Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

Riconoscimento automatico dell'uso dello smartphone durante la guida tramite sensori inerziali

Relatore:
Dott. Luca Bedogni

Presentata da:
Octavian Bujor

Sessione III
Anno Accademico 2016/2017

A Noemi

la più grande gioia e benedizione della nostra famiglia.

Introduzione

La guida distratta dovuta all'uso di dispositivi mobili contribuisce a superare 1,25 milioni di vittime a causa degli incidenti stradali a livello mondiale. Usare il cellulare alla guida aumenta di circa quattro volte il rischio di incidente perché riduce i tempi di reazione e rende difficile la tenuta della corsia e mantenere la distanza di sicurezza [1]. La distrazione più allarmante sono i messaggi, infatti leggere o scrivere un messaggio mentre si guida prende gli occhi dalla strada per 5 secondi, a una velocità di 90 km/h è come attraversare in lunghezza un intero campo da calcio con gli occhi chiusi [2]. La grande diffusione degli smartphone, che secondo *Statista* ha superato le 2 miliardi di unità nel 2016 [3], non fa altro che aumentare il numero delle vittime, infatti negli Stati Uniti nel 2016 c'è stato il record di pedoni morti a causa dello smartphone, usato sia dai guidatori che dai pedoni stessi [4]. Tanti enti cercano di risolvere questo problema, i governi tramite la legislazione e campagne di sensibilizzazione, le industrie automobilistiche e tecnologiche investono miliardi nei veicoli a guida autonoma e infine le compagnie assicurative stanno sviluppando sistemi per penalizzare i guidatori distratti. La compagnia Allstate, gigante statunitense delle assicurazioni, ha condotto uno studio su centinaia di migliaia di guidatori, analizzando oltre 160 milioni di viaggi che ha confermato l'ipotesi di partenza: utilizzare il telefono al volante aumenta la probabilità di avere incidenti stradali. L'obiettivo è far pagare di più l'assicurazione agli automobilisti indisciplinati e premiare con costi inferiori quelli che rispettano le regole e non toccano il telefono mentre guidano [5]. Esistono delle applicazioni che disattivano le notifiche e attivano

la modalità silenziosa quando l'utente imposta lo stato di guidatore, ma gli esperimenti hanno dimostrato che gli utenti non sono molto disponibili a farlo manualmente. Ci sono diversi metodi per riconoscere automaticamente se un dispositivo si trova all'interno di un veicolo, è più difficile invece determinare se lo smartphone viene usato dal guidatore oppure da un passeggero che può continuare a usarlo senza essere un pericolo per gli altri. In letteratura è stata pubblicata [6] la proposta di un metodo di riconoscimento del guidatore in uso dello smartphone sfruttando la differenza dell'accelerazione centripeta nelle curve. Tale metodo prevede l'utilizzo dello smartphone e di un punto di riferimento che potrebbe essere un adattatore per accendisigari, la porta ODB-II o lo smartphone di un passeggero. Il problema è che la maggioranza non è in possesso del punto di riferimento perché guidano da soli in auto oppure non hanno il desiderio di acquistare appositi apparecchi.

In questo lavoro di tesi viene studiata la fattibilità di un nuovo metodo per riconoscere automaticamente l'uso dello smartphone alla guida basato sull'apprendimento automatico implementato con le reti neurali artificiali. L'approccio proposto è in grado di funzionare sia con due dispositivi come fatto in [6] che con un unico dispositivo. I risultati sono soddisfacenti in quanto con due dispositivi dopo una curva si supera la soglia di 82% di riconoscimenti corretti rispetto agli 80% dello stato dell'arte. Considerando tre o cinque curve consecutive dipende dalla scelta delle soglie di affidabilità ma tipicamente i risultati dello stato dell'arte superano di qualche punto le percentuali del nuovo metodo. Cambia lo scenario quando viene usato un unico dispositivo essendo un problema più difficile, l'obbiettivo è di fare il riconoscimento anche dopo curve multiple infatti si è ottenuto che con una curva si arriva da 68,8% a 80% di riconoscimenti corretti a seconda delle soglie scelte, per arrivare al 80% bisogna valutare almeno da tre a sette curve consecutive. Purtroppo non essendoci altre ricerche che fanno il riconoscimento con un unico dispositivo questi ultimi risultati non possono essere confrontati ma sono sicuramente positivi in quando mostrano la fattibilità di questo nuovo approccio.

Indice

Elenco delle figure

Capitolo 1

Background

1.1 Smartphone - un piattaforma di sensori

Il sensore è un dispositivo di input che misura una quantità fisica in diretta interazione con il sistema misurato. I suoi valori grezzi vengono trasformati per avere informazioni sull'ambiente e utenti. Dato il loro prezzo contenuto vengono integrati negli smartphone moderni che sono equipaggiati di tantissimi input che possono essere usati per la ricerca. Gli smartphone sono diventati dispositivi potenti, relativamente economici e versatili per la collezione di dati. La loro diffusione che secondo le stime di *Statista* [3] nel 2016 si è superato il limite di 2 miliardi di utenti smartphone nel mondo li rende accessibili sia al pubblico generale che accademico.

Alcuni dei sensori usati per la raccolta di dati sono:

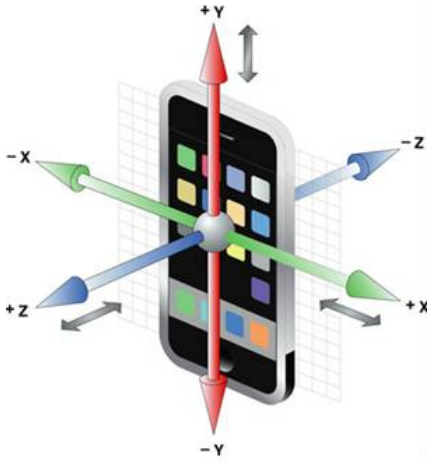
- Fotocamera (anche più di una)
- Microfono (anche più di uno)
- Accelerometro a 3 assi
- Giroscopio a 3 assi
- Magnetometro

- Sensore di prossimità
- Sensore di illuminazione ambientale
- Sensore di riconoscimento del tocco
- Sensore di impronte
- GPS

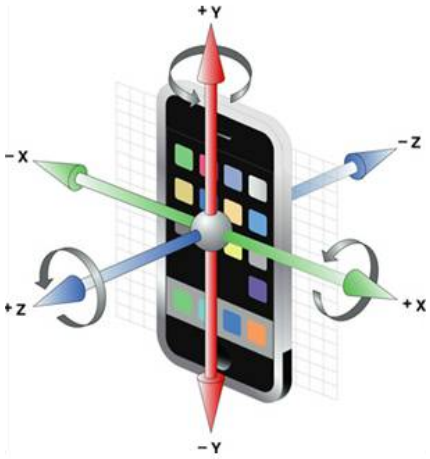
Per le ricerche spesso vengono aggregati i dati di più sensori, ad esempio nel sistema MIROAD (A Mobile-Sensor-Platform for Intelligent Recognition Of Aggressive Driving) [7] vengono usati: la fotocamera, l'accelerometro, il giroscopio, il magnetometro e il GPS per il riconoscimento dello stile di guida. Gli autori di [8] hanno usato lo smartphone insieme ai dispositivi indossabili per riconoscere meglio i movimenti aggressivi del guidatore e con una buona precisione l'angolo di rotazione del voltante. In questa ricerca di tesi sono stati usati i seguenti sensori: accelerometro, giroscopio e magnetometro che vengono descritti nelle sezioni successive.

1.1.1 Accelerometro

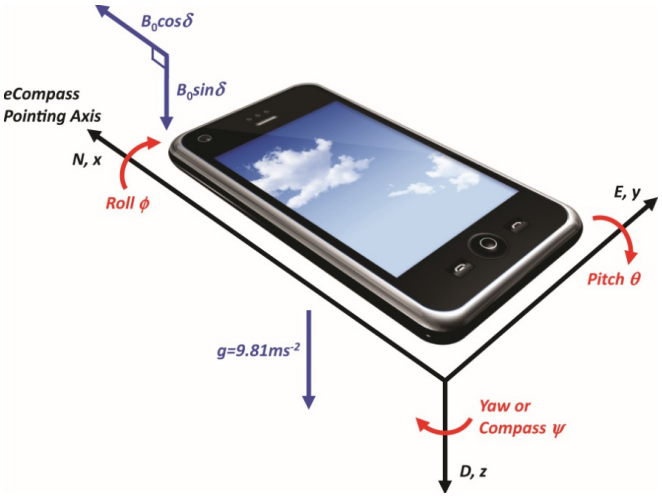
L'accelerometro è un sensore di movimento che misura l'accelerazione subita dal dispositivo nelle tre direzioni (assi): x, y e z nell'unità di misura m/s^2. Ha un prezzo molto ridotto ed è presente nella maggioranza degli smartphone, secondo *GSMArena* [9] il 98% degli smartphone prodotti dal 2010 in poi ne possiede uno. L'accelerometro lineare è un sensore software basato su quello hardware con la differenza che il suo valore non include la forza di gravità, la sua disponibilità varia a seconda del sistema operativo. Nella figura 1.1a viene mostrata la posizione e l'orientamento degli assi rispetto allo smartphone.



(a) Assi accelerometro



(b) Assi giroscopio



(c) Assi magnetometro

Figura 1.1: Assi dei sensori smartphone

1.1.2 Giroscopio

Il giroscopio è un sensore di movimento che misura la velocità angolare in rad/s dei suoi tre punti di riferimento: x, y e z. Nonostante il suo prezzo ridotto non è così diffuso come l'accelerometro infatti secondo *GSMArena* [9] soltanto il 27% degli smartphone prodotti dal 2010 in poi ne possiede uno. Se consideriamo l'anno 2017 il giroscopio è presente nei 46% dei dispo-

sitivi segnando una tendenza di crescita. Negli smartphone lavora insieme all'accelerometro per rilevare i movimenti in maniera più precisa, spesso viene utilizzato nei videogiochi per migliorare l'esperienza utente. Nella figura 1.1b viene mostrata la posizione e la rotazione degli assi rispetto allo smartphone.

1.1.3 Magnetometro

Il Magnetometro è un sensore di posizione che misura l'intensità e la direzione dei campi magnetici misurati in μT comprese le variazioni nel tempo e nello spazio: x, y e z. Viene comunemente usato nelle applicazioni di bussola, mappe e nei navigatori in cui lavora insieme al GPS per indicare la direzione precisa. E' più diffuso del giroscopio ma meno dell'accelerometro infatti secondo *GSMArena* [9] è presente sui 54% degli smartphone prodotti dal 2010 in poi.

1.2 Riconoscimento della modalità di trasporto

Le tecniche di riconoscimento della modalità di trasporto cercano di riconoscere automaticamente il veicolo usato dal utente (e.g. auto, treno, bicicletta) analizzando i dati dei sensori degli smartphone senza nessun feedback umano. La conoscenza dell'attuale modalità di trasporto costituisce un'informazione preziosa per le applicazioni in diversi domini. Da una parte si possono fare delle applicazioni che offrono delle funzionalità in base al contesto come ad esempio abilitare lo smartphone a riconfigurarsi da solo quando viene rilevata una certa modalità di trasporto. Dall'altra parte quando gli utenti condividono i propri dati sul contesto si possono creare dei sondaggi aggregati che danno informazioni sulla qualità della vita dell'individuo e della collettività.

Il problema del riconoscimento della modalità di trasporto può essere visto come una sottoclasse dei problemi di classificazione delle attività e quindi

possono essere applicate le stesse metodologie di studio per ottenere buoni risultati. Per la classificazione spesso vengono usate tecniche di Machine Learning (ML) in particolare l'apprendimento supervisionato. Per implementare questi sistemi sullo smartphone ci sono problemi di risorse computazionali e batteria limitata.

Bedogni *et al.* in [10] presentano un sistema di riconoscimento della modalità di trasporto efficiente che può essere usato negli smartphone odierni. Si tratta di un sistema innovativo in quanto non usa il GPS (che utilizza tanta energia), si adatta dinamicamente alle capacità del dispositivo scegliendo il miglior trade-off tra energia e accuratezza ed è decentralizzato senza la necessità di addestramento collaborativo. Questo sistema è in grado di riconoscere le seguenti modalità di trasporto: stare fermo, camminata, auto, treno, bicicletta, autobus urbani e bus da viaggio con un'accuratezza tra 80% e 90% a seconda dell'algoritmo applicato.

Gli autori di [10] hanno sviluppato un framework per i sistemi Android che riconosce la modalità di trasporto chiamato WAID (What Am I Doing). WAID può funzionare sia come applicazione che come servizio e supporta due modalità di utilizzo: Addestramento e Previsione. Nella fase di addestramento l'utente specifica la sua corrente modalità di trasporto e l'applicazione inizia a registrare i dati dell'accelerometro e del giroscopio. Per ogni finestra temporale dai dati grezzi viene fatta l'estrazione delle seguenti caratteristiche: *minimo*, *massimo*, *media* e *deviazione standard* che vengono memorizzate. Alla fine della fase di acquisizione sono creati i modelli di classificazione che sono a disposizione del modulo di previsione. La modalità di previsione legge i dati dai sensori e li trasforma in caratteristiche e produce in output la previsione della modalità di trasporto utilizzata in quel istante. I valori previsti sono anche resi disponibili tramite un Content Provider a applicazioni di terze parti che possono cambiare il loro funzionamento in base al contesto rilevato.

1.3 Machine learning

Machine Learning (ML) o l'apprendimento automatico rappresenta un insieme di metodi che forniscono ai computer l'abilità di apprendere senza essere stati esplicitamente programmati. Ci sono molte definizione, la più citata è quella di Tom M. Mitchell [13]:

Definizione 1.3.1. Si dice che un programma apprende dall'esperienza E con riferimento ad alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E.

L'apprendimento automatico è un argomento molto vasto, successivamente vengono descritte solo le parti più rilevanti per questo lavoro di tesi. I suoi compiti vengono suddivisi in due ampie categorie: apprendimento supervisionato e apprendimento non supervisionato a seconda dei dati disponibili.

Apprendimento supervisionato è una tecnica di ML che ha il compito di imparare una funzione che associa l'input all'output basandosi sulle coppie input-output di esempio fornite. Ogni esempio consiste dell'input che solitamente è un vettore di valori e un valore desiderato di output. Analizzando gli esempi forniti l'algoritmo di apprendimento supervisionato inferisce una funzione che può essere applicata a nuovi esempi.

Apprendimento non supervisionato è una tecnica di ML che ha il compito di inferire una funzione che descrive la struttura dei dati forniti. A differenza dell'apprendimento supervisionato non viene fornito l'output desiderato ma solo l'input. Quindi il suo compito consiste nel trovare similarità o gruppi intrinsechi nei dati.

1.3.1 Classificazione

Una delle applicazione del ML è il problema della classificazione che consiste nell'identificare a quale categoria appartengono i nuovi esempi a partire dagli esempi fortini all'apprendimento di cui si conosce la categoria di appartenenza. Il problema così definito viene considerato un'istanza dell'apprendimento supervisionato perché le classi di appartenenza sono conosciute al momento dell'apprendimento.

I problemi di classificazione si suddividono principalmente in tre categorie diverse in base alle caratteristiche delle classi.

Classificazione binaria compito di classificazione in cui le classi sono due, un esempio: determinare se una persona si trova alla guida oppure no. L'output è un valore scalare, solitamente binario.

Classificazione multiclasse compito di classificazione in cui le classi sono più di due ma comunque un numero finito e predefinito con il requisito che ogni istanza appartiene ad una sola classe, un esempio: determinare quale mezzo di trasporto utilizza una persona: auto, treno, bici, autobus. L'output è un valore scalare, intero che identifica una classe unicamente.

Classificazione multi-etichetta compito di classificazione simile alla multiclasse con la differenza che ogni istanza può appartenere a una o più classi, un esempio: data un'immagine determinare quali sono i mezzi di trasporto presenti. L'output è un vettore di valori binari, ogni elemento corrisponde a una classe.

1.3.2 Reti neurali artificiali

Le reti neurali artificiali (ANN) sono sistemi ispirati alle reti neurali biologiche che compongono il cervello degli animali. La rete è composta da nodi (neuroni) interconnessi tra di loro che tipicamente sono organizzati in livelli. Le ANN sono molto usate in letteratura per la loro possibilità di imparare

pattern molto complessi e negli ultimi anni c'è stata una crescita nel loro interesse anche da parte delle aziende che hanno sviluppato diversi servizi e applicazioni anche per smartphone.

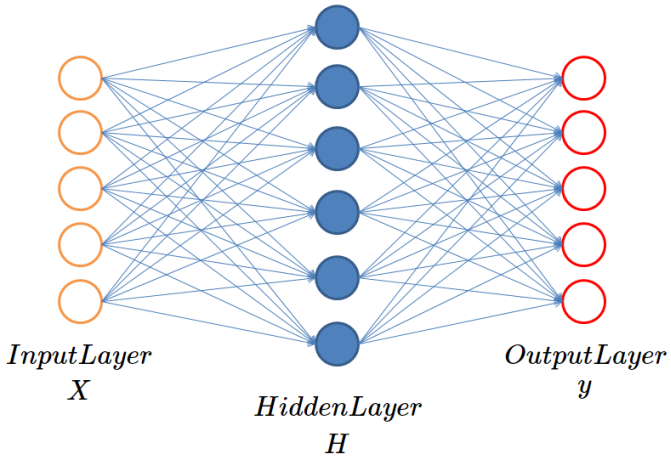


Figura 1.2: Struttura di una rete neurale artificiale

La struttura di una ANN è composta da tre tipi di livelli (visualizzata nella figura 1.2):

Livello di input un unico livello che rappresenta l'input, contiene un numero di neuroni che è uguale alla cardinalità dell'input ovvero al numero delle caratteristiche delle istanze.

