Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Una applicazione context aware per il riconoscimento del tipo di mobilità

Relatore:
Chiar.mo Prof.
Luciano Bononi

Correlatori:
Dott.
Luca Bedogni

Dott.
Marco Di Felice

Presentata da:
Andrea Jonus

Sessione II
Anno Accademico 2013/2014

Indice

Capitolo 1

Introduzione

Il tema affrontato da questa tesi è lo sviluppo di applicazioni context aware per dispositivi mobili, ovvero programmi in grado di riconoscere il contesto di un dispositivo in un determinato momento. In particolare, il mio scopo principale è stato la creazione di un servizio di riconoscimento del tipo di mobilità dell'utente, utilizzabile anche da altri sviluppatori. Questo obiettivo è stato raggiunto producendo una libreria Java per dispositivi Android e una applicazione che sfrutta tale libreria per fornire dati sul tipo di mobilità ad altri programmi del dispositivo. Si è voluto anche fornire un esempio concreto di come si potrebbe impiegare questo sistema, producendo una applicazione che consuma tali informazioni per eseguire operazioni utili all'utente. Infine, sono state prodotte statistiche sull'utilizzo di questo sistema per rilevarne l'efficienza e i consumi, e per stabilirne gli effettivi vantaggi e svantaggi rispetto a metodi già esistenti.
Le tecniche impiegate per la realizzazione di questo progetto sono tratte da una ricerca che ha come autori i dottori Luca Bedogni e Marco Di Felice e il professore Luciano Bononi. Questa tesi vuole essere una dimostrazione pratica dei principi espressi in tale ricerca.

Capitolo 2

Teoria

2.1 Stato dell'arte

Il riconoscimento del contesto in cui si trova un dispositivo in un determinato momento non è un problema nuovo nel campo delle applicazioni mobili. La crescita esplosiva del numero di dispositivi mobili disponibili sul mercato ha portato alla necessità di fornire informazioni sempre più dettagliate alle applicazioni presenti su di essi, per l'automatizzazione di operazioni comuni. L'acquisizione e l'elaborazione di dati riguardanti la posizione dell'utente, in particolare, è stata resa triviale da hardware dedicato (tecnologia GPS). Un caso diverso è quello del riconoscimento del tipo di mobilità dell'utente: esistono diverse ricerche su questo tema, e sono già stati depositati brevetti che descrivono possibili tecnologie (ad esempio [8]), tuttavia non esistono soluzioni standard per affrontare questo problema. Distinguere diversi tipi di veicolo non è un compito che coinvolge un solo tipo di dato, e potrebbe non essere risolvibile utilizzando solo tecniche già in uso nelle applicazioni attuali. Un approccio possibile può essere quello di sfruttare le funzionalità di geolocalizzazione del dispositivo. Un esempio degno di nota è il servizio di Activity Recognition [4] del framework di Android, disponibile agli sviluppatori già dal 2013. Tuttavia, queste soluzioni possono non essere ideali per l'utente o per il programmatore in determinati casi. In particolare, si

vorrebbe evitare l'uso della tecnologia GPS in situazioni di mancanza di connettività o quando il risparmio energetico è un fattore critico. Inoltre, nel caso di Activity Recognition, il riconoscimento è limitato ad un set chiuso di azioni, e non discrimina fra mezzi di trasporto di vario genere.

2.2 By Train Or By Car

Nel Novembre del 2012 è stato pubblicato un paper di ricerca universitaria dal titolo "By train or by car? Detecting the user's motion type through smartphone sensors data" [1]. In tale paper viene descritto un nuovo metodo per rilevare il tipo di mobilità dell'utente attraverso i dati forniti da alcuni sensori del dispositivo, senza sfruttare le funzionalità di geolocalizzazione e permettendo di distinguere diversi mezzi di trasporto. Esso si basa sull'elaborazione dei dati forniti da accelerometro e giroscopio del dispositivo per produrre particolari set di valori che possono poi essere associati ad un tipo di mobilità specifico, in modo da ottenere dei pattern di movimento utilizzabili per confronti successivi. La sequenza delle azioni da eseguire può essere riassunta in questo modo:

1. Accelerometro e giroscopio vengono monitorati costantemente. Dopo un certo numero di millisecondi, i valori correnti di entrambi i sensori vengono utilizzati per il calcolo di un valore, definito "magnitudo", ottenuto con la seguente operazione:

$$magnitude(s) = |v_s| = \sqrt{v_{x,s}^2 + v_{y,s}^2 + v_{z,s}^2}$$

   dove $v_{x,s}$, $v_{y,s}$ e $v_{z,s}$ sono rispettivamente i dati rilevati per gli assi x,y e z del sensore s. La creazione di questo valore è giustificata dalla necessità di impiegare una metrica indipendente dall'orientamento del dispositivo.
   Le magnitudo prodotte periodicamente vengono salvate per l'operazione seguente.

2. Dopo alcuni secondi, le magnitudo registrate vengono analizzate e vengono generati due set di feature definiti nel seguente modo:

$$F_k^s = \{min_{k,s}, max_{k,s}, avg_{k,s}, std_{k,s}\}$$

dove $min_{k,s}$, $max_{k,s}$, $avg_{k,s}$ e $std_{k,s}$ rappresentano il minimo, il massimo, la media e la deviazione standard delle magnitudo registrate nel periodo k per il sensore s. Questi set vengono quindi associati al tipo di mobilità corrispondente al periodo k in cui sono state effettuate le misurazioni. La tripla $\langle F_k^A, F_k^G, m_k \rangle$ risultante, con A e G corrispondenti ad accelerometro e giroscopio e m_k al tipo di mobilità, viene quindi salvata.

3. L'insieme di tutte queste triple viene utilizzato come input per il training di un classificatore, utilizzando un algoritmo di machine learning opportuno. Le analisi effettuate nella ricerca portano alla conclusione che l'algoritmo Random Forest [2] ha prestazioni migliori rispetto agli altri algoritmi analizzati (SVM [3] e Naive Bayes [6]) dal punto di vista dell'accuratezza nelle classificazioni.

4. Il classificatore viene poi utilizzato per definire il tipo di mobilità m_k in nuove triple costruite seguendo lo stesso procedimento dei passaggi 1 e 2.

Le analisi effettuate in [1] indicano come valori ottimali per la frequenza di raccolta delle magnitudo e per la durata dei periodi di raccolta rispettivamente 5 Hz e 5 secondi. Inoltre, è stato osservato che si ottiene una maggiore precisione nelle classificazioni quando training e riconoscimento sono effettuati usando la stessa frequenza. Per maggiori dettagli sul procedimento appena descritto e sull'analisi dei risultati ottenuti si rimanda al paper di ricerca.
La precisione del riconoscimento attraverso questo metodo ovviamente non raggiunge il 100%, e ci si aspetta di incontrare casi dove il tipo di mobilità viene classificato erroneamente. Per ovviare al problema di sporadiche valutazioni errate, [1] suggerisce di considerare come tipo di mobilità effettivo

quello più ricorrente all'interno di un certo numero di classificazioni recenti, definito "storico". Questo approccio purtroppo aumenta anche il tempo richiesto dal dispositivo per reagire a cambiamenti di mezzo di trasporto.
Infine, viene descritta una possibile applicazione per dispositivi Android che sfrutti le tecniche appena descritte per eseguire operazioni sul dispositivo, ad esempio attivare o disattivare alcune funzionalità, a seconda del tipo di mobilità e delle preferenze dell'utente. Un'altra feature possibile è la condivisione delle classificazioni generate da essa con altre applicazioni attraverso un Content Provider, per consentirne la trasmissione ad altri dispositivi o l'utilizzo per operazioni differenti.

