair-jr/driverBehaviorDataset>

[36] Jair Ferreira Júnior, Eduardo Carvalho, Bruno V. Ferreira, Cleidson de Souza, Yoshihiko Suhara, Alex Pentland, Gustavo Pessin, Driver behavior profiling: An investigation with different smartphone sensors and machine learning, 2017

[37] <https://en.wikipedia.org/wiki/IOS>

[38] <https://swift.org/>

[39] <https://cocoapods.org/>

[40] <https://developer.apple.com/documentation/coreml>

[41] <https://www.python.org/>

[42] <https://pandas.pydata.org/>

[43] <http://scikit-learn.org/stable/index.html>

[44] <https://pypi.org/project/coremltools/>

[45] <https://www.cs.waikato.ac.nz/ml/weka/downloading.html>

[46] <https://github.com/danielgindi/Charts>

[47] https://en.wikipedia.org/wiki/Confusion_matrix