Livello nascosto è un livello opzionale il suo scopo è di creare caratteristiche più complicate per migliorare il risultato. Nella rete possono esserci anche più di uno, in questo caso la rete viene definita *deep*.

Livello di output un unico livello che rappresenta l'output, il numero di neuroni contenenti dipende dal problema, ad esempio per la classificazione binaria è sufficiente anche un solo neurone.

La struttura della ANN cambia a seconda dal problema che si vuole risolvere, è qui che si trova la difficoltà del loro utilizzo, il progettista deve

capire quanti livelli e quanti neuroni usare insieme ad altri parametri di configurazione per ottenere un buon risultato che riesca a prevedere dei valori soddisfacenti per i nuovi esempi.

1.3.3 Dataset

I dati sono fondamentali per la maggioranza dei problemi di apprendimento automatico. Nel caso di apprendimento supervisionato la struttura e la qualità dei dati contribuiscono indirettamente al successo del processo. Tipicamente il dataset viene suddiviso in due parti: dati di addestramento e dati di test. Ognuna delle parti viene suddivisa in due insiemi: X input e Y output. L'idea alla base del funzionamento dell'apprendimento supervisionato come menzionato precedentemente è di addestrare un modello con i dati di esempio X e Y che sia in grado di inferire una funzione che prende in input X' mai visto prima restituendo Y'. Ogni istanza dell'input è un vettore di caratteristiche tipicamente numeriche.

Caratteristica

La caratteristica (feature) è una proprietà individuale e misurabile di un fenomeno osservato, nella classificazione la scelta delle caratteristiche discriminanti è un passo fondamentale per ottenere un risultato soddisfacente. Il numero delle caratteristiche ha un grande impatto sull'efficienza dell'algoritmo. A partire dai dati grezzi che rappresentano le caratteristiche iniziali viene fatta la selezione di quelle più significative per ridurre l'input che può essere ridondante e troppo vasto. La selezione delle caratteristiche può essere fatta sia manualmente che in maniera algoritmica a seconda del numero delle caratteristiche.

1.3.4 Valutazione e validazione

Un modello dopo essere addestrato viene valutato per misurare la sua capacità di previsione viene usate spesso l'accuratezza e la funzione di perdita

(loss). La valutazione sui dati di addestramento ha relativamente poco senso perché sono dati che il modello ha già visto ma comunque può essere presa in considerazione per capire quanto ha imparato dagli esempi. Interessa di più invece la valutazione sui dati di test, cioè quei dati che il modello non ha mai visto in precedenza e se ottiene un buon risultato vuol dire che ha imparato bene dagli esempi fortini nella fase di addestramento.

Una caratteristica importante di un modello è la capacità di generalizzazione, se riesce ad ottenere un buon risultato sui dati di addestramento non è detto che riesca a generalizzare bene perché il modello potrebbe anche memorizzare troppo gli esempi e non riuscire a generalizzare sui dati mai visti. Anche i dati di test non sono sufficienti a questo scopo perché la scelta di questi dati potrebbe essere fatta in maniera più o meno favorevole alla generalizzazione perché comunque quei dati potrebbero non rappresentare tutto il dataset.

Una modalità per valutare la capacità di generalizzazione del modello è la cross-validazione, ne esistono diversi tipi di seguito verrà descritta la *K-fold cross validation* perché usata in questo lavoro di tesi. La K-fold CV consiste nel suddividere il dataset in k gruppi o folders all'incirca della stessa dimensione. La procedura viene ripetuta k e ad ogni iterazione viene scelto un folder differente che verrà usato come validazione e gli altri $k-1$ folders per l'addestramento. In fine viene fatta la media dei risultati delle k iterazioni. In questo modo tutti i dati sono usati sia per l'addestramento che per la valutazione una porzione alla volta perciò il risultato della procedura rappresenta tutto il dataset. Nella figura 1.3 viene mostrato lo schema del processo.

1.3.5 Tensorflow

TensorFlow è un framework open source per il machine learning, creato e sviluppato dal Google Brain Team, definito pronto per la produzione e usato da tante aziende tra cui anche la stessa Alphabet. Più in generale è una libreria per il calcolo numerico che usa i grafi per il flusso di dati, in cui i nodi

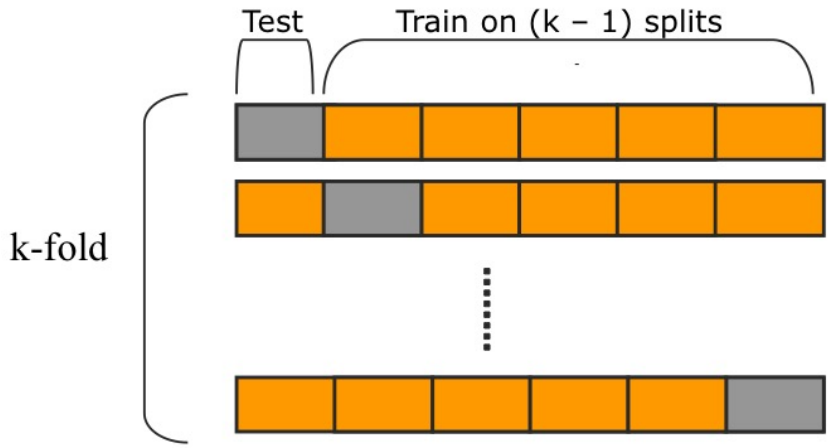


Figura 1.3: K-fold cross-validation

rappresentano operazioni matematiche e gli archi vettori multidimensionali di dati chiamati *tensors*. Questa architettura flessibile e scalabile permette di eseguire la computazione da una o più CPU, GPU, sul desktop, server e dispositivi mobili. Il grande vantaggio di questo framework rispetto agli altri è la sua possibilità di essere eseguito anche nei dispositivi mobili con la versione Lite. Nonostante sia una libreria generica, TensorFlow è stata progettata soprattutto per i modelli di reti neurali profonde [14].

Con TensorFlow lite si può usufruire dei modelli sui dispositivi mobili in pochi passi:

- Creare e addestrare un modello TensorFlow.
- Convertire il modello creato con TensorFlow Lite Converter.
- Caricare il modello convertito sul dispositivo mobile.
- Usare le API messe a disposizione dalla libreria per sviluppare applicazioni che usano il modello.

Nel 2016 Google presenta il Tensor processing unit (TPU) un circuito integrato (ASIC) costruito appositamente per l'apprendimento automatico e adattato a TensorFlow. Lo scopo dei TPU è di offrire una soluzione su misura

molto ottimizzata per quanto riguarda il rapporto prestazione e consumo energetico.

1.3.6 Keras

Keras è una libreria open source ad alto livello per le reti neurali, è stata sviluppata per permettere una sperimentazione veloce, portando le idee a risultati in meno tempo possibile [15]. Il suo punto forte è che riesce a creare i modelli usando altre librerie come TenserFlow, CNTK e Theano, permettendo di usare tutte le potenzialità di queste librerie senza dover imparare a usarle. Questa astrazione offre anche la possibilità di non essere legato al backend usato, senza cambiare il modello è possibile usarne uno piuttosto che un altro.

Capitolo 2

Stato dell'arte

2.1 Riconoscimento dello stile di guida

Per promuovere la sicurezza stradale e del conducente i ricercatori [11] hanno trovato che il guidatore si comporta in maniera relativamente più sicura quando viene monitorato e gli vengono segnalati i comportamenti potenzialmente aggressivi e poco sicuri. Per il riconoscimento dello stile di guida sono state fatte ricerche utilizzando dispositivi diversi come lo smartphone, gli indossabili e i bus CAN delle auto. Lo scopo di questi studi è di classificare il tipo di guida in auto e di riconoscere situazioni pericolose che vanno segnalate al guidatore sensibilizzarlo alla guida sicura.

2.1.1 MIROAD

Gli autori di [7] hanno sviluppato un sistema basato sullo smartphone accessibile ed economico per il riconoscimento di determinati eventi alla guida il suo nome è MIROAD (A Mobile-Sensor-Platform for Intelligent Recognition Of Aggressive Driving). Il sistema viene posizionato al centro del cruscotto con la fotocamera posteriore verso l'esterno mentre vengono usati i seguenti sensori: la fotocamera, l'accelerometro, il giroscopio, il magnetometro e il GPS.

Gli eventi riconosciuti da MIROAD sono:

- Svolta a destra/sinistra di 90°
- Inversione di marcia 180°
- Svolta aggressiva a destra/sinistra di 90°
- Inversione di marcia aggressiva 180°
- Accelerazione aggressiva
- Frenata aggressiva
- Cambio aggressivo della corsia
- Eccesso di velocità
- Rimozione del dispositivo

Il sistema ha due modalità di funzionamento: Attiva e Passiva. Nella modalità attiva sta in ascolto degli eventi ma non registra i dati finché non viene rilevato un evento aggressivo, in quel caso viene attivata la registrazione di tutti i sensori per cinque minuti e nello stesso tempo viene anche registrato un video. Il suo scopo è di incrementare la sicurezza quindi il guidatore viene notificato tramite l'audio dell'evento potenzialmente pericoloso. Nella modalità passiva invece tutti i dati vengono memorizzati per analisi future sempre in finestre di tempo di cinque minuti con la possibilità di riprodurre in un secondo momento il video sincronizzato ai sensori per riesaminare i dati.

Per il riconoscimento della manovra viene prima determinato l'inizio e la fine della stessa, dopo si confrontano i dati correnti con dei pattern registrati in precedenza che vengono utilizzati da un classificatore identificando il tipo di manovra e la presenza di aggressività.

2.1.2 DNA del guidatore

Il bus CAN (Controller Area Network) è uno standard seriale per i bus di campo per collegare diverse unità di controllo elettronico (ECU), usato

principalmente nell'ambito automotive come bus per gli autoveicoli. Nelle autovetture moderne questo standard viene implementato e con l'aumento di delle ECU che aggiungono sempre più sensori rende possibile un riconoscimento dello stile di guida più diretto ed affidabile. Fugiglando *et al.* in [12] presentano il concetto di "DNA del guidatore" come un modo per descrivere la complessità del comportamento della guida tramite un insieme di quantità individuali facili da misurare. Queste quantità sono responsabili di alcuni aspetti del comportamento di guida come nella metafora i geni sono responsabili dei tratti di un individuo.

Il DNA del guidatore può essere interpretato come l'insieme dei fattori che in relazione all'ambiente esterno determinano le attitudini di guida della persona. Questi fattori vengono chiamati dimensioni del DNA (come i geni nella metafora) che rappresentano le caratteristiche del comportamento di guida determinate da quantità misurabili.

Le quattro dimensioni che compongono il DNA sono:

Frenata (guida prudente) questa misura rappresenta l'intensità delle frenate che potrebbero essere aggressive, confortevoli o sicure. Per determinare il valore istantaneo viene usata l'accelerazione frontale normalizzata con il percentile della distribuzione di tutti i guidatori nel database. Il punteggio va da 0 a 5, un valore basso rappresenta un pattern di frenata più aggressivo della media, uno alto invece si riferisce a un comportamento migliore.

Svolta (guida attenta) questa dimensione è molto simile alla precedente con la differenza che viene usata l'accelerazione laterale. La misura descrive l'aggressività nella svolta e più in generale nel uso del volante sopratutto a velocità elevate.

Eccesso di velocità (guida sicura) questo fattore rappresenta il rischio di incidente indotto dall'eccesso di velocità sopratutto in caso di condizioni meteorologiche avverse. Per determinare questa misura vengo-

no usati questi fattori: la velocità, il limite della velocità (GPS) e la presenza di pioggia. Il punteggio istantaneo va da 1 a 5.

Efficienza energetica (consumo di carburante) questa misura rappresenta il consumo di carburante determinato dai giri motore (RPM). Il punteggio viene confrontato con la distribuzione dei valori degli altri guidatori nel database similmente alla frenata.

Ogni guidatore j viene caratterizzato dal proprio $DNA(d_{j,1}, ..., d_{j,4})$ di guida dove $d_{j,i}$ è la media dei valori istantanei del i-simo fattore. Per confrontare il comportamento dei guidatori si è pensato di introdurre un punteggio sintetico che va da 1 a 100 che rappresenta qualità del comportamento di guida di ogni persona. Vengono proposti tre metodi di per il calcolo del punteggio totale: fare la media di tutti fattori, fare la media pesata usando dei pesi che determinano l'importanza di un fattore e infine proiettare le quattro dimensioni in una dimensione unica eseguendo una Principal Component Analysis (PCA).

Il differenza principale di questo approccio rispetto al classico metodo con il machine learning è la poca necessità di potenza di calcolo, questo permette di implementare visualizzazione real-time da inserire all'interno delle auto in modo da assistere il guidatore a migliorare lo stile di guida.

2.2 Riconoscimento del guidatore

Nonostante i divieti di legge la distrazione causata dall'uso dello smartphone alla guida porta migliaia di fatalità ogni anno e il numero è sempre in crescita. Anche l'industria della telefonia mobile sta cercando di gestire la distrazione con delle app che attivano la modalità silenzioso e disabilitano le notifiche quando viene attivata la modalità di guida. Queste applicazioni dovrebbero capire quando un utente è alla guida perché l'esperienza mostra che gli utenti non sono molto disponibili a impostare la modalità manualmente. In letteratura ci sono diversi metodi per determinare se l'utente di uno

smartphone si trova all'interno di un veicolo e.g. [10], è più difficile invece determinare se l'utente è il guidatore oppure un semplice passeggero.

Gli autori di [6] presentano un metodo low-infrastructure che percepisce l'accelerazione dovuta alle dinamiche del veicolo per determinare la posizione dello smartphone all'interno del veicolo. La posizione viene usata come euristica per determinare se il dispositivo viene usato dal guidatore o da un passeggero. Il metodo si basa sul fatto che l'accelerazione centripeta varia in base alla posizione nell'auto e quindi si tratta di confrontare l'accelerazione dello smartphone con l'accelerazione di un punto di riferimento per determinare la posizione del dispositivo.

2.2.1 Fenomeno fisico

Quando un veicolo svolta si sviluppa una forza centripeta in direzione ortogonale alla direzione del movimento del veicolo e verso il centro della svolta. Questa forza genera un'accelerazione centripeta a che punta sempre verso il centro della curva. Assumendo che la svolta sia un cerchio perfetto, l'accelerazione centripeta (a) può essere ottenuta dalla velocità angolare (ω), velocità tangenziale (v) e il raggio (r) della curva come segue:

$$a = \omega v = \omega^2 r = \frac{v^2}{r} \tag{2.1}$$

La relazione di questi parametri è illustrata nella figura 2.1. I dispositivi posizionati dalla parte del guidatore e dalla parte del passeggero avranno la stessa velocità angolare ma raggi diversi. In base all'equazione (2.1) si può vedere che un raggio diverso con la stessa velocità angolare porterà a una differenza nell'accelerazione centripete nelle posizioni rispettive.

2.2.2 Funzionamento del sistema

Il sistema si basa sul fatto che l'accelerazione centripeta cambia in base alla posizione dello smartphone nel veicolo e per confrontarla viene usato un punto di riferimento. Poiché viene usato un punto di riferimento per

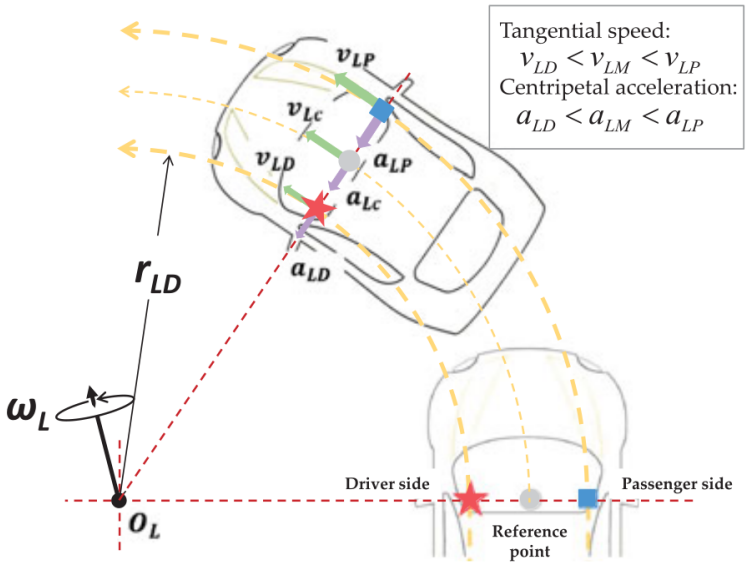


Figura 2.1: Illustrazione della diversa accelerazione centripeta in parti differenti del veicolo

poter fare la differenza di accelerazione basta una sola curva per trovare la soluzione, il suo ruolo diventa fondamentale per questo motivo nel sistema sono implementati tre punti di riferimenti diversi:

1. Un economico adattatore per accendisigari che contiene un accelerometro, visto che l'accendisigari è al centro il sistema riesce direttamente a confrontare i valori dell'accelerometro con lo smartphone.