Capitolo 3

Progettazione

Nel Capitolo 2 si è potuto vedere come [1] fornisca delle indicazioni molto precise sulla metodologia da seguire per realizzare uno strumento di riconoscimento del tipo di mobilità. Inoltre, viene descritta nel dettaglio una possibile applicazione che sfrutti questa tecnologia per fornire un servizio di gestione delle funzionalità del dispositivo.
Tuttavia è stato necessario un lavoro di progettazione più dettagliato per stabilire tutte le caratteristiche del prodotto finito e gli strumenti da impiegare per la sua realizzazione. Parte di questi sono stati definiti durante il lavoro svolto per il corso di Laboratorio di Applicazioni Mobili, ma in seguito è stato necessario effettuare aggiunte e modifiche alle caratteristiche stabilite inizialmente. Nelle prossime sezioni sono illustrati i dettagli della progettazione per la versione finale del lavoro per questa tesi.

3.1 Caratteristiche e Requisiti

Parte delle caratteristiche del progetto sono derivate direttamente dalla descrizione dell'applicazione di prova in [1], dunque il loro soddisfacimento è stato considerato prioritario fin dall'inizio. L'applicazione consegnata per il corso di Laboratorio di Applicazioni Mobili si è dunque basata su questi requisiti fondamentali:

- L'applicazione deve essere sviluppata per il sistema operativo Android.
- L'applicazione deve fornire un servizio di riconoscimento del tipo di mobilità, basandosi sulle tecniche descritte in [1].
- Il riconoscimento deve supportare tre tipi di mobilità diversi: automobile, treno, a piedi.
- L'applicazione deve permettere di effettuare misurazioni di training per i tre tipi di mobilità supportati.
- Le classificazioni prodotte devono essere registrate in un Content Provider.
- L'applicazione deve permettere la gestione di profili associati a ciascun veicolo riconosciuto. Ogni profilo conterrà lo stato (attivo o inattivo) delle funzionalità di vivavoce, connessione Wi-Fi e connessione Bluetooth, da applicare al riconoscimento del veicolo corrispondente.

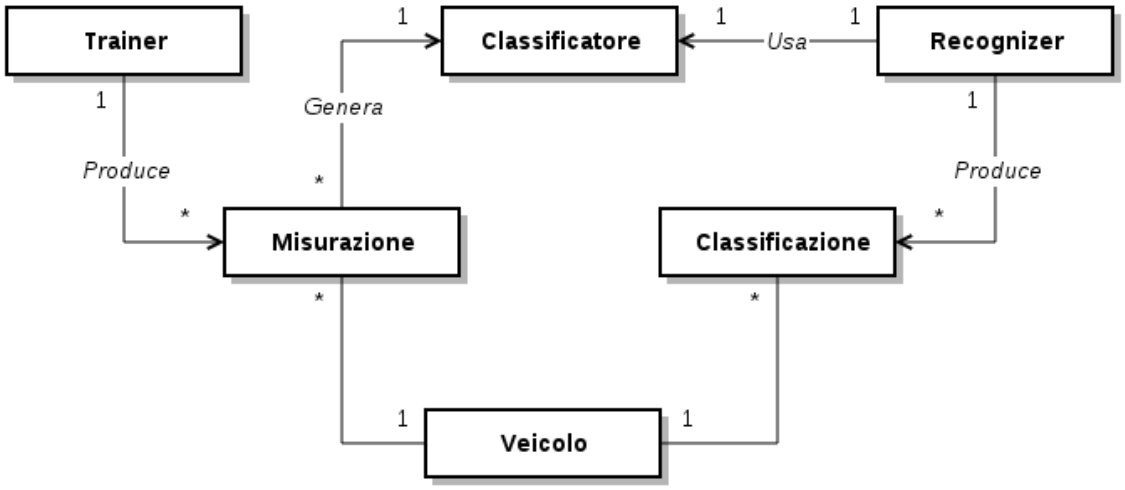
Una volta creata questa prima applicazione, sono stati discussi diversi requisiti aggiuntivi, che hanno portato a cambiamenti radicali nella struttura del progetto:

- Il progetto deve essere diviso in due applicazioni: la prima deve gestire le funzionalità di riconoscimento e condivisione delle classificazioni, la seconda deve gestire i profili definiti dall'utente e lo stato del dispositivo.
- Dev'essere possibile definire una lista di transizioni fra diversi tipi di mobilità da ignorare.
- Le funzionalità di riconoscimento devono essere raccolte in un insieme di classi indipendente dal resto del progetto, in modo da poter produrre una libreria Android per il riconoscimento del tipo di mobilità.
- L'utente può modificare la lista dei veicoli riconosciuti dall'applicazione, aggiungendo o rimuovendo elementi.

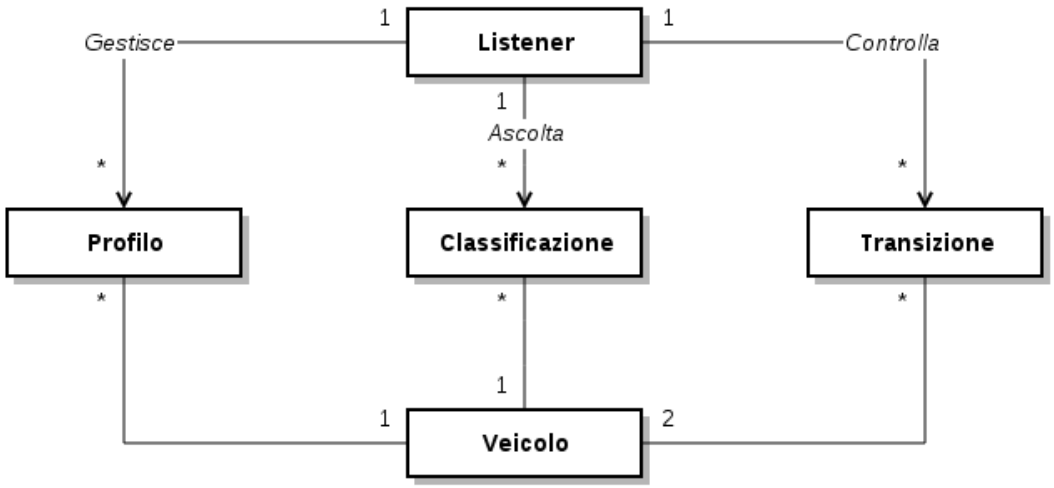
- L'utente può visualizzare le classificazioni prodotte dall'applicazione di riconoscimento.

La separazione del progetto in due applicazioni non è casuale: si è voluto infatti produrre un servizio specializzato nel riconoscimento del tipo di mobilità. Questo faciliterà la futura creazione di software che sfrutti le classificazioni fornite da esso.
Anche l'aggiunta della feature di personalizzazione dei veicoli supportati è fondamentale. Le statistiche prodotte in [1] suggeriscono infatti che sia possibile applicare questo sistema di riconoscimento ad un set arbitrario di veicoli, con la condizione che abbiano pattern di movimento differenti. Sarà importante stabilire quali siano i limiti nella distinzione di due veicoli con pattern simili.

La somma dei requisiti elencati sopra ha permesso di stabilire una struttura generale del progetto, illustrata dalla figura 3.1.



(a) Recognizer



(b) Profiler

Figura 3.1 Modelli di dominio per le applicazioni

3.2 Strumenti

3.2.1 Piattaforma

Per la realizzazione di questo progetto si è scelto di sfruttare la piattaforma Android. In particolare, le applicazioni sono state sviluppate per la versione 3.0 o più recente di questa piattaforma. Esse sono quindi compatibili con più dell'80% dei dispositivi Android disponibili sul mercato [5]. La scelta del sistema operativo è stata influenzata dalla necessità di impiegare lo strumento descritto nella prossima sezione, disponibile solo per il linguaggio di programmazione Java.

Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	10.6%
4.1.x	Jelly Bean	16	26.5%
4.2.x		17	19.8%
4.3		18	7.9%
4.4	KitKat	19	20.9%

Data collected during a 7-day period ending on August 12, 2014.
Any versions with less than 0.1% distribution are not shown.

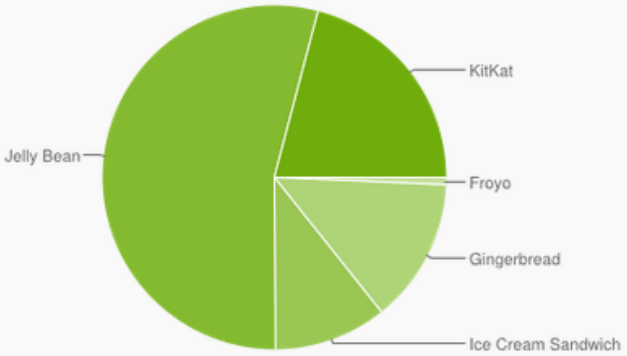


Figura 3.2 Percentuali di dispositivi Android per versione

3.2.2 Libreria Weka

La realizzazione del sistema di riconoscimento è legata strettamente alle indicazioni fornite in [1]. Lo strumento impiegato per le operazioni di classificazione e training è la libreria Java "Weka 3" [9], una collezione di algoritmi di machine learning per operazioni di data mining. Come descritto nel Capitolo 2, per generare il classificatore viene adoperata l'implementa-

zione di Weka dell'algoritmo Random Forest [2], date le prestazioni migliori rispetto agli altri presi in considerazione. Tuttavia, non si può escludere che future analisi dimostrino vantaggi nell'uso di algoritmi diversi da Random Forest, in termini di precisione o risorse necessarie per la generazione del classificatore. Durante la realizzazione del progetto è stata tenuta in conto questa possibilità, e sono stati forniti metodi per l'utilizzo di algoritmi di vario genere.

3.2.3 Strumenti di Sviluppo

Una volta stabiliti la piattaforma e il linguaggio da impiegare per il progetto, è stato possibile effettuare una scelta degli strumenti da utilizzare. L'intero progetto è stato realizzato attravero l'ambiente di sviluppo Eclipse, dotato di ADT (Android Development Tools), il plugin ufficiale per la creazione di applicazioni Android. Esistono altri IDE a disposizione degli sviluppatori, tuttavia le altre soluzioni più famose (Android Studio e IntelliJ IDEA) non erano adeguate agli scopi di questo progetto.
Come sistema di versionamento è stato scelto Git, per la sua compatibilità con la maggior parte dei servizi di hosting web per software, e per le mie precedenti esperienze nel suo utilizzo in altri progetti. Per la pubblicazione del codice e la gestione delle issues è stato utilizzato il servizio BitBucket. Le versioni stabili delle applicazioni sono state invece distribuite (gratuitamente) ad un gruppo di tester attraverso Google Play, il negozio virtuale ufficiale di Google.

3.2.4 Altro

Oltre alle componenti e ai servizi descritti nelle sezioni precedenti, è giusto citare altri strumenti che sono stati impiegati durante la realizzazione del progetto.
Per creare gli elementi grafici delle applicazioni, come icone e bottoni, sono stati impiegati i software Inkscape e GIMP. Tutte le risorse di tipo drawable

incluse nel progetto sono elementi standard del framework di Android oppure versioni modificate di immagini vettoriali disponibili gratuitamente in rete. La funzionalità di creazione e visualizzazione di grafici è fornita dalla libreria aChartEngine per applicazioni Android. Questa libreria è disponibile gratuitamente e rilasciata sotto licenza Apache 2.0.

Capitolo 4

Realizzazione

Una volta stabilite le caratteristiche e i requisiti del progetto, è stato possibile procedere alla realizzazione effettiva. Inizialmente è stata sviluppata un'applicazione singola per il corso di Laboratorio di Applicazioni Mobili. Questa è servita come dimostrazione pratica delle tecniche descritte in [1]. Successivamente ho concentrato i miei sforzi nella creazione della libreria Android e nella separazione delle due funzioni principali del progetto, quella di riconoscimento e quella di gestione dello stato del dispositivo, in due applicazioni distinte. Infine, sono state apportate modifiche per migliorare l'interfaccia grafica e renderla più semplice per l'utente finale.
Il progetto è stato sviluppato nell'arco di due anni. Di conseguenza, sono esistite più versioni stabili delle due applicazioni che compongono WAID. La versione illustrata in questa tesi è quella attualmente disponibile ai tester.

4.1 Libreria aVehicleRecognizer

L'insieme delle classi che gestiscono il riconoscimento del tipo di mobilità è completamente indipendente dal resto del sistema, ed è racchiuso in una libreria nominata "aVehicleRecognizer". Questa libreria sfrutta i metodi e le classi inclusi in Weka 3 per analizzare i dati forniti dall'applicazione e fornire diverse funzionalità. Attraverso la classe ModelManager è possibile generare

un nuovo modello di classificazione a partire da un set di dati, e creare valutazioni a partire da esso. La classe astratta VehicleManager invece fornisce metodi per il training e il riconoscimento dei tipi di mobilità. La libreria non comprende invece metodi per salvare e caricare i dati prodotti, siano essi valutazioni o misurazioni di training. Per offrire maggiore flessibilità, la decisione sul modo di preservare ed eventualmente condividere tali informazioni è lasciato allo sviluppatore. La libreria non richiede permessi aggiuntivi all'applicazione Android in cui viene utilizzata, dato che il suo funzionamento dipende solamente dai sensori presenti sul dispositivo.
Di seguito sono illustrate nel dettaglio le diverse classi che compongono aVehicleRecognizer.

4.1.1 Classi

MagnitudeFeatures

Questa classe rappresenta il set di feature calcolato a partire dalle magnitudo rilevate da VehicleObserver. Questo set è formato dal valore massimo, minimo, medio e dalla deviazione standard dei valori registrati.

VehicleInstance

Rappresenta una classificazione del tipo di mobilità, e viene utilizzata come classe per scambiare dati fra libreria e applicazione. Contiene i set di feature per accelerometro e giroscopio, il veicolo associato alla classificazione e il timestamp del momento in cui è stata creata.

ModelManager

É la classe che gestisce il riconoscimento del tipo di mobilità. Sfrutta le funzionalità della libreria Weka 3 per generare un modello di classificazione a partire dai dati forniti dall'applicazione. Tale modello viene poi serializzato e salvato in un file all'interno della cartella di default dell'applicazione (/data/data/<package_name>/), e viene impiegato per classificare oggetti di tipo VehicleInstance. ModelManager fornisce

anche metodi per importare ed esportare modelli o set di istanze nel formato di Weka (.arff).

VehicleManager

Questa classe astratta rappresenta lo scheletro su cui sono costruite le funzionalità di lettura dei sensori e comunicazione con l'applicazione. Fornisce metodi per impostare il ModelManager e la frequenza con cui vengono inviate le classificazioni.

VehicleRecognizer

Basata su VehicleManager, effettua il riconoscimento del tipo di mobilità attraverso ModelManager, a partire dai dati forniti da MagnitudeListener. Genera e classifica oggetti di tipo VehicleInstance che vengono inviati all'applicazione tramite l'interfaccia VehicleObserver.