2. La velocità del veicolo ottenuta dalla porta OBD-II può essere usata per calcolare l'accelerazione centripeta al centro del veicolo e questa può essere confrontata con il valore dello smartphone.

3. Se presente può essere usato lo smarphone di un passeggero.

La realizzazione dell'approccio proposto viene suddivisa in tre sotto task:

Allineamento delle coordinate viene fatto quando lo smartphone cambia posizione in modo tale che l'accelerazione centripeta e la velocità angolare del telefono siano allineate a quelle del veicolo.

Calibrazione dei dati viene fatta dopo l'inizio di una curva, lo smartphone inizia a raccogliere dati dal punto di riferimento e fa la calibrazione sia sui suoi dati che su quelli raccolti. La calibrazione consiste in tre passi: interpolazione dei dati, sincronizzazione delle tracce, aggiustamenti all'accelerazione, che hanno come obiettivo di sincronizzare le tracce e di ridurre il rumore dei dati causati da diversi hardware.

Riconoscimento della posizione usando la differenza cumulativa dell'accelerazione centripeta combinata alla direzione della curva determinata dal segno della velocità angolare.

2.2.3 Algoritmo di riconoscimento

E' essenziale capire quali sono i fattori più importanti che contribuiscono alla differenza dell'accelerazione centripeta in due punti diversi del veicolo. Il seguente lemma evidenzia questi fattori:

Lemma 2.2.1. *La differenza di accelerazione centripeta fra due posizioni all'interno del veicolo è determinata dalla velocità angolare e la distanza relativa fra le due posizioni.*

Pertanto, data la differenza dell'accelerazione centripeta e la direzione il sistema proposto è in grado di determinare se il dispositivo è del guidatore o del passeggero. L'algoritmo determina la posizione usando la seguente ipotesi:

$$\begin{cases} (a - a_M)\omega > 0, \mathcal{H}_0 : passeggero \\ (a - a_M)\omega < 0, \mathcal{H}_1 : guidatore \end{cases} \tag{2.2}$$

dove a è l'accelerazione centripeta dello smartphone misurata dall'asse X del accelerometro, a_M è l'accelerazione centripeta del punto di riferimento e ω è la velocità angolare misurata dall'asse Z del giroscopio dello smartphone. Il segno di ω determina la direzione della curva, e.g. ω è positivo quando il veicolo gira a sinistra.

L'algoritmo può migliorare il riconoscimento combinando i risultati di più curve (e.g. N curve) attraverso questo semplice voto di maggioranza:

$$\begin{cases} \sum_{i=1}^{N} \frac{(a^i - a_M^i)\omega^i}{|(a^i - a_M^i)\omega^i|} > 0, \mathcal{H}_0 : passeggero \\ \sum_{i=1}^{N} \frac{(a^i - a_M^i)\omega^i}{|(a^i - a_M^i)\omega^i|} < 0, \mathcal{H}_1 : guidatore \end{cases} \tag{2.3}$$

dove a^i, a_M^i e ω^i sono l'accelerazione centripeta dello smartphone, l'accelerazione centripeta del punto di riferimento e la velocità angolare dello smartphone nell'i-esima curva.

2.2.4 Valutazione

La valutazione è stata fatta in condizioni di guida reali usando tutti i punti di riferimento proposti. Le metriche di valutazione sono:

Accuratezza definita come la percentuale di curve classificate correttamente.

Percentuale di riconoscimento e Percentuale dei falsi positivi la percentuale di riconoscimento è la percentuale degli smartphone guidatore riconosciuti correttamente dal sistema, invece percentuale dei falsi positivi è la percentuale degli smartphone passeggero riconosciuti come guidatore.

Latenza di riconoscimento è il tempo impiegato dal sistema per fare la decisione.

I risultati della valutazione sono soddisfacenti in quanto con un'unica curva arriva a superare l'80% la percentuale di riconoscimento e meno di 10% i falsi positivi. La situazione migliora quando si usando più curve, nel caso di tre curve arriva al 94% i riconosciuti correttamente e a 3% i falsi positivi, aumentando le curve a cinque i riconosciuti salgono a quasi 90% e i falsi positivi arrivano a meno di 1%. La latenza è composta dal tempo di esecuzione e dal tempo di svolta, l'algoritmo impiega del tempo trascurabile sotto al millisecondo quindi il tempo determinante è quello della svolta, che

negli esperimenti fatti mediamente è di 10 secondi. Questo sistema arriva a un'accuratezza molto elevata con poche curve, i risultati presentati sono la media di due esperimenti fatti in città diverse.

Capitolo 3

Raccolta dati

I dati sono fondamentali per questo tipo di ricerca, per costruire un modello di apprendimento automatico per il riconoscimento della posizione in auto è necessario avere dati dei sensori registrati in diverse posizioni del veicolo facendo lo stesso percorso. Come visto in 2.2.3 l'accelerazione centripeta nelle svolte cambia a seconda della posizione nel veicolo, avendo i dati derivanti da più posizioni è possibile addestrare un classificatore. In letteratura non sono stati pubblicati dataset così specifici, ci sono [16, 17] che contengono percorsi registrati in auto ma da una sola posizione e hanno dimensioni molto ridotte. Pertanto una parte di questo lavoro è stata la raccolta dei dati grezzi, si è deciso di registrare i percorsi da tre posizioni diverse: lato sinistro (guidatore), centro e lato destro (passeggero). In questo capitolo viene descritta la procedura di raccolta dei dati e il dataset prodotto.

3.1 Dispositivi di registrazione

La grande quantità di sensori presenti negli smartphone permette di usarli come dispositivi di raccolta dei dati. Il modello di smartphone e le sue caratteristiche tecniche non sono rilevanti, è sufficiente che abbia tutti i sensori necessari: accelerometro, giroscopio, magnetometro e GPS. Si è deciso di usare smartphone identici in quanto sensori diversi possono causare differenze

nei valori rilevati e quindi più difficili da confrontare. Per la registrazione dei dati sono stati procurati tre smartphone Yotaphone 2 con il sistema operativo Android.

3.2 Applicazione Android

Per la raccolta dei dati è stata sviluppata una semplice applicazione Android: SensorsRecorder, mostrata nella figura 3.1.

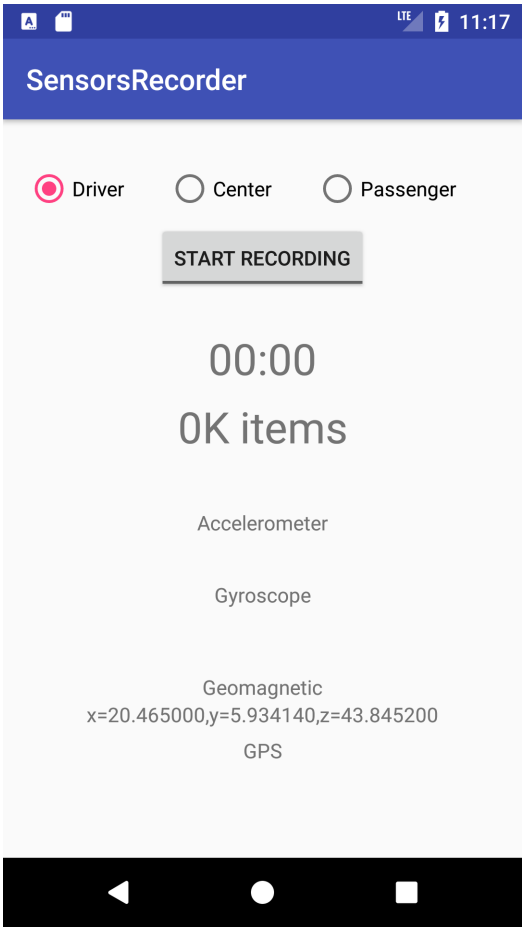


Figura 3.1: Applicazione Android - SensorsRecorder

Le sue funzionalità sono:

- specificare la posizione del dispositivo all'interno dell'auto

- avviare/fermare la registrazione
- visualizzare la durata della registrazione
- visualizzare la quantità di record registrati
- visualizzare i valori dei sensori in tempo reale
- salvare nella memoria di massa i dati grezzi

L'applicazione campiona i dati dei sensori a una frequenza di 100 Hz, fa eccezione solo il GPS che viene campionato al cambiamento della posizione invece che a una frequenza predefinita. Per prevenire perdite di dati il salvataggio nella memoria di massa viene fatto in continuo ogni 1000 record, è un'operazione asincrona e quindi non interferisce con il campionamento. I file prodotti sono nel formato CSV il cui nome contiene il posizionamento dello smartphone, la data e l'orario di avvio della registrazione, ll loro contenuto è di un record per riga. La struttura dei record è descritta nelle figure 3.2, 3.3.

```
{tipo_sensore};{asse_x};{asse_y};{asse_z};{tempo}
```

Figura 3.2: Struttura dei record dei sensori di movimento e posizionamento

```
P;{latitudine};{longitudine};{altitudine};{velocità};{tempo}
```

Figura 3.3: Struttura dei record GPS

Elenco dei sensori registrati:

- Accelerometro
- Accelerometro lineare, sensore software
- Giroscopio
- Magnetometro
- GPS

3.3 Autovettura

L'autovettura usata per registrare i dati è stata una Fiat Punto 188 del 2013 bifuel benzina/metano. Gli smartphone sono stati incollati con biadesivo, il primo dalla parte del guidatore, il secondo al centro e l'ultimo dalla parte del passeggero come mostrato nella figura 3.4.

Figura 3.4: Posizionamento smartphone nell'autovettura

3.4 Dataset creato

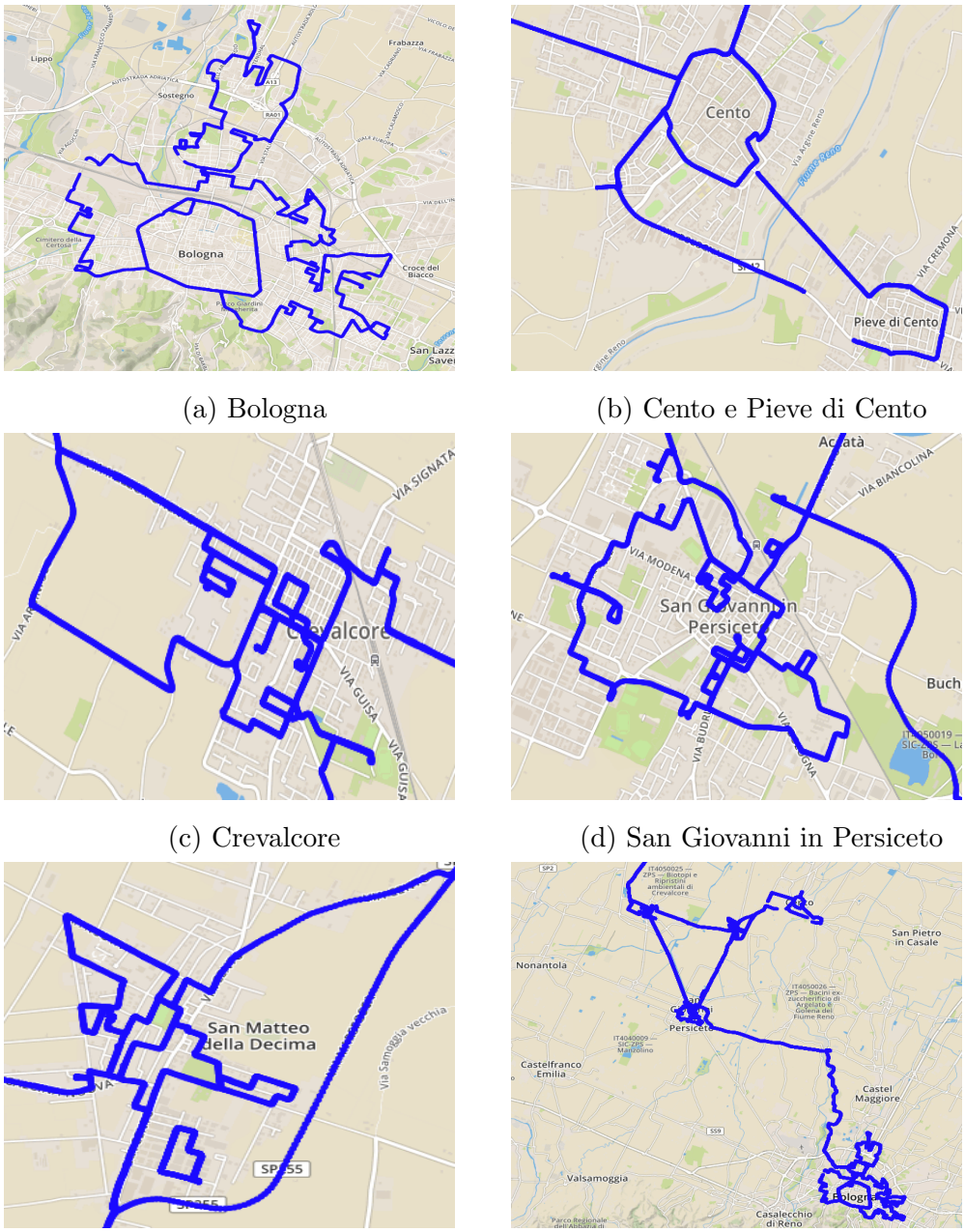
Il dataset creato è composto di tre parti che rappresentano lo stesso percorso ma registrati in tre posizioni diverse all'interno del veicolo: a sinistra (guidatore), al centro e a destra (passeggero). Ogni parte è composta da circa 7,5 milioni di record che arrivano ad avere una dimensione di circa 290MB per un totale di circa 22,5 milioni di record e 870MB. Il percorso complessivo corrisponde a circa 190 km percorsi in circa 6 ore di guida.

3.4.1 Distribuzione geografica

Per creare un dataset di qualità e più rappresentativo della realtà si è ritenuto importante registrare in località diverse sia in su reti stradali urbane che extraurbane.

La registrazione dei dati è stata effettuata nelle seguenti località (100% - tutti i dati registrati):

- Bologna, 47%, figura 3.5a
- Cento e Pieve di Cento, 5%, figura 3.5b
- Crevalcore, 8%, figura 3.5c
- San Giovanni in Persiceto, 11%, figura 3.5d
- San Matteo della Decima, 6%, figura 3.5e
- Percorso extraurbano che collega le località sopra elencate, 23%, figura 3.5f



(a) Bologna (b) Cento e Pieve di Cento

(c) Crevalcore (d) San Giovanni in Persiceto

(e) San Matteo della Decima (f) Visione complessiva

Figura 3.5: Percorsi effettuati per la raccolta dei dati

3.4.2 Normalizzazione

Visto che i dati grezzi provengono da tre smartphone distinti che hanno posizioni diverse e non è stato possibile avviare la registrazione contempora-

neamente per poterli correlare tra di loro sono state necessarie delle normalizzazioni: la sincronizzazione delle tracce e la normalizzazione dell'ampiezza.

Sincronizzazione delle tracce

Poiché la registrazione non avviane nello stesso istante in tutti i dispositivi le tracce sono disallineate nel tempo. Per una giusta correlazione dei dati e per la possibilità di confronto è necessario che le tracce siano allineate rispetto al tempo. Come prima idea potrebbe essere quella di eliminare tutti i dati fino alla prima curva in questo modo le tracce saranno allineate, questo approccio non va bene perché la parte iniziale di quando il veicolo è fermo è essenziale per la seconda normalizzazione. La sincronizzazione è stata fatta manualmente con l'aiuto di uno script e consiste nell'eliminare porzioni di dati non rilevanti (il tempo in cui l'auto è ancora ferma prima di partire) in modo tale che in tutte le tracce dello stesso percorso il momento della partenza sia nello stesso tempo. Viene fatta due tracce alla volta, e.g. guidatore e centrale poi centrale e passeggero.

Più nello specifico il processo è composto dai seguenti passi:

1. Identificare il tempo della prima curva delle entrambe le tracce T_1 *e* T_2 indicati con t_1 *e* t_2.

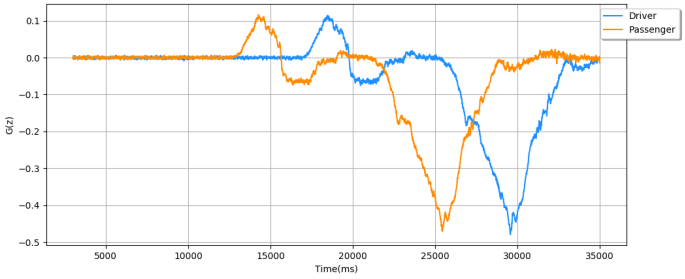
2. Confrontare t_1 *e* t_2 e individuare così la traccia iniziata prima: se $t_1 > t_2$ allora T_1 è iniziata prima altrimenti T_2.

3. Eliminare la porzione della traccia che va dal tempo 0 al tempo $|t_1 - t_2|$ dalla traccia iniziata prima.

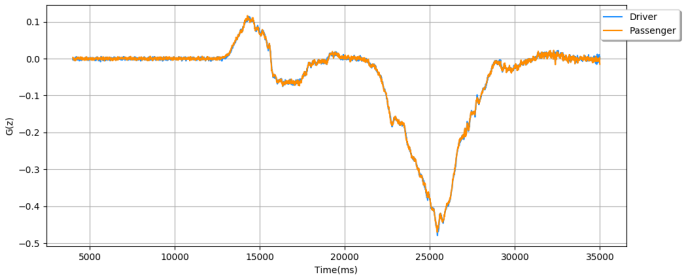
Nella figura 3.6 sono mostrate due tracce prima e dopo la sincronizzazione.