VehicleTrainer

Anch'essa basata su VehicleManager, questa classe si limita a generare oggetti VehicleInstance con nome del veicolo definito dall'applicazione, saltando la fase di classificazione. Queste istanze possono quindi essere utilizzate per il training di un modello di classificazione.

MagnitudeListener

Questa classe implementa l'interfaccia SensorEventListener, e viene quindi utilizzata per registrare le misurazioni di un sensore (accelerometro o giroscopio). Periodicamente, calcola un set di feature a partire dai dati raccolti e lo invia come oggetto MagnitudeFeatures all'istanza di VehicleManager a cui è associata.

VehicleObserver

Questa interfaccia viene registrata dall'applicazione tramite VehicleManager. Ha un unico metodo, onNewInstance(VehicleInstance), che permette l'invio di un'oggetto di tipo VehicleInstance da VehicleManager all'applicazione.

Figura 4.1 Classi Libreria aVehicleRecognizer

4.2 WAID Recognizer

Questa applicazione gestisce le due funzionalità principali di questo progetto: il riconoscimento del tipo di mobilità dell'utente e la condivisione dei dati registrati con altre applicazioni. La prima si basa essenzialmente sulla libreria aVehicleRecognizer, descritta nella sezione precedente, mentre la seconda sfrutta un Content Provider, ovvero un'interfaccia di accesso condivisa al database di WAID Recognizer. Applicazioni esterne possono quindi recuperare o modificare la lista dei veicoli riconosciuti, le loro misurazioni di training e la lista aggiornata delle valutazioni prodotte. Il database impiegato per la registrazione dei dati è di tipo SQLite.
Di seguito sono descritte le varie componenti dell'interfaccia grafica dell'applicazione, insieme al loro funzionamento.

4.2.1 Riconoscimento

Attraverso un simbolo di accensione è possibile controllare lo stato della funzionalità di riconoscimento del tipo di mobilità. La sua pressione da parte dell'utente causa l'attivazione del servizio di riconoscimento, e le valutazioni prodotte vengono visualizzate in un elenco, comprendente nome del veicolo e timestamp del riconoscimento, e con un'icona che sostituisce il simbolo di accensione. Un'ulteriore pressione dell'icona del veicolo riconosciuto provoca la terminazione del servizio.

4.2.2 Training

Questa scheda permette la gestione delle misurazioni per i veicoli riconosciuti. Attraverso una lista è possibile scegliere il mezzo per cui effettuare il training, e un selettore permette di scegliere se sovrascrivere o meno le precedenti misurazioni. Un simbolo di attivazione simile a quello presente nella scheda di Riconoscimento permette l'attivazione e lo spegnimento del servizio di training.

4.2.3 Storico

Questa scheda presenta uno storico delle valutazioni prodotte dal servizio di riconoscimento durante il tempo. L'interfaccia presenta due selettori: il primo permette di scegliere il periodo temporale fra giornata odierna, settimana o mese passato oppure dalla prima valutazione memorizzata; il secondo selettore determina il tipo di informazione visualizzata, e permette di scegliere fra veicolo riconosciuto, magnitudo dell'accelerometro e magnitudo del giroscopio. Le informazioni vengono presentate in un grafico temporale prodotto attraverso la libreria aChartEngine.

4.2.4 Impostazioni

La finestra delle impostazioni si divide in tre sezioni: la prima presenta la lista dei veicoli riconosciuti, le due successive comprendono le configurazioni rispettivamente per la modalita di riconoscimento e per la modalità di training. Dalla sezione dei veicoli è possibile aggiungere o eliminare veicoli dalla lista. Ogni veicolo viene elencato insieme alla propria icona associata e al numero di misurazioni di training effettuate per esso. Attraverso le configurazioni di riconoscimento e training è invece possibile specificare il numero di secondi fra una valutazione e l'altra e il numero di secondi di ritardo nell'avvio del servizio di training. Dal menu delle impostazioni è anche possibile riportare il modello di classificazione e i veicoli riconosciuti ad uno stato di default, attraverso il pulsante apposito.

4.3 WAID Profiler

Sfruttando i dati forniti da WAID Recognizer, questa applicazione permette la gestione automatica di alcune funzionalità del dispositivo, seguendo profili impostati dall'utente e associati ai vari veicoli riconosciuti. Nella versione attuale della applicazione è possibile decidere lo stato (attivo o inattivo) della connessione Wi-Fi, della connessione Bluetooth e del vivavoce. L'uten-

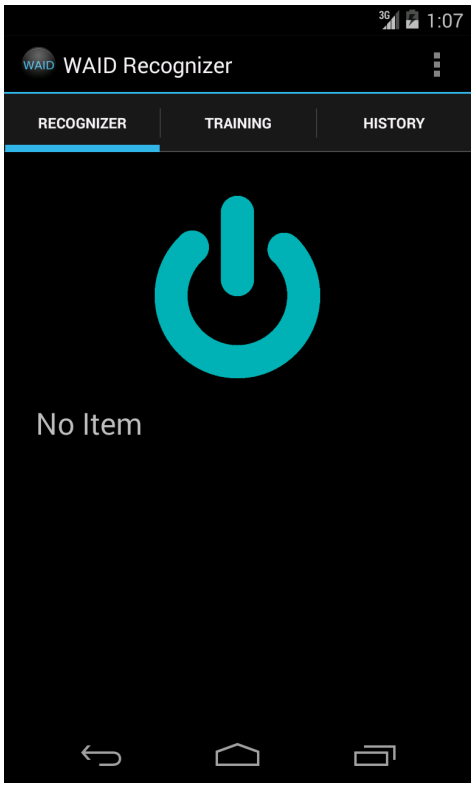
te può anche decidere di "ignorare" un tipo di mobilità, mantenendo lo stato del dispositivo inalterato durante il suo riconoscimento.
Come specificato nel capitolo 2, la precisione delle valutazioni fornite da WAID Recognizer non è assoluta. WAID Profiler sfrutta due tecniche diverse per ovviare a questo problema ed evitare cambi di stato superflui. La prima è descritta in [1], e consiste nel considerare come tipo di mobilità più probabile il veicolo più ricorrente in un certo numero di valutazioni recenti, che viene definito "storico". Questo sistema evita passaggi di stato causati dal riconoscimento errato di un veicolo con un pattern di movimento simile a quello del mezzo effettivo. Il secondo metodo sfrutta una serie di liste di transizioni da ignorare, associate ai vari veicoli riconosciuti da WAID Recognizer. Queste "blacklist" di transizioni sono utili in casi dove il riconoscimento dell'azione è eseguito in maniera corretta, ma un passaggio di stato non sarebbe coerente con il contesto in cui si trova il dispositivo. Un esempio pratico può essere un viaggio in treno: nei momenti in cui il mezzo è fermo, il sistema riconoscerebbe l'azione come "fermo", ma l'utente sarebbe comunque a bordo del treno. Queste blacklist sono configurabili attraverso l'applicazione.
Di seguito sono descritti i diversi elementi dell'interfaccia grafica.