Normalizzazione dell'ampiezza

Per confrontare le tracce di diverse posizioni all'interno dell'auto è indispensabile che gli smartphone durante la registrazione siano posizionati nella stessa direzione e inclinazione. Posizionare i dispositivi in questo modo è



(a) Tracce non sincronizzate



(b) Tracce sincronizzate

Figura 3.6: Sincronizzazione temporale

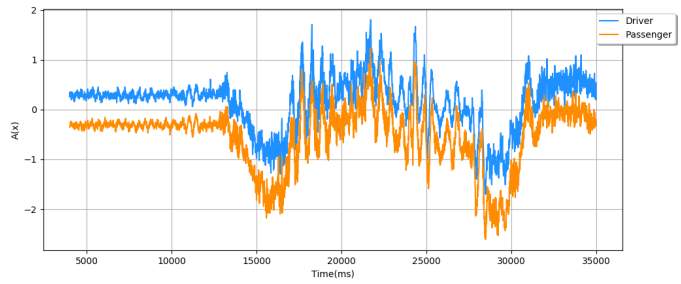
molto difficile e quindi è stato implementato tale metodo di normalizzazione sui dati già registrati. Questo processo viene fatto in automatico da uno script e a differenza della normalizzazione temporale nella quale si modifica la traccia nel suo complesso in questo caso in ogni record vengono aggiornati i valori delle assi. Il principio di funzionamento è molto semplice: da ogni record viene sottratta la media dei primi valori registrati quando l'auto era ancora ferma, ogni traccia all'inizio contiene una porzione di quando l'auto è ancora ferma.

Più nello specifico il processo è composto dai seguenti passi:

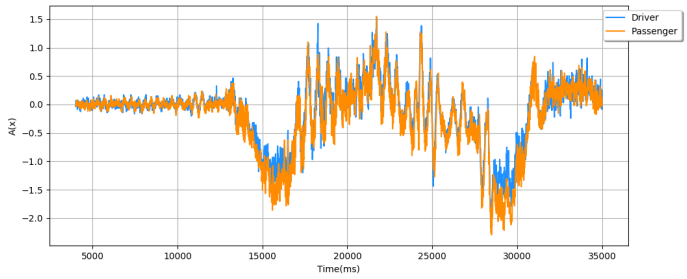
1. Viene fatta la media dei primi 400 elementi della traccia indicata con μT_0^{400}.

2. Per ogni elemento della traccia T viene sottratto μT_0^{400}.

Nella figura 3.7 sono mostrate due tracce prima e dopo la normalizzazione.



(a) Tracce non normalizzate



(b) Tracce normalizzate

Figura 3.7: Normalizzazione dell'ampiezza

3.5 Trasformazione dei dati

Il dataset descritto precedentemente contiene valori grezzi dei sensori senza dare a loro una semantica particolare, per questa ricerca sono necessari solo i dati corrispondenti a delle curve, come descritto nella sezione 2.2.1 in caso di svolta si crea la differenza l'accelerazione centripeta a seconda della posizione dello smartphone. Pertanto vengono considerati soltanto i dati in corrispondenza delle curve dopo averle individuate.

3.5.1 Riconoscimento delle curve usando il giroscopio

Il giroscopio rileva la velocità angolare quindi è possibile usare il suo valore per determinare i confini di una curva e anche la sua direzione. A seconda della posizione dello smartphone bisogna considerare un asse piuttosto che un altro, nel dataset registrato gli smartphone erano posizionati come mo-

strato nella figura 3.4 quindi l'asse da considerare è lo Z. Siccome il valore rappresenta la velocità angolare una curva inizia quando l'asse Z diventa diversa da 0 e finisce quando ritorna a valere 0. Per rendere più stabile il riconoscimento e non prendere in considerazione ogni rumore viene presa come soglia 0.04 sulla media degli ultimi 10 valori. La direzione della curva viene determinata dal segno della velocità angolare, segno positivo corrisponde a una curva sinistra e destra nel caso contrario.

L'algoritmo di riconoscimento delle curve e la loro direzione è descritto nella seguente equazione:

$$\begin{cases} \frac{\sum_{t=T}^{T+10} G(Z,t)}{10} \geq 0.04, \mathcal{H}_0 : curva\ a\ sinistra \\ \frac{\sum_{t=T}^{T+10} G(Z,t)}{10} \leq -0.04, \mathcal{H}_1 : curva\ a\ destra \end{cases} \tag{3.1}$$

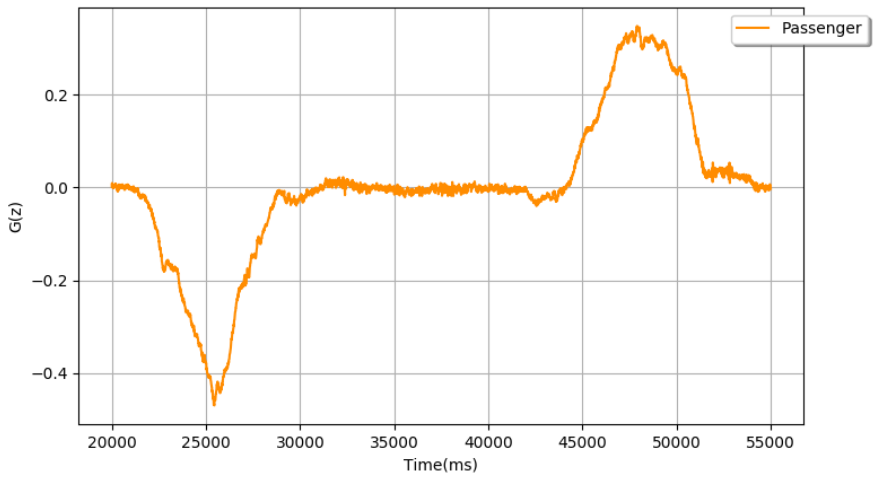
dove T è il tempo di inizio osservazione e $G(Z,t)$ è il valore dell'asse Z del giroscopio rilevato all'istante t.

Nella figura 3.8 vengono mostrati due grafici dell'asse Z del giroscopio prima e dopo il riconoscimento delle curve.

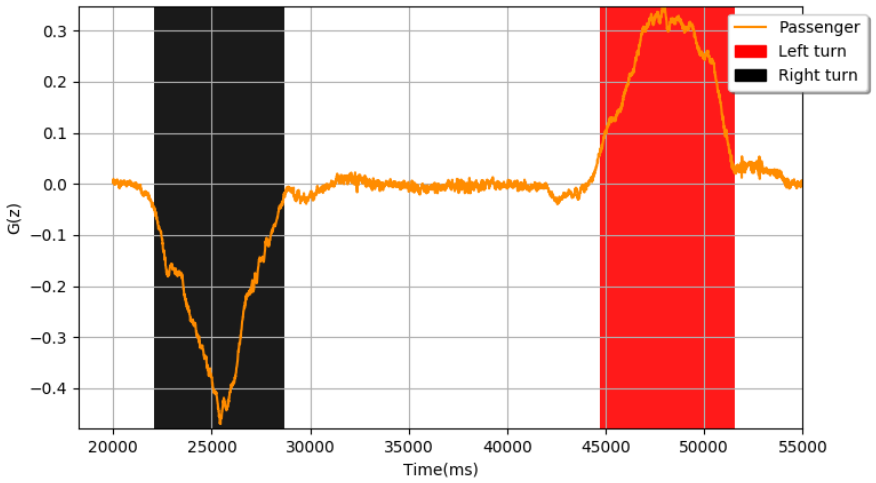
3.5.2 Calcolo ampiezza delle curve

L'ampiezza della curva è una caratteristica importante perché è un fattore da cui dipende l'accelerazione centripeta come si vede nell'equazione 2.1. Per confrontare due curve distinte non basta l'accelerazione centripeta se non si conosce anche l'ampiezza delle stesse. Per determinare l'ampiezza è possibile usare il valore del magnetometro che misura l'intensità del campo magnetico e quindi la direzione del veicolo. In base alla posizione degli smartphone come mostrato nella figura 3.4 l'asse X del magnetometro si è rilevato utile per questo tipo di misurazione.

L'idea alla base per il calcolo dell'ampiezza è di fare la differenza tra il valore del magnetometro all'inizio della curva e il valore alla fine della curva, in questo modo si ottiene la differenza della direzione e di cui viene fatto il valore assoluto in quanto non ci interessa il segno ma solo l'ampiezza. Nella figura 3.9 viene mostrato il grafico del magnetometro rappresentante due



(a) Prima del riconoscimento delle curve



(b) Dopo il riconoscimento delle curve

Figura 3.8: Asse Z del giroscopio, riconoscimento curve

curve di ampiezza diversa, si può notare che come cambia la differenza, nel primo caso è di $30\mu T$ mentre nel secondo $50\mu T$ quindi si può dedurre che la seconda curva è molto più ampia della prima.

L'equazione seguente definisce il calcolo dell'ampiezza:

$$a = |M(X,t) - M(X,t')| \tag{3.2}$$

dove t è l'istante di inizio curva, t' è l'istante di fine curva e $M(X,t)$ è il valore dell'asse X del magnetometro rilevato all'istante t.

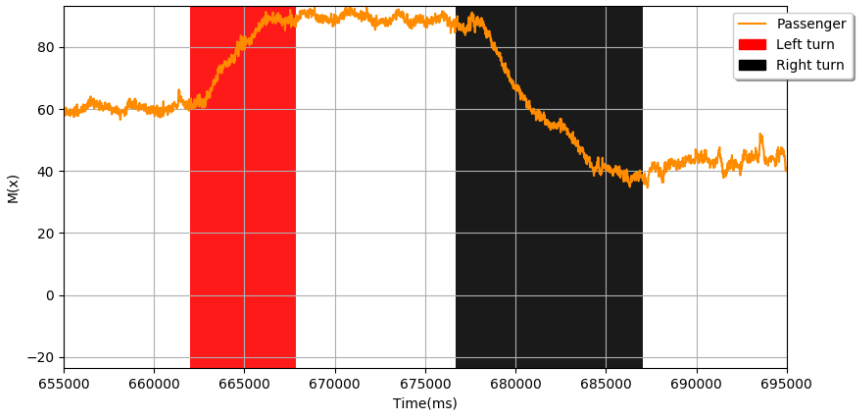


Figura 3.9: Asse X del magnetometro, due curve di ampiezza diversa

3.5.3 Selezione delle caratteristiche delle curve

Dopo aver preso in considerazione solo la porzione dei dati che riguardano le curve il dataset si è ridotto di molto ma non abbastanza perché per ogni curva ci sono ancora migliaia di record e il loro numero cambia in base alla durata della stessa. Come accennato nella sezione 1.3.3 per ottenere un dataset per l'apprendimento automatico è necessario che ogni istanza abbia lo stesso numero di caratteristiche.

Si è considerato opportuno rappresentare una curva con le seguenti caratteristiche:

direzione destra/sinistra determinata come descritto precedentemente.

durata in millisecondi è la differenza del tempi di fine e inizio curva.

ampiezza determinata con il metodo descritto precedentemente.

valori accelerometro comprende il *minimo*, *massimo* e la *media* dei valori dell'asse X dell'accelerometro, contiene solo l'asse X perché come spiegato nella sezione 2.2.3 è l'asse che rappresenta l'accelerazione centripeta.

valori accelerometro lineare comprende il *minimo*, *massimo* e la *media* dei valori dell'asse X dell'accelerometro lineare, per lo stesso motivo di sopra.

valori giroscopio comprende il *minimo*, *massimo* e la *media* dei valori dell'asse Z del giroscopio, rappresenta la velocità angolare.

3.6 Creazione di nuovi dataset

Il processo di trasformazione dei dati descritto nella sezione precedente ha prodotto nuovi dati in cui il focus è sulla curva invece che sul valore grezzo del sensore. Nel dataset così creato ogni record rappresenta una curva contenente le caratteristiche elencate precedentemente. La sua dimensione si è ridotta di molto, sono circa 1100 curve (570 a destra e 525 a sinistra) uniche per ogni traccia, si ricorda che ogni traccia contiene le stesse curve ma da posizioni diverse nel veicolo: guidatore, centrale, passeggero. Successivamente vengono descritti i due dataset creati per l'apprendimento automatico.

3.6.1 Dataset - due tracce contemporaneamente

Si è voluto riprodurre la ricerca descritta nella sezione 2.2, cioè usare lo smartphone e un altro punto di riferimento per riconoscere se lo smartphone viene usato dal guidatore, come punto di riferimento verrà usato il secondo smartphone. Dalle tre traccie disponibili vengono usate solo due, cioè quella del guidatore e quella del passeggero, dove una viene considerato target e l'altra punto di riferimento. In questa ricerca invece di usare l'algoritmo

originale descritto nella sezione 2.2.3 viene usato l'apprendimento automatico, nello specifico un classificatore supervisionato. L'idea è quindi quella di riconoscere la posizione dello smartphone usando i dati di due smartphone contemporaneamente e facendo la differenza dei valori dei sensori.

Per ogni curva viene inserito nel record le caratteristiche della curva e le informazioni delle due tracce. La struttura di un record è definita dalle seguenti relazioni:

$$R = d, t, a, T_{Ax,p}, T_{Lx,p}, T_{Gz,p}, T_{Ax,p,s}, T_{Lx,p,s}, T_{Gz,p,s}, p$$

$$T_{Ax,p} = m_{Ax,p}, \mu_{Ax,p}, M_{Ax,p}$$

$$T_{Lx,p} = m_{Lx,p}, \mu_{Lx,p}, M_{Lx,p}$$

$$T_{Gz,p} = m_{Gz,p}, \mu_{Gz,p}, M_{Gz,p}$$

$$T_{Ax,p,s} = m_{Ax,p} - m_{Ax,s}, \mu_{Ax,p} - \mu_{Ax,s}, M_{Ax,p} - M_{Ax,s}$$

$$T_{Lx,p,s} = m_{Lx,p} - m_{Lx,s}, \mu_{Lx,p} - \mu_{Lx,s}, M_{Lx,p} - M_{Lx,s}$$

$$T_{Gz,p,s} = m_{Gz,p} - m_{Gz,s}, \mu_{Gz,p} - \mu_{Gz,s}, M_{Gz,p} - M_{Gz,s}$$

Dove d, t, a sono: direzione, durata e ampiezza della curva; m, M sono: minimo e massimo dell'asse di un sensore e.g. $m_{Ax,p}$ asse x dell'accelerometro, prima traccia; p la posizione dello smartphone che è un valore binario: 1 guidatore, 0 passeggero, questo campo è l'output che verrà usato per addestrare il classificatore e anche per valutare i suoi risultati. Le caratteristiche di input quindi escludendo la posizione sono 21.

Per fare le previsioni verrà costruito un record come descritto precedentemente escludendo la posizione che verrà appunto prevista dal modello. Per costruire un record è necessario avere due smartphone uno posizionato dalla parte del guidatore e l'altro dalla parte del passeggero. Per avere una previsione basta avere anche una sola curva, ma bisogna attendere la fine della curva per poter procedere.

Per avere un dataset da usare con il classificatore binario (guidatore / passeggero) i record devono di classi diverse, siccome in questo caso si parla di

curve, si è deciso di inserire ogni curva come due record diversi, che sono due punti di vista differenti. Nel primo viene inserita la curva usato lo smartphone del guidatore come prima posizione e lo smartphone del passeggero come punto di riferimento, nel secondo al contrario, avendo settato il campo della posizione alternato. Il dataset così costruito in totale ha circa 2200 record che sono classificati in questo modo: 50% guidatore e 50% passeggero.

3.6.2 Dataset - una traccia per volta

Siccome nel dataset precedente ogni record tiene traccia di entrambi gli smartphone questo implica che anche per fare le previsioni bisogna avere due smartphone, quindi se questo metodo dovesse essere implementato non sarebbe possibile riconoscere la posizione quando nel veicolo è presente una sola persona. Per questo motivo è stato creato un altro dataset che prende in considerazione un solo smartphone alla volta, quindi ogni record contiene i dati di un solo dispositivo. Costruendo un dataset in questo modo non si riesce a sfruttare il concetto di base dell'algoritmo descritto in 2.2.3 perché non c'è la differenza di accelerazione centripeta tra due posizioni diverse nello stesso veicolo.

L'idea alla base di questo nuovo metodo è di mettere a confronto i dati provenienti da più curve diverse invece che da posizioni diverse. Supponiamo di avere due curve che sono cerchi perfetti una a destra e l'altra a sinistra, analizzando solo il lato guidatore, notiamo che nella curva destra il raggio è maggiore perché il dispositivo è più lontano dal centro del cerchio, e anche la velocità tangenziale è più grande, basandosi sull'equazione 2.1 allora anche l'accelerazione centripeta sarà più grande rispetto alla curva sinistra. La relazione di questi parametri in questa situazione è illustrata nella figura 3.10.

Il record contiene le caratteristiche della curva di cui si vuole sapere la posizione dello smartphone in più le caratteristiche delle N curve precedenti di direzione opposta alla curva corrente e M curve precedenti della stessa direzione. La struttura dei record è descritta dalle seguenti relazioni:

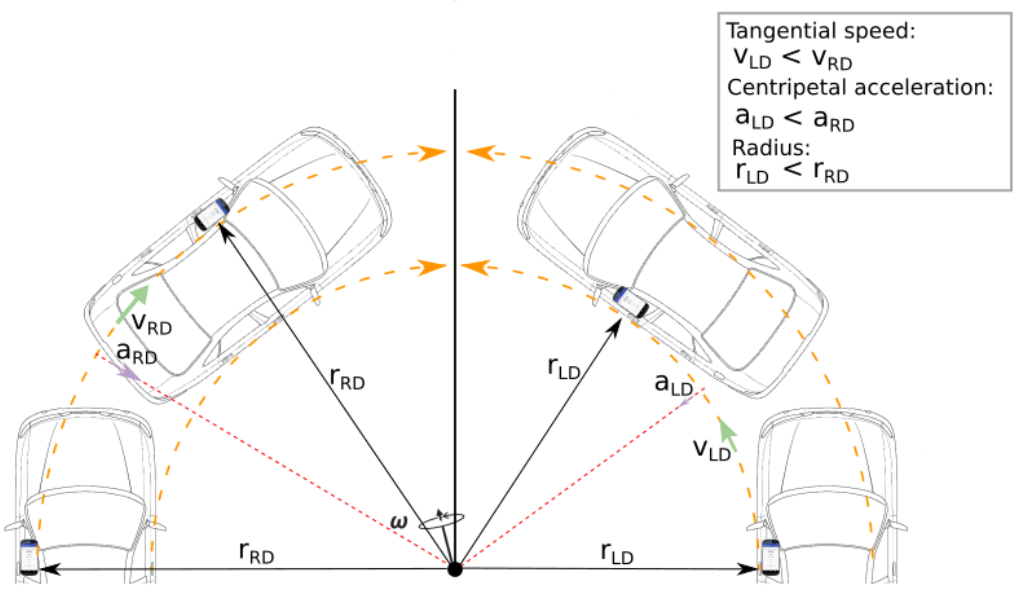


Figura 3.10: Illustrazione diversa accelerazione centripeta in curve diverse dalla parte del guidatore

$$R^i = \{d_i\} \cup t^i \cup a^i \cup Ax^i \cup Lx^i \cup Gz^i \cup \{p_s\}$$

$$d_i : direzione\ della\ i - esima\ curva$$

$$t_i : durata\ della\ i - esima\ curva$$

$$a_i : ampiezza\ della\ i - esima\ curva$$

$$t^i = \{t_i\} \cup \{t_i - t_c \forall c \in N^i\} \cup \{t_i - t_c \forall c \in M^i\}$$

$$N^i = \{c \in N : i_c > (|N| - \lambda_N), i_c <= i\}$$

$$M^i = \{c \in M : i_c > (|M| - \lambda_M), i_c <= i\}$$

$$N = \{c \in C : d_c \neq d_i\}$$

$$M = \{c \in C : d_c = d_i\}$$

$$C : tutte\ le\ curve\ fino\ alla\ i - esima$$

$$\lambda_N : numero\ di\ curve\ precedenti\ di\ direzione\ opposta\ alla\ i - esima$$

$$\lambda_M : numero\ di\ curve\ precedenti\ di\ direzione\ uguale\ alla\ i-esima$$

$$a^i = \{a_i\} \cup \{a_i - a_c \forall c \in N^i\} \cup \{a_i - a_c \forall c \in M^i\}$$

$$Ax^i = \{m_{Ax,i}, \mu_{Ax,i}, M_{Ax,i}\} \cup Ax^i_N \cup Ax^i_M$$

$$Ax^i_N = \{m_{Ax,i} - m_{Ax,i,N}, \mu_{Ax,i} - \mu_{Ax,i,N}, M_{Ax,i} - M_{Ax,i,N}\}$$

$$Ax^i_M = \{m_{Ax,i} - m_{Ax,i,M}, \mu_{Ax,i} - \mu_{Ax,i,M}, M_{Ax,i} - M_{Ax,i,M}\}$$

$$Lx^i = \{m_{Lx,i}, \mu_{Lx,i}, M_{Lx,i}\} \cup Lx^i_N \cup Lx^i_M$$

$$Lx^i_N = \{m_{Lx,i} - m_{Lx,i,N}, \mu_{Lx,i} - \mu_{Lx,i,N}, M_{Lx,i} - M_{Lx,i,N}\}$$

$$Lx^i_M = \{m_{Lx,i} - m_{Lx,i,M}, \mu_{Lx,i} - \mu_{Lx,i,M}, M_{Lx,i} - M_{Lx,i,M}\}$$

$$Gz^i = \{m_{Gz,i}, \mu_{Gz,i}, M_{Gz,i}\} \cup Gz^i_N \cup Gz^i_M$$

$$Gz^i_N = \{m_{Gz,i} - m_{Gz,i,N}, \mu_{Gz,i} - \mu_{Gz,i,N}, M_{Gz,i} - M_{Gz,i,N}\}$$

$$Gz^i_M = \{m_{Gz,i} - m_{Gz,i,M}, \mu_{Gz,i} - \mu_{Gz,i,M}, M_{Gz,i} - M_{Gz,i,M}\}$$

$$p_s : posizione\ dello\ smartphone$$

Dove m, M sono: minimo e massimo dell'asse di un sensore e.g. $m_{Ax,i}$ asse x dell'accelerometro della curva i. Il numero di caratteristiche varia a seconda della scelta dei valori di λ_N e λ_M, una volta scelti rimangono invariati nel dataset creato, con $\lambda_N = \lambda_M = 1$ si hanno 34 caratteristiche. Si possono creare diversi dataset con valori differenti per analizzare il risultato al cambiare di λ_N e λ_M. Per fare le previsioni verrà costruito un record escludendo la posizione che verrà appunto prevista dal modello. Per costruire un record a differenza del dataset precedente è sufficiente un solo smartphone posizionato o dalla parte del guidatore o dalla parte del passeggero ma non basta una sola curva ma almeno $1 + \lambda_N + \lambda_M$ curve. Quindi per avere una previsione bisogna attendere la fine di più curve con una latenza maggiore nel riconoscimento rispetto al dataset precedente.

Anche in questo caso per avere un dataset da usare con il classificatore binario (guidatore/passeggero) i record devono di classi diverse, siccome in questo caso si parla di curve, si è deciso di inserire ogni curva come due record

diversi, che sono due punti di vista differenti. Nel primo viene inserita la curva usato lo smartphone del guidatore, nel secondo viene usato lo smartphone del passeggero, avendo settato il campo della posizione alternato. Il dataset così costruito in totale ha circa 2160 record che sono classificati in questo modo: 50% guidatore e 50% passeggero.

Capitolo 4

Modello

In questo capitolo verrà descritto il modello della rete neurale costruito con Keras [15] usando Tensorflow [14] come backend. Comporre un modello in Keras è molto semplice la difficoltà sta nella scelta degli iperparametri più adatti al problema e dataset. Il problema che si vuole risolvere dato in input i dati dei sensori riconoscere se lo smartphone da cui provengono i dati si trova nella parte del guidatore oppure del passeggero, quindi è un problema di classificazione binaria perché ci sono due classi: guidatore e passeggero.

4.1 Costruzione del modello

Il modello costruito è di tipo sequenziale che è di default in Keras. Come descritto nella sezione 1.3.2 una ANN è composta di almeno tre livelli che verranno descritti successivamente.

4.1.1 Livelli

In Keras ci sono diversi tipi di livelli, ogni livello deve essere configurato a seconda del problema che si vuole risolvere. Dopo aver fatto diverse prove con più configurazioni di rete si è deciso di usare la struttura più semplice possibile in quando è la più adeguata a questo tipo di problema e la più

efficiente. Questa struttura è composta da tre livelli: livello di input, livello nascosto, livello di output, che sono illustrati nella figura 4.1.

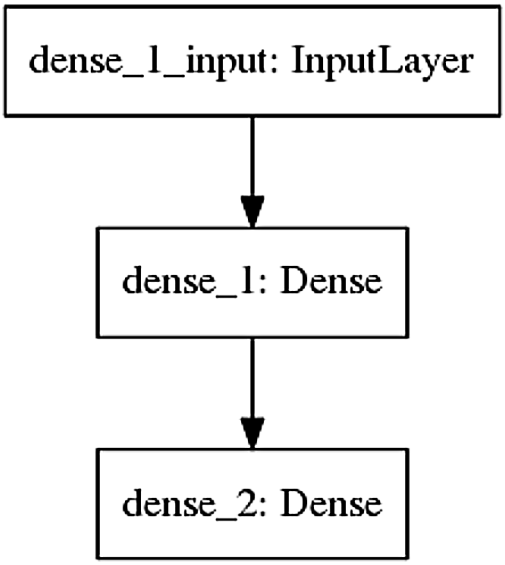


Figura 4.1: I tre livelli della rete neurale

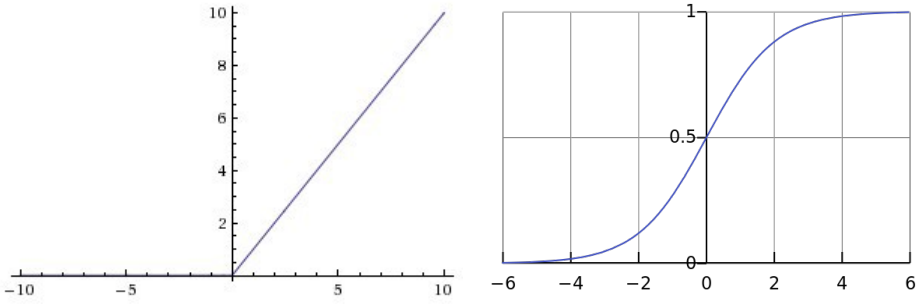
Per ogni livello sono state definite le seguenti configurazioni:

Tipo che definisce come sono collegati i neuroni tra di loro, si è scelto di usare il livello *denso* che è un classico livello delle ANN, è completamente connesso in cui ogni nodo di input è connesso ad ogni nodo di output.

Numero neuroni per livello, nel livello di input il numero è uguale alla cardinalità dell'input cioè al numero di caratteristiche di ogni istanza, nel modello creato questo viene definito dal dataset usato i due dataset creati hanno caratteristiche diverse. Nel livello nascosto viene definito dinamicamente a seconda del dataset. Il livello di output invece ha sempre un solo neurone che rappresenta la previsione.

Funzione di attivazione del nodo che viene specificata per ogni livello, tutti i nodi di un livello hanno la stessa funzione. La funzione definisce il comportamento del nodo. Nel livello di input che viene definito implicitamente in Keras, viene usata la funzione lineare di attivazione

cioè $f(x) = x$. Nel livello nascosto viene usata la funzione *relu* [18] che viene definita come $f(x) = max(0, x)$ mostrata nella figura 4.2a. Nel livello di output invece viene usata la funzione *sigmoid* definita come $f(x) = \frac{1}{1+e^{-x}}$ mostrata nella figura 4.2b che è una scelta classica per i classificatori binari.



(a) Relu (b) Sigmoid

Figura 4.2: Funzioni di attivazione usate Relu e Sigmoid

4.1.2 Compilazione del modello

Dopo aver definito il tipo di modello e i livelli, l'ultima fase della costruzione è la sua compilazione e questa fase richiede la scelta dei seguenti parametri: algoritmo di ottimizzazione, funzione di perdita e le metriche di valutazione usate dal modello nella fase di addestramento e test.

I parametri scelti per compilazione del modello sono:

Algoritmo di ottimizzazione viene usato dal modello per minimizzare la funzione $J(\theta)$ nei parametri del modello, è stato scelto *adam* (Adaptive Moment Estimation) [19] perché dopo aver fatto più prove è risultato il migliore.

Funzione di perdita (loss) è la funzione che definisce il costo dei parametri del modello, è stata scelta la *binary crossentropy* perché spesso viene usata nei problemi di classificazione binaria ed è risultata la migliore nei test.

Metriche di valutazione che vengono usate dal modello per valutare se stesso nella fase di addestramento e test, è stata scelta solo l'accuratezza.

4.2 Addestramento

Una volta che si ha il modello costruito e compilato si procede all'addestramento, una fase in cui bisogna specificare dei parametri e scegliere la porzione del dataset destinato a questa fase. La suddivisione dei dati nei due insiemi addestramento e test è molto importante perché da questo possono cambiare i risultati. Quando si usa la cross-validazione k-fold allora questa suddivisione viene fatta in automatico, ad esempio con 5 folders il rapporto sarebbe 20/80 test e addestramento. Tipicamente viene usata una percentuale maggiore per l'addestramento dai 60% in su e il resto per il test, dipende molto dalla dimensione del dataset. In questa fase è stato usato un rapporto di 20/80 e 5 k-fold cross-validazione.

L'addestramento è la fase che impiega più tempo di tutte le altre, a seconda dei parametri, numero di caratteristiche e la dimensione del dataset può impiegare da qualche minuto a dei giorni. Il numero delle epoche è un parametro molto importante in questa fase, che rappresenta il numero di iterazioni dell'algoritmo di ottimizzazione sui parametri del modello. Anche da questo dipende il tempo dell'addestramento e bisogno trovare il numero adatto per ogni modello e dataset, un valore troppo basso potrebbe causare a un modello poco addestrato e un numero troppo alto potrebbe peggiorare i risultati. Questo parametro è stato determinato con la tecnica della ricerca a griglia che verrà descritta nella sezione 4.4.

4.3 Valutazione

Per valutare il modello è stato usato il valore dell'accuratezza sia della fase di addestramento che la fase di test, l'ultimo è più importante ma è utile

sapere anche a quanto è arrivato nella fase di training perché se non supera il valore del test è un segno di qualche problema nei dati o nel modello. Quando si fanno più esperimenti come nel caso della cross-validazione, in cui viene fatta la media dei risultati, viene preso in considerazione anche la deviazione standard perché è importante capire la distribuzione che si è formata e non solo il valore della media.

Siccome in tutto il processo sono presenti alcuni comportamenti random, come l'ordine delle osservazioni nella fase di addestramento e i parametri della funzione di ottimizzazione nel inizializzati random ogni esperimento anche usando la cross-validazione, è stato fatto un algoritmo che esegue più volte (solitamente 10 volte) la cross-validazione per poi fare la media di tutte le iterazioni. In questo modo si cerca di dare un risultato più stabile possibile in quanto una sola iterazione potrebbe aver attribuito dei parametri random che fanno avvicinare o allontanare la funzione alla soluzione ottima cioè il minimo globale della funzione di perdita.

4.4 Ottimizzazione dei parametri

Tutti i parametri descritti nelle sezioni precedenti sono stati scelti con il metodo ricerca a griglia che consiste nel creare una griglia con i parametri che si vuole testare ad esempio più algoritmi di ottimizzazione o funzione di perdita. Il modello viene configurato con la combinazione dei parametri e dopo viene valutato con la tecnica della *K-fold cross validation* descritta nella sezione 1.3.4. Questa fase consiste di fare più prove con parametri diversi per capire quale sia l'abbinamento migliore per il dataset. Tutti gli esperimenti sono statti fatti con 5-fold cross-validazione e 5 ripetizioni come descritto nella sezione 4.3 per un totale di 25 ripetizioni per ogni esperimento.