4.3.1 Ascolto

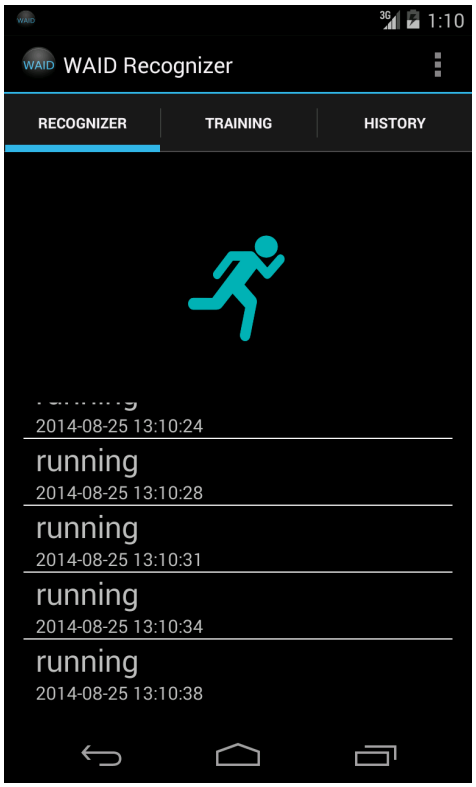
Questa scheda è graficamente simile a quella di Riconoscimento nell'applicazione WAID Recognizer. Esiste un solo elemento con cui l'utente può interagire, il simbolo di accensione, che controlla lo stato del servizio di ascolto. Questo servizio, una volta attivo, controlla aggiornamenti nel Content Provider delle valutazioni di WAID Recognizer. Ad ogni nuova valutazione, viene calcolato il nuovo veicolo predetto in base allo storico delle valutazioni precedenti. Tale veicolo viene rappresentato attraverso la propria icona, che sostituisce il simbolo di accensione durante l'esecuzione del servizio. La pressione di tale icona causa la terminazione del servizio di ascolto.

4.3.2 Profili

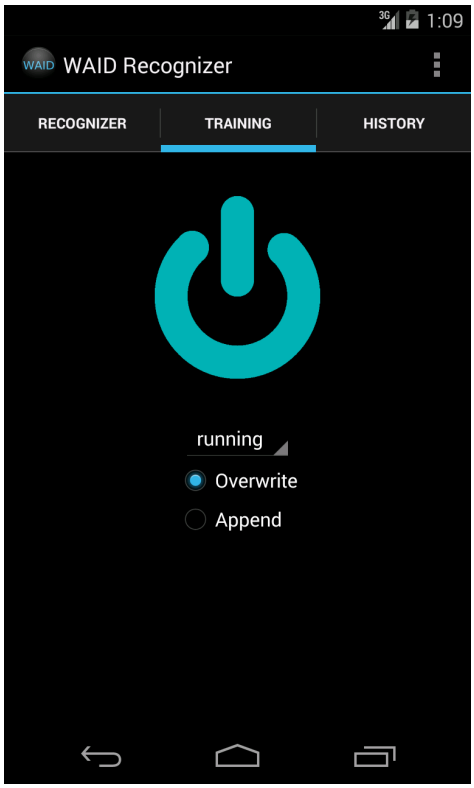
Attraverso questa scheda è possibile gestire i profili per i veicoli riconosciuti e impostare alcuni parametri di comportamento dell'applicazione. La sezione dei profili comprende un elenco dei veicoli, e ad ogni elemento è associato un insieme di selettori che permettono di scegliere lo stato delle varie funzionalità del dispositivo e se ignorare o meno tale veicolo. Esiste un'ulteriore sezione che permette di definire le blacklist di transizioni per i diversi tipi di mobilità. I parametri configurabili dell'applicazione sono la lunghezza dello storico sul quale viene calcolato il veicolo predetto e il salvataggio e recupero dello stato del dispositivo precedente all'esecuzione del servizio.