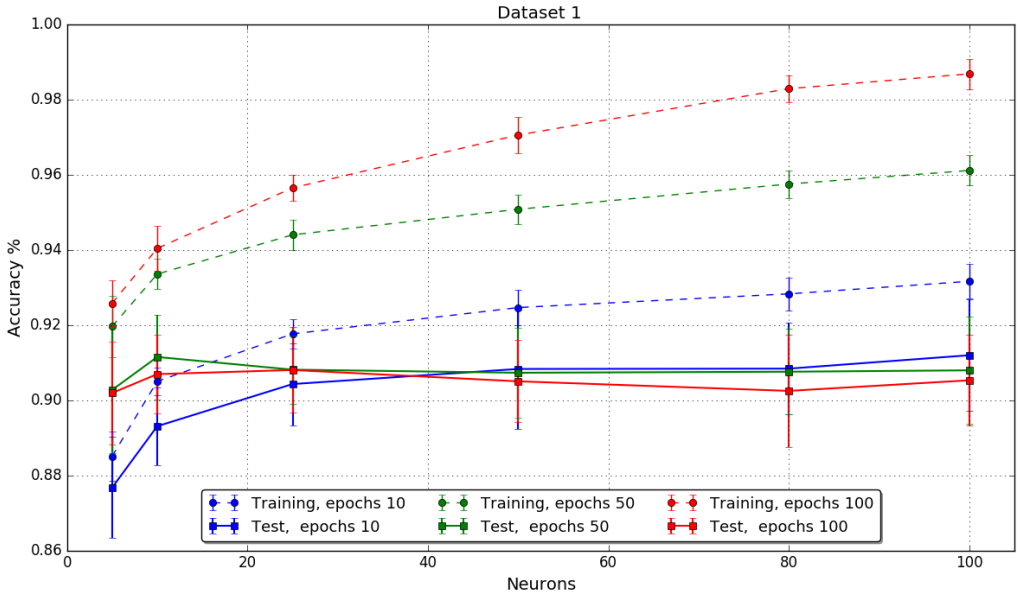
4.4.1 Ottimizzazione del numero di neuroni

L'analisi al variare dei numero di neuroni nel livello intermedio è utile per capire come configurare il modello, questa analisi viene fatta solo in fase

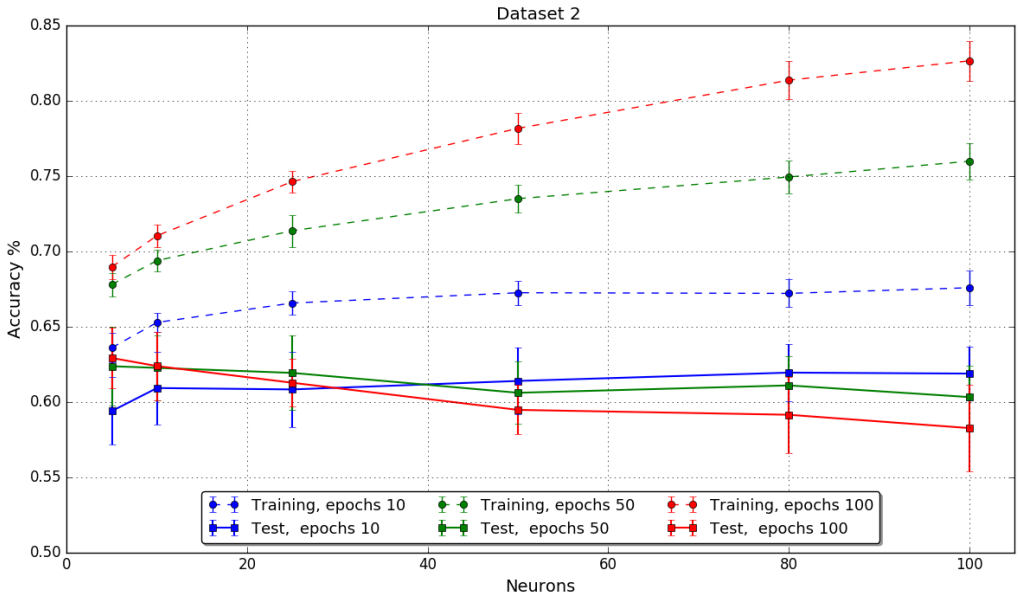
iniziale, una volta scelto il numero non è più necessaria. Possiamo notare in entrambi i grafici (figura 4.3) che all'aumentare del numero di neuroni la percentuale del training cresce in continuo invece il valore del test cresce solo fino a un certo punto e poi inizia la discesa. Questo comportamento è dovuto al fatto che con un numero maggiore di neuroni, la rete memorizza di più i dati di addestramento e quindi non riesce a generalizzare sui dati di test. E' interessante osservare come varia l'andamento all'aumentare dei neuroni considerando epoche diverse, notiamo in entrambi i casi lo stesso pattern ma su scala diversa. Mentre i risultati del training sono in crescita all'aumentare dei neuroni e delle epoche i risultati di test cambiano orientamento a da un certo punto in poi con un numero grande di epoche scendono. Considerando la situazione con 100 neuroni si ha che con poche epoche c'è meno differenza tra il training e test invece con il massimo numero di epoche 100 si ottiene una grande differenza quindi meno possibilità di generalizzare. Occorre scegliere il numero di neuroni che corrisponde alla maggiore accuratezza e minore deviazione che rappresenta un valore medio più stabile.

4.4.2 Analisi al variare delle epoche

Determinare il numero ottimale di epoche è molto importante in quanto incide moltissimo sul risultato, un numero troppo basso comporta ad una rete poco addestrata che ottiene dei risultati bassi sia nel training che nel test, un numero troppo elevato invece ottiene dei risultati molto buoni nel training perché impara molto i dati e un risultato troppo basso nel test. Questa analisi aiuta a trovare l'equilibrio giusto, si valutano i risultati al cambiare del numero di epoche con diversi numeri di neuroni trovati interessanti nella sezione precedente. Nella figura 4.4 sono illustrati i grafici di questa analisi, possiamo notare la conferma dell'affermazione detta sopra, dopo un certo livello i risultati di test iniziano a scendere. Si può notare inoltre che nella figura 4.4b si ripete lo stesso pattern visto nella sezione precedente cioè all'aumentare delle epoche e dei neuroni aumenta sempre di più la differenza tra il training e test. Invece nella figura 4.4a il pattern non si ripete perché



(a) Variando il numero di neuroni primo dataset



(b) Variando il numero di neuroni secondo dataset

Figura 4.3: Analisi al variare del numero di neuroni

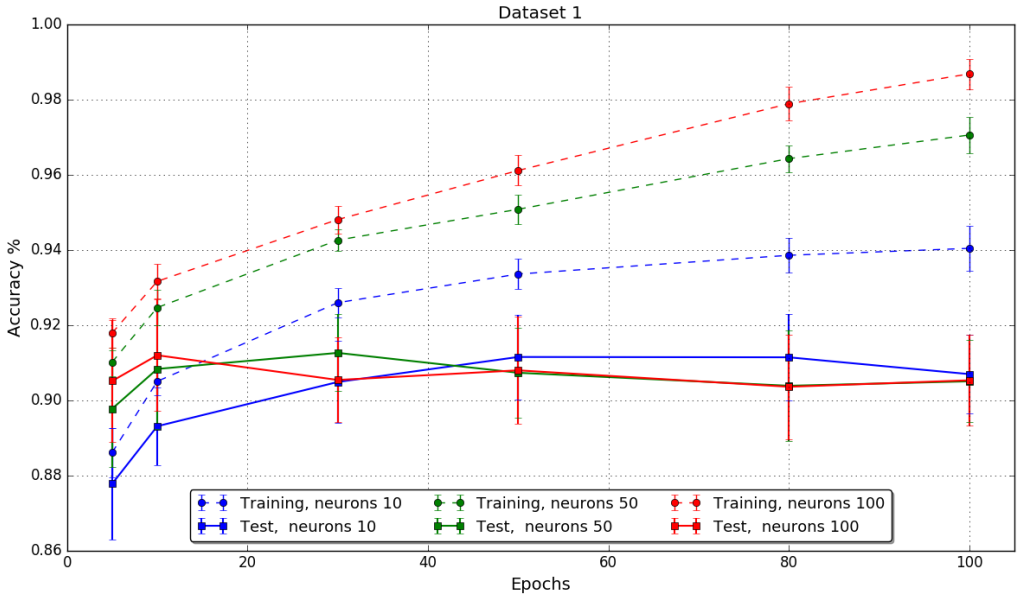
il comportamento si stabilizza dopo 50 epoche e ad arrivare a 100 tutti i test hanno lo stesso risultato. Nella figura 4.5 invece possiamo vedere come

evolve l'algoritmo di apprendimento all'interno di un'iterazione senza cross-validazione e ripetizioni, questo è quello che succede all'interno di ognuna delle 25 ripetizioni totali. Nella fase di ottimizzazione è molto importante visualizzare l'evoluzione delle metriche durante il processo per identificare subito la presenza di problemi. Dai grafici possiamo notare che da un certo punto in poi il l'accuratezza del test non cresce più ma rimane più o meno costante o nel caso del secondo modello diminuisce. E' interessante anche il fatto che con un numero alto di epoche i risultati sono meno stabili.

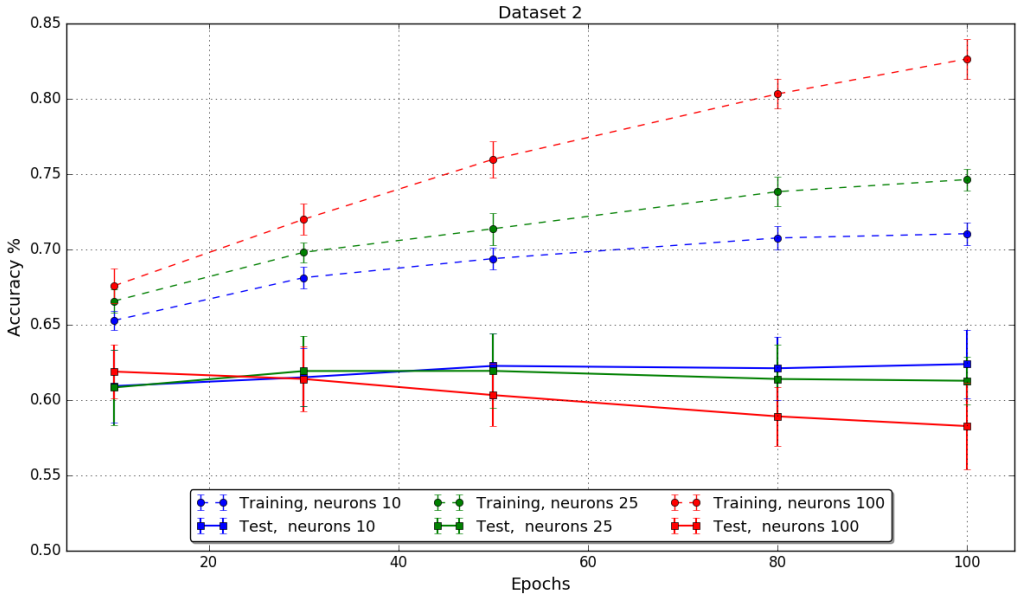
4.4.3 Scelta dei parametri

Dopo aver fatto l'analisi al variare dei neuroni e delle epoche si devono scegliere dei valori da assegnare per ogni dataset. Avendo due dataset diversi descritti nella sezione 3.6, il primo che usa i dati degli entrambi gli smartphone nello stesso record con meno caratteristiche e il secondo che ne usa solo uno ma che salva più curve risulta avere più caratteristiche. Il modello quindi dovrebbe risolvere due problemi simili ma comunque diversi perché l'input è molto differente, nel primo caso è abbastanza semplice in quanto ci sono 21 caratteristiche a differenza del secondo che ne ha al minimo 34 quando $N = M = 1$. La difficoltà dei problemi non sta solo nel numero di caratteristiche ma anche dalla sua natura, è più facile confrontare i valori degli smartphone che riguardano la medesima curva perché si ha la stessa velocità e ampiezza, è più complesso confrontare curve diverse che possono avere proprietà differenti. Pertanto i modelli risultanti sono diversi in cui varia il numero di neuroni nel livello di input (numero di caratteristiche) e nel livello nascosto. L'idea è che se il problema è più complesso potrebbero servire più neuroni per trovare una buona approssimazione ma bisogna stare attenti a non esagerare perché con un numero troppo elevato a rete potrebbe memorizzare gli esempi andare in *overfitting* e non generalizzare sui nuovi dati.

Quindi avendo fatto l'analisi descritta nelle sezioni precedenti si è scelto di i seguenti parametri che sono più adatti ai due dataset:



(a) Variando il numero di epoche primo dataset

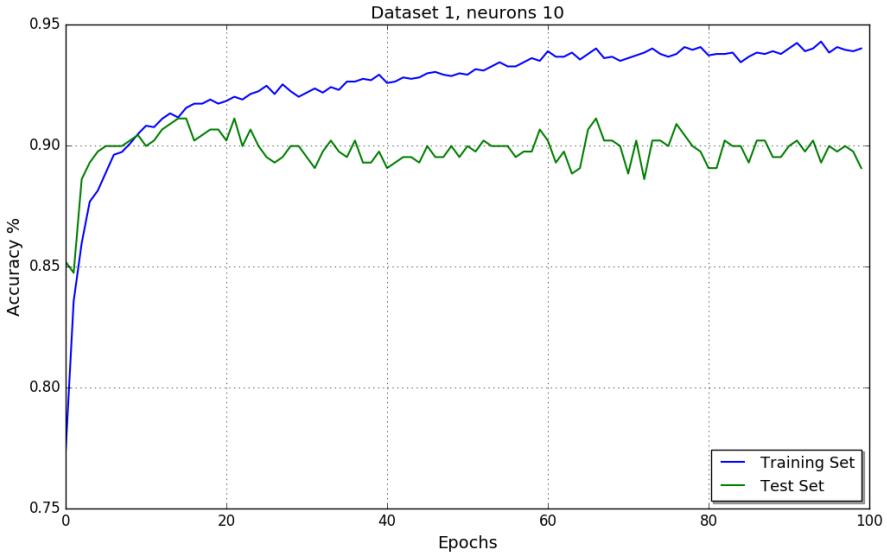


(b) Variando il numero di epoche secondo dataset

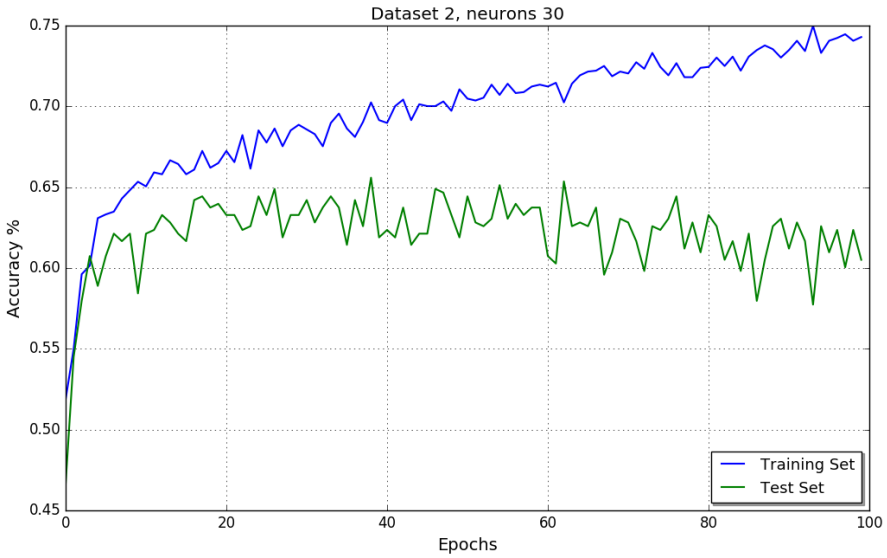
Figura 4.4: Analisi al variare del numero di epoche

Primo dataset 10 neuroni nel livello nascosto e 50 epoche.

Secondo dataset 25 neuroni nel livello nascosto e 50 epoche.



(a) Variando il numero di epoche primo dataset



(b) Variando il numero di epoche secondo dataset

Figura 4.5: Analisi al variare del numero di epoche nell'addestramento

4.4.4 Analisi al variare del numero di curve precedenti

Quando si usa il secondo dataset è interessante vedere come cambia il risultato in base al numero di curve precedenti N,M che si sceglie nel data-

set tenendo fissi i parametri scelti per questo dataset cioè 25 neuroni e 50 epoche. Per questa analisi sono stati generati 10 dataset diversi, con valori di N,M crescenti a partire da 1 fino ad arrivare a 10. Più si aumenta il numero più aumentano le caratteristiche perché vengono salvate più curve nello stesso record, anche in questo caso il risultato mostrato nella figura 4.6 conferma le aspettative. All'aumentare del numero di curve la rete inizia a memorizzare un percorso composto dalle curve precedenti quindi non riesce a generalizzare più di tanto. Il valore scende di poco all'aumentare delle curve prese in considerazione anche perché con un numero alto è possibile la variazione della tipologia di strada ad esempio da urbana a extraurbana. Un numero maggiore di curve vuol dire che prima di avere un risultato bisogna aspettare di percorrere più curve e quindi aumenta la latenza di riconoscimento. Aumenta anche il tempo di computazione della previsione in quanto all'aumentare delle curve aumentano le caratteristiche. Pertanto si è deciso di scegliere $N = M = 1$ valori usati per tutte le analisi successive.

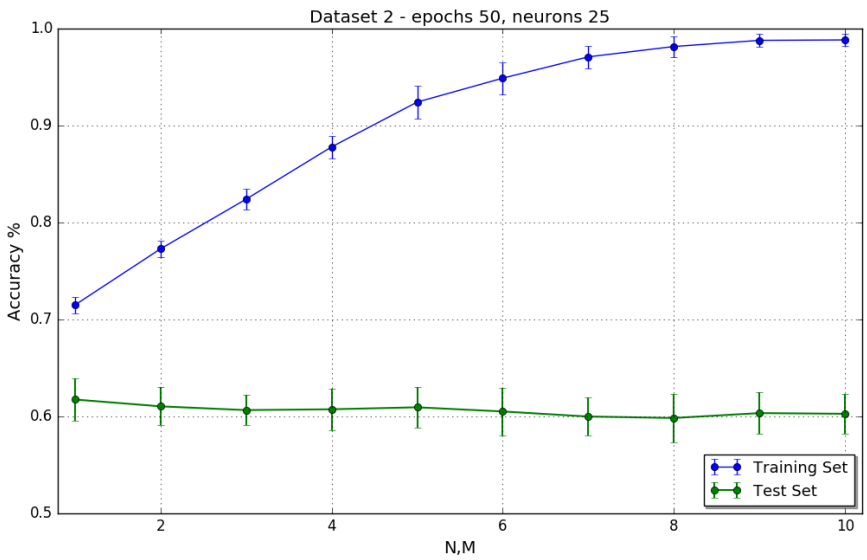


Figura 4.6: Analisi al variare del numero delle curve precedenti N e M

Capitolo 5

Analisi

In questo capitolo verranno fatte le analisi dei risultati per entrambi i dataset prendendo in considerazione percorsi reali e più curve consecutive. Per quanto riguarda il primo dataset verrà fatto anche un confronto con lo stato dell'arte descritto nella sezione 2.2.

5.1 Modalità delle analisi

In questa sezione verranno descritte le modalità di analisi che sono le stesse per entrambi i dataset, gli esperimenti sono statti fatti con 5-fold cross-validazione e 10 ripetizioni come descritto nella sezione 4.3 per un totale di 50 ripetizioni per ogni esperimento.