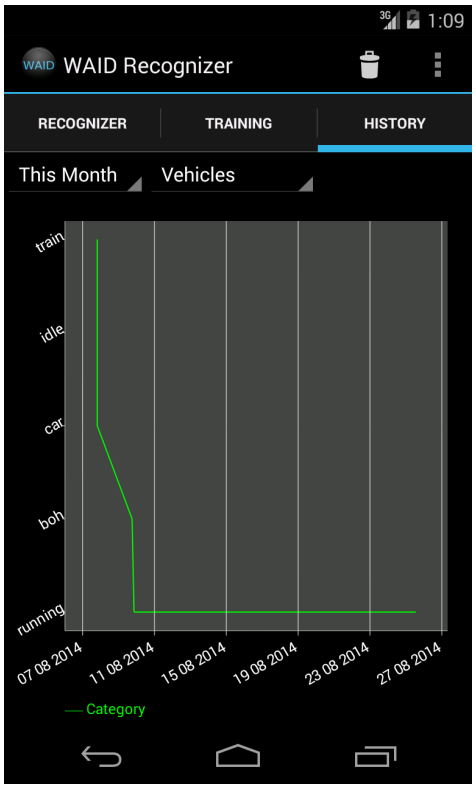
(a) Scheda Riconoscitore



(b) Riconoscitore in funzione

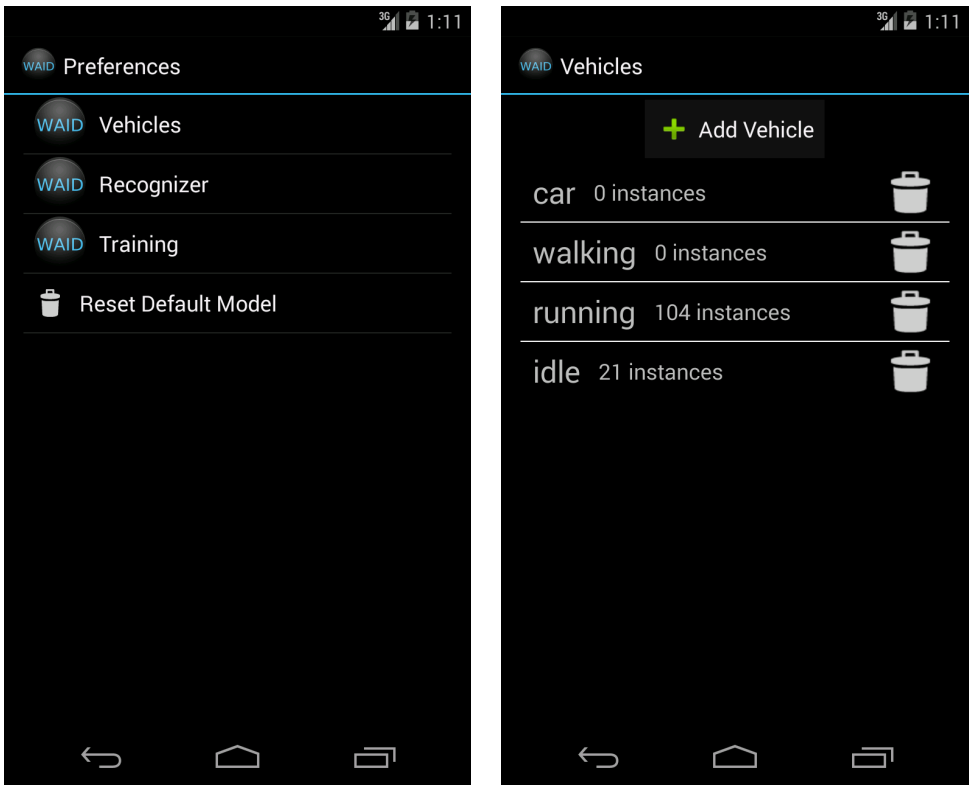


(c) Scheda Training



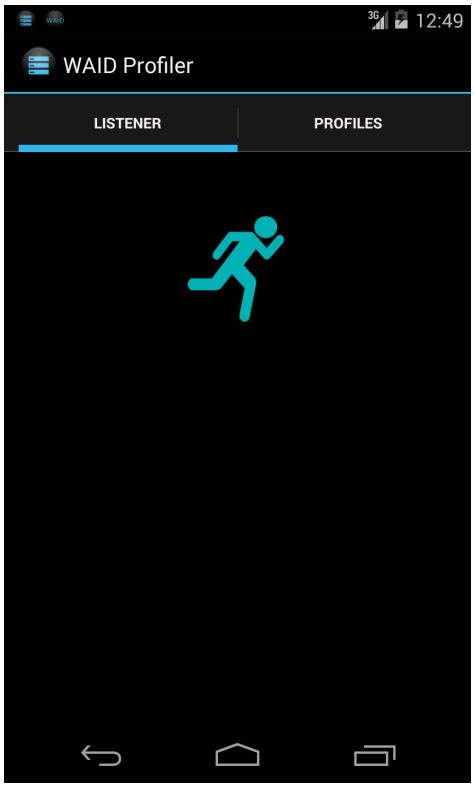
(d) Scheda Storico

Figura 4.2 Interfaccia di WAID Recognizer - Schede

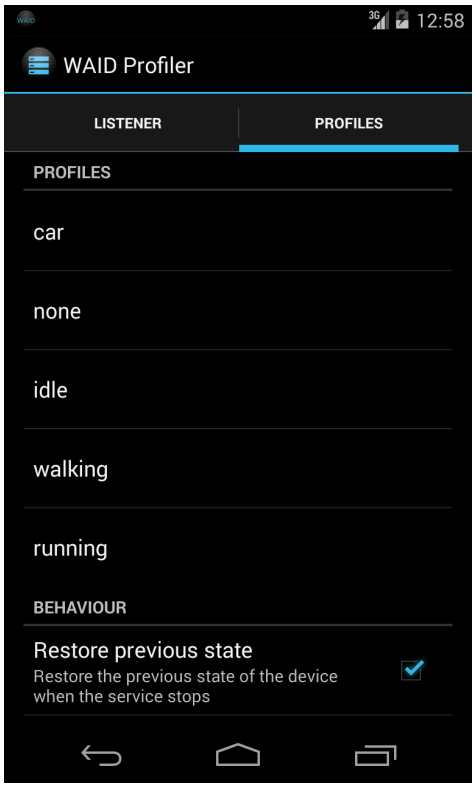


(a) Impostazioni **(b)** Lista dei veicoli

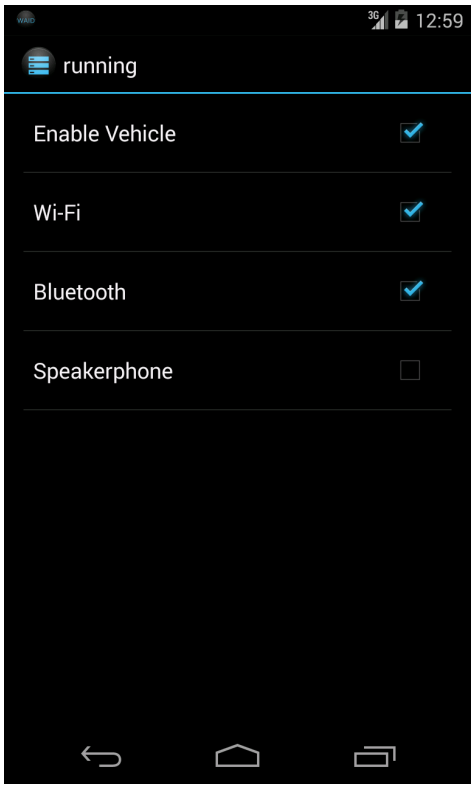
Figura 4.3 Interfaccia di WAID Recognizer - Impostazioni



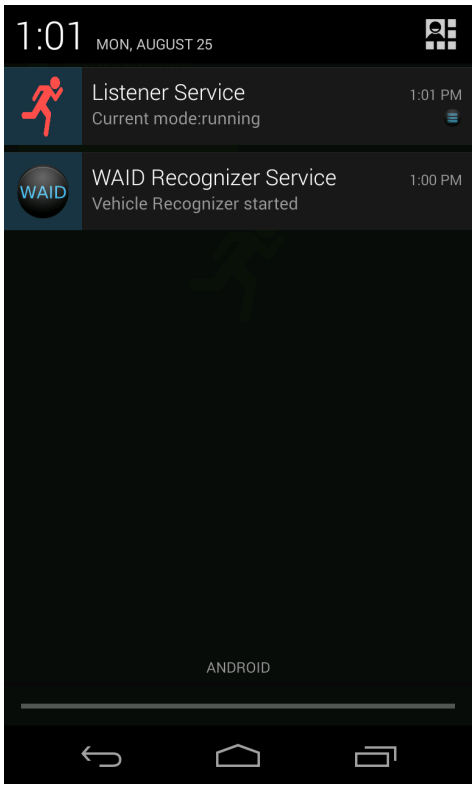
(a) Ascolto in funzione



(b) Scheda Profili



(c) Profilo per corsa



(d) Notifica del riconoscimento

Figura 4.4 Interfaccia di WAID Profiler

Capitolo 5

Risultati

Dopo la realizzazione del software è seguita una fase di testing dell'applicazione di riconoscimento. Questa è stata necessaria per ottenere un riscontro sul funzionamento effettivo delle tecniche descritte in [1] in un contesto reale. Le statistiche prodotte dalla ricerca infatti rappresentano risultati sperimentali, mentre l'uso dell'applicazione da parte dell'utente può introdurre una componente di errore non trascurabile. Inoltre, avendo introdotto la feature di modifica della lista di veicoli riconosciuti, è stato necessario analizzare i limiti di questo metodo di riconoscimento in presenza di tipi di mobilità diversi da quelli considerati inizialmente.

5.1 Dati di utilizzo

5.1.1 Risultati teorici

In [1] è stata prodotta una valutazione della precisione di un classificatore generato con l'algoritmo Random Forest. Le statistiche sono state generate utilizzando una lista di 72000 magnitudo per ogni tipo di mobilità preso in considerazione: "camminare", "automobile" e "treno". Effettuando una 10-fold cross-validation del set di misurazioni utilizzato per generare il classificatore, è stato possibile fare alcune osservazioni. Innanzitutto viene osservato come si ottenga una precisione ottimale utilizzando l'unione dei set

di feature per accelerometro (F^A) e giroscopio (F^G). Inoltre, viene osservato come la lunghezza dei periodi di raccolta per ogni set di feature non incida particolarmente sulla precisione del classificatore.
I risultati ottenuti da questa analisi sono riassunti nella Tabella 5.1.

Categoria	F^A	F^G	$F^{AG} = F^A \cup F^G$
camminare	97,40 %	93,10 %	96,70 %
automobile	62,80 %	65,90 %	94,10 %
treno	94,50 %	94,50 %	98,80 %
totale	93,49 %	91,94 %	97,71 %

Tabella 5.1 Precisione per diversi set di feature usando l'algoritmo Random Forest

5.1.2 Statistiche reali

I risultati illustrati nella sezione precedente sono stati ottenuti confrontando il classificatore con lo stesso set di misurazioni usato per generarlo. Inoltre, i set utilizzati comprendono 7200 misurazioni per tipo di mobilità, contando una lunghezza dei periodi di raccolta di 5 secondi.
In un contesto reale, la precisione delle classificazioni potrebbe essere minore, a causa di diversi fattori. In primo luogo, l'utente potrebbe non essere disposto ad effettuare un numero elevato di misurazioni di training per ogni veicolo. Bisogna inoltre tenere conto di possibili misurazioni errate durante il training, ad esempio nei momenti in cui il veicolo è fermo. Infine, possono esistere tipi di mobilità con pattern di movimento molto simili fra di loro.
Le statistiche prodotte per questa tesi sono state realizzate prendendo in considerazione cinque tipi di mobilità differenti: fermo, camminare, automobile, treno, bicicletta. Per studiare il rapporto fra i set di training e la precisione del classificatore, le classificazioni sono state prodotte impiegando set di dimensioni diverse. Inoltre, è stato preso in considerazione l'effetto che l'aggiunta di uno storico delle classificazioni di lunghezza $|S|$ può avere sulla

precisione.
I risultati del primo set di classificazioni raccolto sono illustrati dalla Tabella 5.2.