5.1.1 Selezione dei percorsi

Per analizzare i risultati nelle condizioni più reali viene fatta la selezione dei veri percorsi dai dataset, senza cambiare l'ordine delle curve. Si ricorda che con 5-fold cross-validazione ad ogni iterazione 20% dei dati viene lasciato da parte per il test, questi dati rappresentano le curve che compongono il percorso che verrà analizzato nello stesso ordine in cui sono stati registrati. L'ordine è importante quando si vogliono prendere in considerazione più curve perché una qualsiasi permutazione delle stesse non rappresenta la realtà

e.g. potrebbe capitare che una curva della strada urbana segua all'improvviso dopo una curva extraurbana senza passaggi intermedi. Questo serve anche per simulare il comportamento del modello su strada reale mentre il veicolo percorre in ordine una curva dopo l'altra senza saltare improvvisamente da un posto all'altro.

5.1.2 Le soglie

Le soglie (thresholds) servono per determinare quale probabilità deve essere considerata come accettabile dal sistema, questo è un parametro che si può lasciare a libera scelta dall'eventuale utilizzatore del sistema in futuro. Si possono configurare a seconda delle esigenze e del grado di sicurezza che si vuole ottenere dalla previsione. Si ricorda che l'output di una previsione è definita dal ultimo livello della rete neurale che ha come attivazione la funzione sigmoidea illustrata nella figura 4.2b, che restituisce una valore compreso tra 0 e 1 rappresentando la probabilità che il record passato abbia la posizione 1 cioè del guidatore. Nella fase di analisi è importante come viene trattata questa probabilità, si potrebbe dire che se è più grande del 50% allora è guidatore altrimenti passeggero, questo è una soglia netta che non prevede situazioni di incertezza e quindi restituisce sempre un risultato. Questo approccio potrebbe essere poco sicuro perché la probabilità del 51% è piuttosto bassa e quindi non troppo affidabile, in applicazioni critiche potrebbe non bastare. L'altra possibilità è di usare due soglie una sotto della quale consideriamo la posizione del passeggero e la seconda sopra della quale risulta guidatore e le probabilità che sono al centro diventano incertezze. Usando queste due soglie si riesce a configurare a piacere la sicurezza della risposta stando attenti agli incerti perché più si aumenta la distanza delle due soglie più aumentano il numero egli incerti. In uno scenario di riconoscimento in tempo reale usando queste soglie nella previsione si avrebbero tre risultati: guidatore, passeggero e incerto. Si nota che la soglia netta di cui si è parlato prima può essere espressa anche con due soglie entrambe impostate al 50%, nelle sezioni successive verrà mostrato il risultato al variare delle soglie.

5.1.3 Curve multiple

A seconda delle applicazioni in cui si vuole utilizzare questo riconoscimento può essere più utile valutare una curva oppure più curve consecutive. Nello stato dell'arte nella sezione 2.2 si è visto che la percentuale di riconoscimenti corretti cresce all'aumentare delle curve prese in considerazione, quindi nei casi in cui è molto importante l'accuratezza del risultato è necessario valutare più curve. In questo scenario si suppone di essere alla guida quindi si fanno le previsioni solo sui record dei guidatori, le curve vengono processate una dopo l'altra con il metodo descritto successivamente.

L'implementazione della valutazione delle curve multiple è semplice, dato n, il numero di curve che si vuole considerare, viene creato un buffer circolare che contiene le ultime n previsioni, viene fatta la media del buffer che viene confrontata con le due soglie e quindi si ottiene 1 se la percentuale supera la soglia del guidatore, 0 se la percentuale è inferiore alla soglia del passeggero e incerto altrimenti. Se la previsione risulta incerta allora viene contata per il numero degli incerti altrimenti viene inserito nella lista dei risultati il valore risultato dal confronto con le soglie: 1 o 0. Alla fine viene fatta la media di tutti i risultati e quindi risulta un'altra percentuale che rappresenta l'accuratezza del risultato visto che sono considerati solo i dati del guidatore nella previsione. Nelle sezioni successive verrà fatta l'analisi dei risultati al variare del numero di curve considerate.

5.2 Analisi del primo dataset

I risultati del primo dataset sono soddisfacenti perché sono in linea con quelli riscontrati nello stato dell'arte con cui verranno confrontati nella sezione 5.2.2. I risultati sono molto soddisfacente ma bisogna ricordare che questo scenario in cui vengono usati due dispositivi non è facilmente da implementare su larga scala quindi resta più utile per la ricerca che per l'applicazione.

5.2.1 Analisi al variare del numero di curve

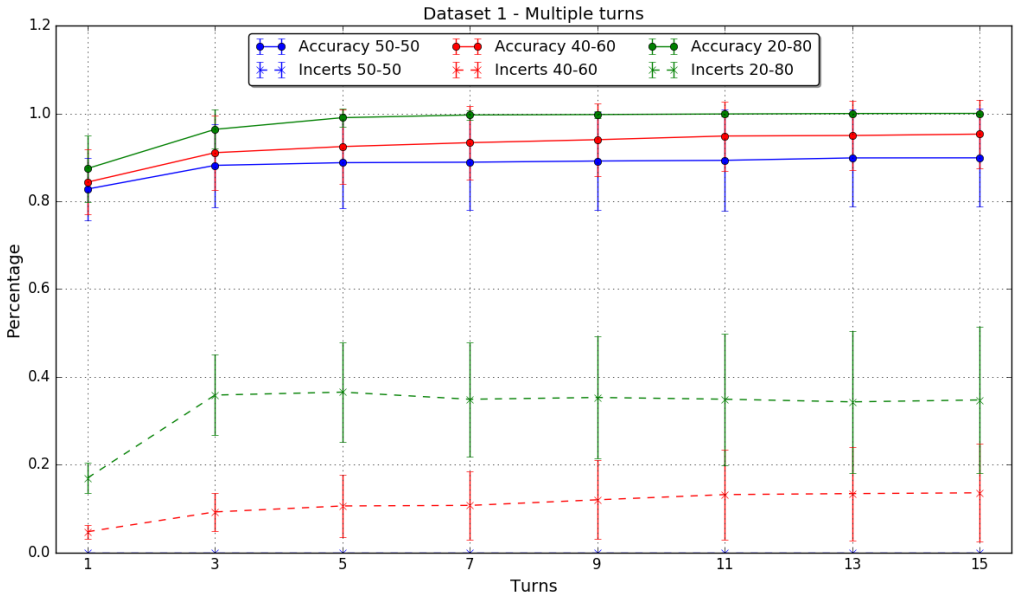


Figura 5.1: Dataset uno - curve multiple

L'analisi al variare del numero di curve è molto utile se si vuole pensare a un'applicazione di questo metodo perché il numero di curve è un fattore importante visto che da questo dipende la latenza di riconoscimento, un numero di curve maggiore corrisponde a un tempo di riconoscimento maggiore. Nella figura 5.1 è illustrato il grafico al cambiare delle curve da una a quindici curve per tre differenti soglie (passeggero-guidatore): 50-50, 40-60, 20-80. I risultati con una curva sono: 82,5% per le soglie 50-50, 84,3% per 40-60 e infine 87,5% per 20-80. Possiamo anche notare come alla crescita della distanza tra le due soglie cresce anche il numero di incerti, nel caso della soglia netta non ci sono incerti quindi il risultato riguarda tutte le curve. Nel caso delle due soglie la percentuale riguarda solo le curve in cui si è riuscito a dare una risposta e non su tutte, invece la percentuale degli incerti è rispetto a tutte le curve. Analizzando il cambiamento all'aumentare delle curve possiamo notare che già con tre curve si ha un miglioramento della percentuale che passa a 88%, 91% e 96,4% per 50-50, 40-60, 20-80 rispettivamente. Aumenta

anche il numero degli incerti però che arriva 9% per 40-60 e a 35,7% per 20-80, l'ultima è una percentuale importante, vuol dire che più di una volta su tre la previsione è incerta ma quando sa rispondere risponde con quasi la certezza. Considerando ancora più curve si ha che solo 20-80 a 99,5% con 7 curve, gli altri invece rimangono più o meno costanti sui risultati di 3 curve. Questa situazione probabilmente è dovuta al fatto che i casi di errata previsione sono distribuiti ugualmente nel percorso circa una curva ogni 10 viene riconosciuta male. Questi risultati sono molto soddisfacenti anche a confronto dello stato dell'arte che verrà fatto nella sezione successiva.

5.2.2 Confronto con lo stato dell'arte

Con i risultati del primo dataset è interessante fare un confronto con lo stato dell'arte descritto nella sezione 2.2. Purtroppo sono state pubblicati i risultati solo fino a 5 curve consecutive e non si hanno informazioni sulla varianza di questi dati e sui numeri degli incerti. Quindi il grafico della figura 5.2 non ha gli intervalli di confidenza e il numero degli incerti. Dal grafico si può notare che considerando una sola curva l'approccio delle reti neurali usato in questo lavoro supera lo stato dell'arte di poco più del 5% a seconda della soglia analizzata. La differenza sta nel tempo di latenza mentre l'algoritmo originale riesce a rispondere in tempo reale, il nostro approccio risponde solo dopo la fine della curva. Nel caso in cui vengono considerate più curve consecutive l'algoritmo originale supera le due soglie 50-50 e 40-60 e rimane sotto solo alla soglia più sicura del 20-80. Da questo grafico si nota la flessibilità dell'approccio proposto, cioè se si vuole una maggiore accuratezza si può scegliere una soglia con uno spread maggiore.

5.2.3 Analisi al variare delle soglie

Dall'analisi al variare del numero delle curve è emesso che a parte il numero delle curve anche le soglie scelte hanno un grande contributo sui risultati e sopratutto sul numero degli incerti. Quest'analisi viene fatta cambiando le

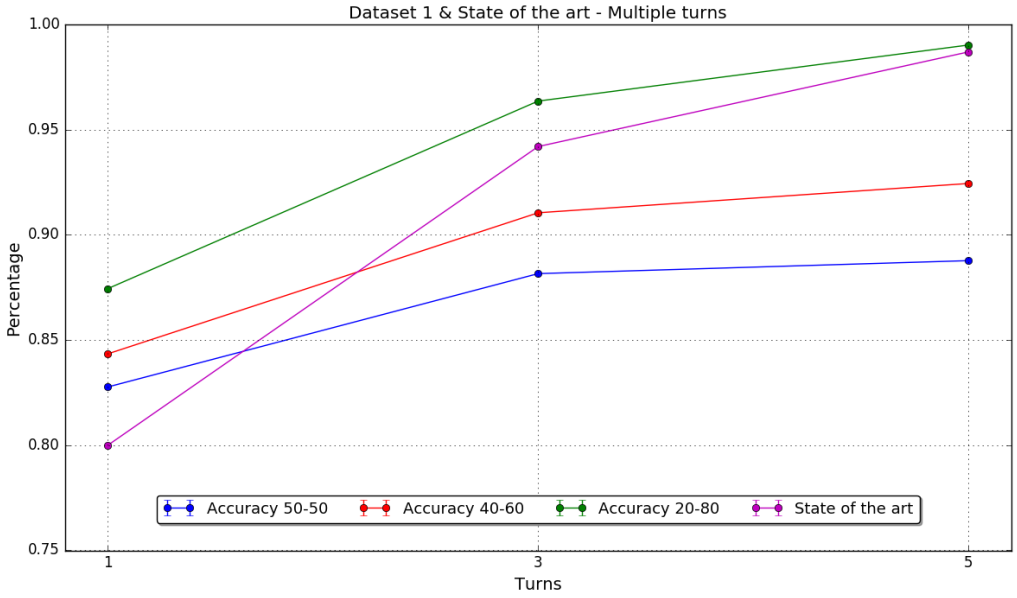


Figura 5.2: Dataset uno - confronto con lo stato dell'arte

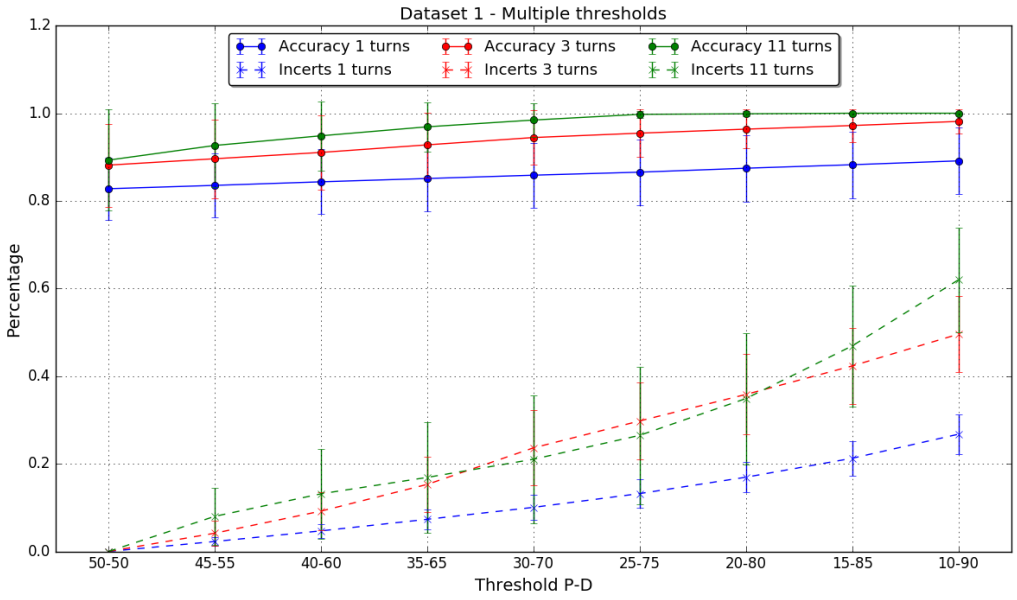


Figura 5.3: Dataset uno - variando le soglie

soglie che vanno da 50-50 a 10-90 con un passo di 5 in entrambe le direzioni, verranno valutati tre numeri di curve: 1, 3 e 11. Nel grafico 5.3 si può notare

come cambia la percentuale al variare delle soglie, possiamo notare l'aumento delle incertezze che arrivano a superare anche il 60% con 11 curve e soglie 10-90, mentre la percentuale con questi parametri arriva a 100% come anche 11 curve con le soglie 15-85 che rispetto alla precedente ha un'incertezza al 47%. Per scegliere delle soglie così alte in cui la metà dei casi sono incerti vuol dire che l'utilizzo di queste informazioni è molto critico per quell'applicazione. Se si vuole un numero di incerti minori si potrebbe usare la soglia 40-60 che ottiene comunque degli ottimi risultati con 3 e 11 curve.

5.2.4 Analisi al variare dell'accuratezza

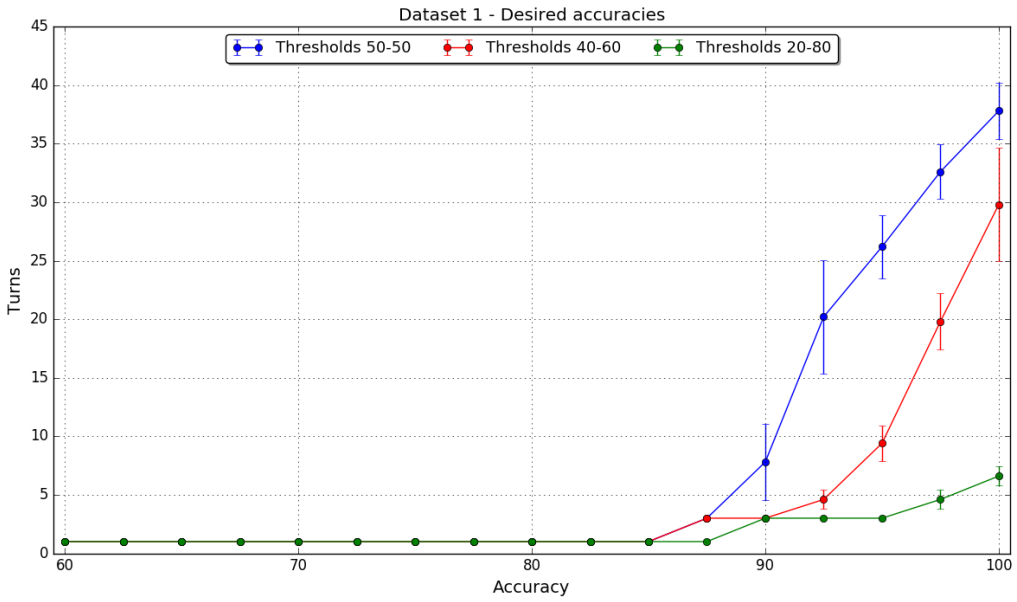


Figura 5.4: Dataset uno - variando l'accuratezza desiderata

Questa analisi è diversa dalle precedenti perché dispone i dati diversamente e si vuole dare un risultato diverso, cioè il numero curve necessarie per arrivare ad una certa percentuale di accuratezza desiderata. Nella figura 5.4 è illustrato il grafico che mostra le curve necessarie a variare della percentuale desiderata, e si può subito notare che fino al 85% bastata una curva in tutte le soglie. Per arrivare vicino al 100% ci vogliono 7 curve con le soglie

20-80 e molte di più con le soglie 40-60 che si arriva a 29 curve e 50-50 a 37 curve. Questo era un risultato prevedibile in quanto anche nelle altri grafici con le soglie 50-50 e 40-60 non si arrivava molto vicino alla certezza.