Categoria	Misurazioni di training	Precisione con $\|S\| = 0$	Precisione con $\|S\| = 5$	Precisione con $\|S\| = 10$
fermo	2457	77,78 %	81,80 %	84,20 %
camminare	1047	75,84 %	85,37 %	86,44 %
automobile	1203	49,09 %	51,82 %	50,90 %
treno	627	76,91 %	83,70 %	81,49 %
bicicletta	166	68,93 %	75,14 %	86,44 %

Tabella 5.2 Precisione delle classificazioni - Set 1

Si può notare come le classificazioni per "automobile" abbiano una precisione notevolmente inferiore rispetto agli altri tipi di mobilità. Questo è dovuto in parte all'uso di un veicolo differente da quello impiegato per raccogliere le misurazioni di training. La differenza di pattern di movimento fra i due tipi di automobile causa il mancato riconoscimento da parte del classificatore. Questo problema è stato poi risolto effettuando nuove misurazioni di training. Il cambiamento nella categoria "automobile" ha però avuto effetti anche sulla classificazione degli altri tipi di mobilità, come si può osservare dai risultati elencati nella Tabella 5.3.

Si nota immediatamente come la precisione delle categorie "fermo", "camminare" e "bicicletta" sia rimasta simile ai valori precedentemente calcolati, mentre il numero di classificazioni corrette per "treno" sia sceso sotto il 50 %, a fronte di un drastico aumento di precisione per "automobile". Il motivo è probabilmente lo squilibrio fra le quantità di misurazioni di training per queste ultime due azioni, il cui pattern di movimento è evidentemente simile. Un altro problema riscontrato è illustrato dai risultati di "treno" e "bicicletta". Essi evidenziano che, in casi dove il numero di valutazioni corrette per

Categoria	Misurazioni di training	Precisione con $\|S\| = 0$	Precisione con $\|S\| = 5$	Precisione con $\|S\| = 10$
fermo		81,40 %	84,11 %	84,86 %
camminare		82,79 %	85,46 %	88,21 %
automobile	2428	90,82 %	96,58 %	97,14 %
treno		48,74 %	42,56 %	41,19 %
bicicletta	166	50,42 %	52,94 %	42,02 %

Tabella 5.3 Precisione delle classificazioni - Set 2

un veicolo scende sotto la metà del totale, l'uso di uno storico delle classificazioni diminuisce ulteriormente la precisione per quel tipo di mobilità.

Un'ulteriore raccolta di misurazioni di training per "bicicletta" e "treno" ha portato a risultati più incoraggianti, illustrati nella Tabella 5.4.

Categoria	Misurazioni di training	Precisione con $\|S\| = 0$	Precisione con $\|S\| = 5$	Precisione con $\|S\| = 10$
fermo	4132	82,53 %	88,98 %	90,45 %
camminare	1946	91,84 %	99,01 %	100 %
automobile	2428	94,36 %	100 %	100 %
treno	1043	73,13 %	76,17 %	77,57 %
bicicletta	504	62,81 %	76,31 %	82,59 %

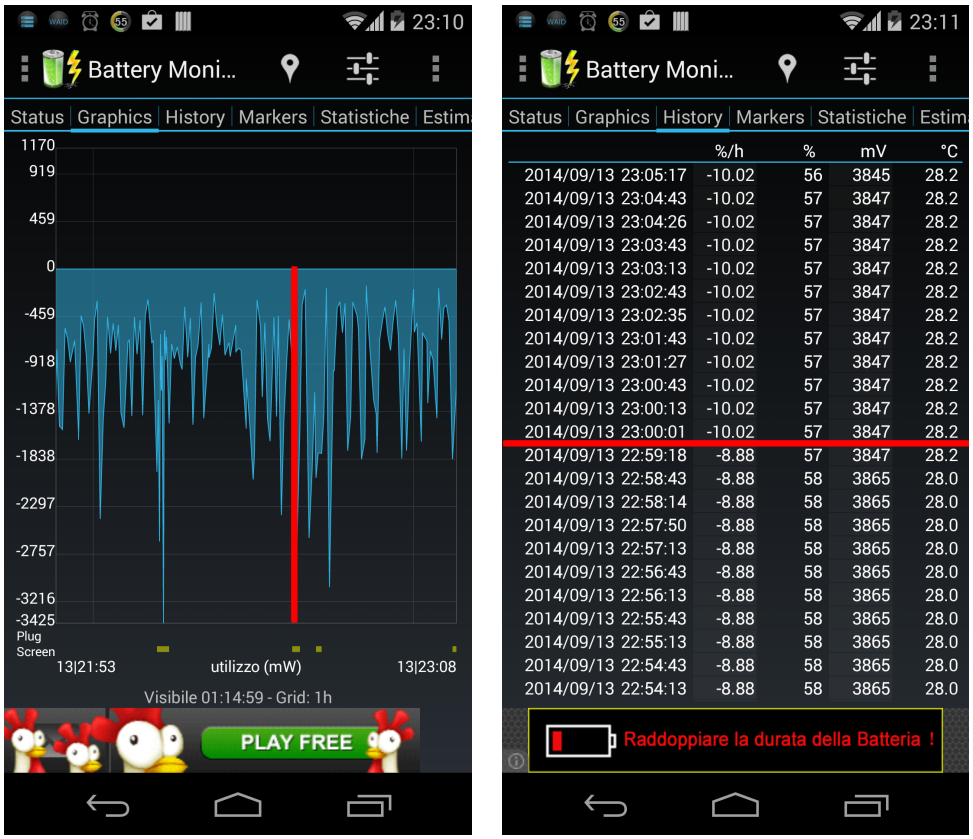
Tabella 5.4 Precisione delle classificazioni - Set 3

Le statistiche ottenute da queste misurazioni lasciano intuire quindi che la precisione nel riconoscimento di un tipo particolare di mobilità non dipenda solamente dal training effettuato per esso, ma anche dal training effettuato

per le altre categorie di veicolo. Si può osservare anche che la precisione effettiva del classificatore può raggiungere e superare il 90% per alcuni tipi di mobilità, anche con un tempo di training inferiore alle 3 ore (2160 misurazioni). Infine, è chiaro il guadagno significativo in precisione attraverso l'utilizzo di uno storico di classificazioni.

5.2 Consumi

Dato che il sistema di riconoscimento del tipo di mobilità non sfrutta la tecnologia GPS, il consumo energetico del dispositivo è influenzato solamente dal funzionamento delle applicazioni di riconoscimento e gestione dei profili. Queste mantengono il dispositivo attivo costantemente per elaborare i dati forniti dai sensori, quindi il consumo è maggiore rispetto a quello di un dispositivo in stand-by.
La Figura 5.1 illustra i consumi delle applicazioni WAID Recognizer e WAID Profiler in uno smartphone LG Nexus 5, monitorati attraverso Battery Monitor Widget. La linea rossa rappresenta il momento in cui vengono attivate le applicazioni. Si può osservare che il loro utilizzo non provoca un aumento considerevole nei consumi del dispositivo, e la percentuale di carica impiegata nell'arco di un'ora aumenta di meno del 2% con una batteria da 2300 mAh.