5.3 Analisi del secondo dataset

I risultati del secondo dataset sono diversi da quello del primo, come accennato prima il riconoscimento con uno smartphone unico è un problema più difficile e quindi ci si aspetta una percentuale di minore successi. In questo caso anche quando viene analizzata la situazione con una curva unica in realtà nel record sono inclusi i dati di almeno altre due curve una omogenea e l'altra disomogenea, quindi non si ha nessun risultato prima di avere questi dati. Quindi l'analisi parte già dal primo record completo che include le due curve precedenti. Questi risultati dimostrano che l'approccio di riconoscimento con un solo smartphone è fattibile e anche implementabile su larga scala visto che non ha bisogno di altri dispositivi ma di solo uno smartphone.

5.3.1 Analisi al variare del numero di curve

Nella figura 5.5 è illustrato il grafico al variare delle curve, si può subito notare come a confronto del primo dataset le percentuali sono più basse sopratutto quelle con un numero basso di curve. Prendendo in considerazione il caso della lettura istantanea cioè con una unica curva si hanno i seguenti risultati: 68,8% per la soglia netta, 73,7% per le soglie 40-60 e 80,5% per le soglie 20-80. Considerando che sono previsioni fatte con un unico dispositivo si può dire che sono molto soddisfacenti, variano di poco più di 10% rispetto agli stessi dati con il primo dataset che usava due smartphone. Un fattore da non trascurare è il numero degli incerto che con 20-80 è altissimo supera il 75% questo vuol dire che l'accuratezza rappresenta solo il 25% delle curve e gli incerti arrivano a più di 90% già da 5 curve quindi l'utilizzo di queste soglie è difficile da immaginare. Con le soglie 40-60 invece l'incertezza arriva fino al 50% con 9 curve, che sono la metà ma comunque più bassa delle soglie

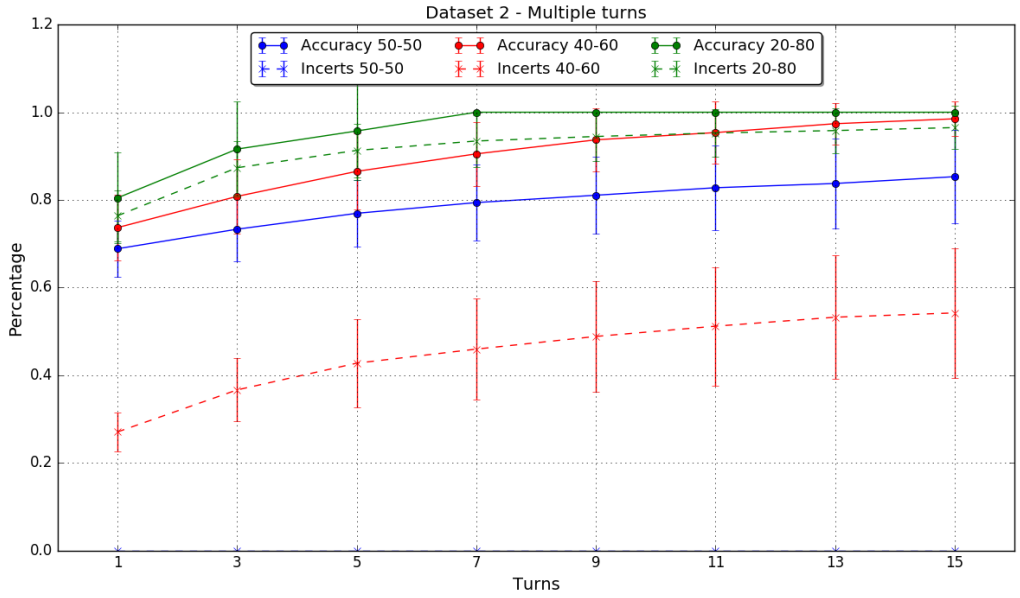


Figura 5.5: Dataset due - curve multiple

20-80. Possiamo notare che con 7 curve l'accuratezza arriva al 100% per 20-80 e a 90% per 40-60 mentre per 50-50 arriva al 79%. Con 15 curve invece le soglie 40-60 arrivano 98,5% con l'incertezza al 54%. Il numero di curve relativamente alto come 15 potrebbe sembrare difficile da raggiungere o comunque comporta ad una latenza di riconoscimento molto alta ma nell'ottica di riconoscere un percorso è un numero abbastanza basso. Nell'indagine fatta dalla Fiat [20] su 400.000 percorsi in cinque paesi europei si è determinato che la maggioranza dei percorsi in auto vengono sono mediamente di 10km una lunghezza ragionevole per arrivare alle curve necessarie senza prendere in considerazione che spesso per arrivare alla strada uscendo dal parcheggio si fanno diverse svolte.

5.3.2 Analisi al variare delle soglie

Variando le soglie in questo caso la prima osservazione che possiamo fare osservando il grafico della figura 5.6 è che la percentuale di incertezza cresce fino ad arrivare molto vicina a 100% con le soglie 10-90 e più in generale con

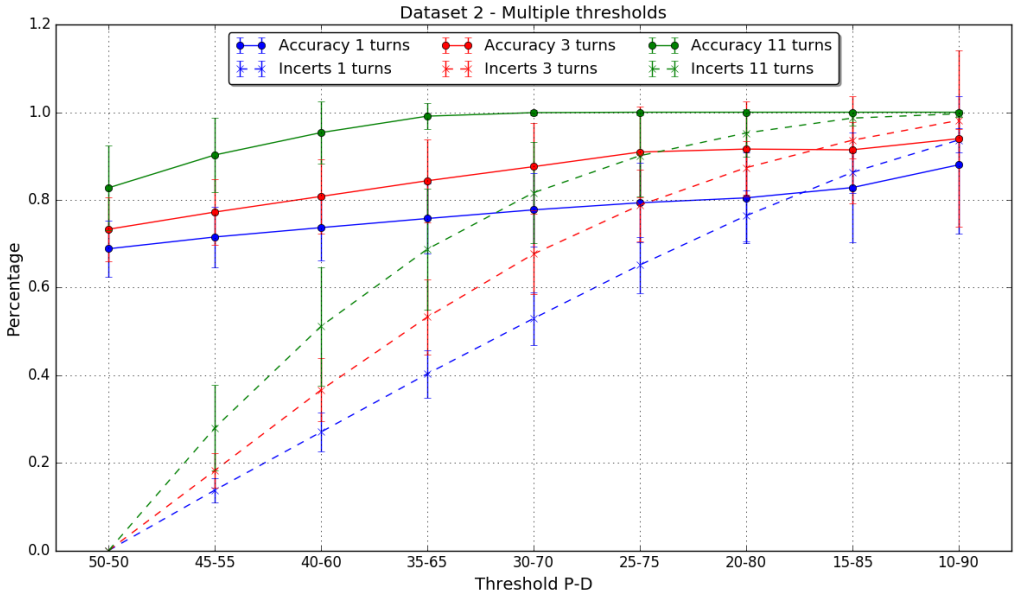


Figura 5.6: Dataset due - variando le soglie

soglie da 30-70 ad arrivare a 10-90 le incertezze in tutti i casi superano il 50%. Questa visione è molto utile per capire poi effettivamente quali soglie possono essere usate nelle varie applicazioni, per quanto riguarda la percentuale di accuratezza si arriva al 100% con 30-70 e 11 curve, e da 20-80 a salire tutte le curve sono sopra al 80%. Quindi da questo scenario si può ipotizzare che le soglie più adeguate sono 50-50, 45-55 e 40-60 anche se ci possono essere delle eccezioni a seconda del campo applicativo. Un motivo per la grande incertezza con soglie con spread maggiore sta nel valore istantaneo di una curva essendo mediamente intorno al 70% è difficile trovare delle curve che possano far parte ad esempio di un intervallo 20-80 perché la maggioranza è all'interno di questo intervallo quindi sono incerti.

5.3.3 Analisi al variare dell'accuratezza

Variando l'accuratezza desiderata si può vedere nella figura 5.7 quante curve sono necessarie per raggiungerla con tre soglie diverse: 50-50, 40-60 e 20-80. Questo grafico non rappresenta gli incerti in quanto mostra le curve

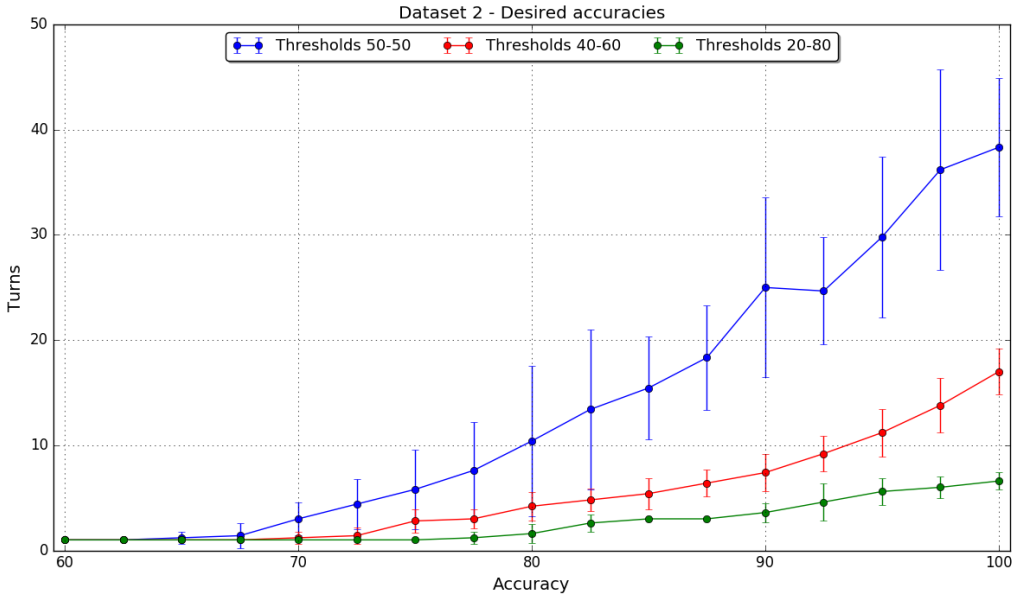


Figura 5.7: Dataset due - variando l'accuratezza desiderata

necessarie, le incertezze sono state analizzate nella sezione precedente. Possiamo notare che per avere una percentuale del 70 bastano tre curve in tutti i casi, per percentuali più basse basta anche una sola curva. Si può notare come la soglia netta cresce più rapidamente rispetto alle altre questo è dato dal fatto che non ci sono incertezze e per arrivare a 80% mediamente servono servono 11 curve, per 90% 25 e per 100% 39 curve. Le altre due sogli hanno un comportamento simile fino al 90%, infatti per arrivare al 80% servono 4 curve per le soglie 40-60 e 2 curve per 20-80. Per arrivare al 90% invece bastano 4 curve per 20-80 e 7 curve per 40-60. La situazione cambia per il 100% dove si crea un distacco maggiore, per le soglie 20-80 servono 7 curve invece per 40-60 ne servono 17. Si ricorda che questi dati riguardano solo i casi in cui il sistema riesce a prevedere un risultato e sono escluse le incertezze.

Conclusione e sviluppi futuri

In questo lavoro di tesi si è studiata la fattibilità di un nuovo metodo per riconoscere automaticamente l'uso dello smartphone alla guida basato sull'apprendimento automatico implementato con le reti neurali artificiali. L'approccio proposto è in grado di funzionare sia con due dispositivi che con un unico dispositivo. I risultati sono soddisfacenti in quanto con due dispositivi dopo una curva si supera la percentuale di riconoscimento dello stato dell'arte e con un unico dispositivo dopo qualche curva consecutiva si riesce a superare la soglia del 80% di riconoscimenti corretti. Per fare questa ricerca sono stati raccolti i dati dei sensori inerziali di tre posizioni nel veicolo su strade urbane ed extraurbane, i dataset prodotti possono essere utilizzati per altre ricerche analoghe e non solo.

Questo è il primo lavoro che ha cercato di risolvere il problema con le reti neurali artificiali e usando un unico dispositivo, pertanto ci sono ancora molti scenari di miglioramenti e sviluppi futuri, alcuni di loro potrebbero essere i seguenti:

- Estendere il dataset raccogliendo dati da veicoli, dispositivi e aree geografiche diverse, questo permetterebbe di avere dati più rappresentativi della realtà che se utilizzati come in questo lavoro potrebbero migliorare i risultati.

- Implementare un'applicazione di raccolta dati multipiattaforma in grado di funzionare su più dispositivi e sistemi operativi diversi.

- Implementare un sistema automatico di sincronizzazione delle tracce che è essenziale nell'ottica di avere molte più tracce provenienti da diversi veicoli.
- Implementare l'allineamento di coordinate per poter effettuare il riconoscimento a prescindere dall'inclinazione dello smartphone.
- Creare altri modelli di apprendimento automatico e fare il confronto tra loro e individuare il migliore.
- Implementare un'applicazione mobile che utilizza il modello proposto per fare il riconoscimento dell'uso dello smartphone alla guida in tempo reale.

Bibliografia

[1] WHO, Road traffic injuries [Online], http://www.who.int/mediacentre/factsheets/fs358/en/

[2] Distracted Driving [Online], https://www.nhtsa.gov/risky-driving/distracted-driving

[3] Smartphone users worldwide 2014-2020 [Online], https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[4] Smartphones may be to blame for unprecedented spike in pedestrian deaths [Online], http://money.cnn.com/2017/03/30/technology/pedestrian-safety-smartphones/index.html

[5] Chatti mentre guidi? L'assicurazione è più cara [Online], http://www.financialounge.com/azienda/financialounge/news/guida-cellulare-assicurazione/

[6] Y. Wang, Y. J. Chen, J. Yang, M. Gruteser, R. P. Martin, H. Liu, L. Liu, C. Karatas. Determining Driver Phone Use by Exploiting Smartphone Integrated Sensors. IEEE Transactions on Mobile Computing (Volume: 15, Issue: 8, Aug. 1 2016) p. 1965 - 1981

[7] D. A. Johnson, M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform" in Intelligent Transportation Systems (ITSC), 2011.

[8] C, Karatas, L, Liu, H. Li et al. Leveraging wearables for steering and driver tracking. INFOCOM 2016, IEEE.

[9] GSMArena.com - The ultimate resource for GSM handset information [Online], https://www.gsmarena.com

[10] Luca Bedogni, Marco Di Felice and Luciano Bononi. Context-aware Android applications through transportation mode detection techniques. Wireless Communications and Mobile Computing 16:2523?2541, 2016.

[11] J. S. Hickman and E. S. Geller, "Self-management to increase safe driving among short-haul truck drivers." Journal of Organizational Behavior Management, 2005.

[12] U. Fugiglando, P. Santi, S. Milardo, K. Abida, C. Ratti. Characterizing the "Driver DNA" Through CAN Bus Data Analysis. CarSys '17 Pages 37-41, 2017.

[13] Mitchell, T. (1997). Machine Learning. McGraw Hill. p. 2. ISBN 0-07-042807-7.

[14] An open-source machine learning framework for everyone [Online], https://www.tensorflow.org

[15] Keras: The Python Deep Learning library [Online], https://keras.io/

[16] Freedman, Ryan (2017): Smartphone recorded driving sensor data: Indianapolis International Airport to Urbana, IL. University of Illinois at Urbana-Champaign. https://doi.org/10.13012/B2IDB-4650469_V1

[17] Ferreira J Júnior, Carvalho E, Ferreira BV, de Souza C, Suhara Y, Pentland A, et al. (2017) Driver behavior profiling: An investigation with different smartphone sensors and machine learning. PLoS ONE 12(4): e0174959. https://doi.org/10.1371/journal.pone.0174959

[18] Rectifier (neural networks), Wikipedia [Online], https://en.wikipedia.org/wiki/Rectifier_(neural_networks)

[19] Kingma, D. P., Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. International Conference on Learning Representations, p. 1 - 13

[20] The eco:Drive "White Paper" [Online], http://www.fiatpress.com/press/detail/11164

Ringraziamenti

Vorrei ringraziare di cuore mia moglie *Alina* per avermi supportato e sopportato in tutto questo percorso di studi e soprattutto durante questo lavoro di tesi. Ringrazio mia figlia *Noemi* che nel suo primo anno di vita ha sempre fatto la brava e dormito di notte. Ringrazio i miei genitori *Galia*, *Efim* e mia sorella *Elvira* per il sostegno che mi hanno dato, sempre, anche nei momenti più difficili. Ringrazio il relatore di questa tesi il *Dott. Luca Bedogni* per avermi proposto questo tema di ricerca utile e interessante e per avermi indirizzato e consigliato su come procedere al meglio. Ringrazio il *Prof. Lorenzo Donatiello* per i suoi preziosi consigli senza dei quali probabilmente non avrei concluso questo percorso di studi. Ringrazio la *Prof.ssa Monica Palmirani* e *Dott. Luca Cervone* che nel lavoro hanno dato importanza ai miei impegni universitari e familiari. Ringrazio *Ameba* e *Michele* per aver condiviso con me una buona parte delle esperienze universitarie. E infine l'ultimo ma non in ordine di importanza ringrazio il Creatore, senza la Sua volontà non esisterebbero tutte le belle persone sopra menzionate.