(a) Grafico del consumo energetico (mW)

(b) Storico dello stato della batteria del dispositivo

Figura 5.1 Consumo energetico di WAID Profiler e WAID Recognizer

Capitolo 6

Sviluppi Futuri

Nelle sezioni precedenti sono state illustrate le prestazioni effettive del servizio di riconoscimento. É stato osservato come esistano delle differenze fra statistiche sperimentali e risultati reali, e si è parlato dei limiti intrinsechi di questo metodo. Di conseguenza, si possono definire alcuni miglioramenti possibili per la logica del sistema di riconoscimento o per le applicazioni Android prodotte.

6.1 GPS

Si è visto come esistano mezzi di trasporto con pattern di movimento molto simili fra di loro dal punto di vista delle misurazioni effettuate da accelerometro e giroscopio. Questo può causare classificazioni errate anche frequenti. Una soluzione parziale a questo problema può essere l'introduzione della velocità del mezzo di trasporto fra i dati analizzati dal sistema di riconoscimento. Questo tipo di dato si può ottenere sfruttando la funzionalità GPS fornita dal framework Android.
Dato che lo stato (attivo o inattivo) del GPS non può essere deciso autonomamente da una applicazione, si potrebbe offrire all'utente la possibilità di attivare o disattivare a piacere l'uso di questa funzionalità per aumenta-

re la precisione nel riconoscimento, con la controindicazione di un maggior consumo energetico da parte del dispositivo.

6.2 Algoritmi di classificazione

L'algoritmo impiegato per generare il classificatore nell'applicazione di riconoscimento è Random Forest, ed è stato scelto seguendo le analisi statistiche effettuate in [1]. Tuttavia, le opzioni prese in considerazione nella ricerca non sono le uniche disponibili. Si potrebbe effettuare uno studio più approfondito di altri algoritmi per trovare soluzioni migliori da un punto di vista dell'accuratezza delle classificazioni e delle risorse richieste per generare il classificatore.

6.3 Applicazione Web

Conservare le misurazioni di training e generare classificatori sono operazioni che richiedono notevoli risorse in termini di memoria e potenza di calcolo, e non tutti i dispositivi presenti sul mercato potrebbero avere i requisiti necessari per eseguirle. Una soluzione potrebbe essere delegare queste operazioni ad un servizio remoto. Si potrebbe creare un'applicazione web che gestisca le misurazioni di training effettuate dall'utente e permetta il download di un classificatore generato a partire da queste.
Un'altra funzionalità di questo servizio potrebbe essere la condivisione delle misurazioni effettuate dagli utenti per generare statistiche di vario tipo, come i veicoli impiegati in particolari contesti geografici o sociali.

6.4 Interfaccia Utente

L'interfaccia utente delle applicazioni WAID Recognizer e WAID Profiler non si può ancora definire completa per un prodotto commerciale. Tralasciando gli aspetti puramente stilistici, esistono miglioramenti possibili per

ridurre l'intervento dell'utente nel funzionamento dell'applicazione.
Uno di questi può essere l'introduzione di una tabella di intervalli temporali per la gestione automatica del training del sistema di riconoscimento. L'utente avrebbe quindi la possibilità di pianificare a priori i momenti in cui effettuare misurazioni per un determinato tipo di mobilità.
Si potrebbe inoltre automatizzare il sistema di training in modo che, superata una certa precisione di classificazione per un determinato veicolo, interrompa la raccolta di misurazioni per quel tipo di mobilità.

Capitolo 7

Conclusioni

In questa tesi è stato mostrato come sia possibile creare un sistema di riconoscimento del tipo di mobilita' a partire da tecniche che sfruttano solamente i sensori presenti su un dispositivo mobile. è stata prodotta un'applicazione basata su tale sistema di riconoscimento e che fornisce un servizio di Content Provider ad altri programmi. Infine, è stata prodotta un'applicazione che utilizza le informazioni provenienti da questo Content Provider per cambiare lo stato del dispositivo. La precisione del sistema di riconoscimento è stata valutata attraverso la raccolta di dati di utilizzo in un contesto reale.
Durante la fase di testing è stato possibile stabilire gli effettivi vantaggi e svantaggi nell'uso di questa tecnologia rispetto a metodi di riconoscimento contestuale correntemente in uso. In particolare, impiegare solamente i sensori presenti sul dispositivo per il riconoscimento comporta un risparmio energetico notevole rispetto all'uso della tecnologia GPS, oltre che rendere il servizio indipendente da altre infrastrutture. Inoltre, individuare pattern di movimento a partire dalle misurazioni dei sensori permette di distinguere diversi veicoli fra di loro, piuttosto che riconoscere in maniera più generale un'azione dell'utente.
D'altra parte si è potuto vedere come, per ottenere una precisione superiore al 90%, sia necessario aumentare il tempo di reazione dell'applicazione introducendo una funzione di decisione basata su uno storico delle classificazioni.

É stato anche osservato che un numero ridotto di misurazioni di training può causare riconoscimenti errati frequenti, specialmente in casi dove due o più tipi di mobilità hanno pattern di movimento simili.
Potrebbe essere possibile migliorare le prestazioni di questo sistema attraverso alcune modifiche o aggiunte. Ritengo che sia importante sviluppare in futuro ulteriori tecniche per automatizzare la raccolta di misurazioni di training, in modo da ridurre l'intervento necessario da parte dell'utente per il corretto funzionamento del sistema di riconoscimento. Si potrebbe inoltre integrare le misurazioni di accelerometro e giroscopio con dati forniti dal GPS, ad esempio la velocità del dispositivo, e studiare l'impatto che questa aggiunta ha sulla precisione del sistema.
In conclusione, la tecnica di riconoscimento del tipo di mobilità descritta in [1] si è rivelata soddisfacente per un impiego in applicazioni mobili rivolte ad un utilizzo comune, ma necessita ulteriore sviluppo per automatizzare le operazioni di raccolta dati. Basandosi su questa tecnica è stato possibile costruire una soluzione standard per gli sviluppatori, sotto forma di una libreria Android che verrà pubblicata nel prossimo periodo e di cui continuerò lo sviluppo in futuro.

Bibliografia

[1] L. Bedogni, M. Di Felice, and L. Bononi. By train or by car? detecting the user's motion type through smartphone sensors data. In *Wireless Days (WD), 2012 IFIP*, pages 1–6, Nov 2012. doi: 10.1109/WD.2012.6402818.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A%3A1010933404324`.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL `http://dx.doi.org/10.1023/A:1022627411411`.

[4] Google. Recognizing the user's current activity, . URL `http://developer.android.com/training/location/activity-recognition.html`. consultato il 1 Settembre 2014.

[5] Google. Dashboards, . URL `https://developer.android.com/about/dashboards/index.html`. consultato il 1 Settembre 2014.

[6] George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-385-9. URL `http://dl.acm.org/citation.cfm?id=2074158.2074196`.

[7] Donald E. Knuth. Computer Programming as an Art. *Communications of the ACM*, 17(12):667–673, December 1974.

[8] A. Lawida, Y.C. Park, and O. Sorenson. Vehicle recognition using multiple metrics, May 6 2005. URL `https://www.google.ca/patents/WO2005041071A1?cl=en`. WO Patent App. PCT/US2004/035,175.

[9] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 0120884070.

Tanto lo so che per settembre non ce la fai.

— Un simpatico mascalzone

Ringraziamenti

Ringrazio Luca Bedogni per la sua disponibilità durante lo svolgimento di questo progetto.

Un grazie sentito alla gente di Blackmesa per i consigli, per le risate e per la birra.

Ringrazio i ragazzi del kendo per le bastonate in compagnia e per aver tollerato le mie paturnie da laureando.

A Duma, Lollum e Berni posso solo dire che mi dispiace per quel *return* mancante. A loro e agli altri Tapiri, grazie.

Ringrazio fortemente la mia famiglia per avermi supportato durante questo percorso, e l'Alice per avermi sopportato in generale.