Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jasna Urbančič

Detekcija prevoznega sredstva z mobilnimi senzorji

magistrsko delo

magistrski program druge stopnje
računalništvo in informatika

Mentor: doc. dr. Veljko Pejović
Somentor: prof. dr. Dunja Mladenić

Ljubljana, 2018

University of Ljubljana
Faculty of Computer and Information Science

Jasna Urbančič

Transportation mode detection based on mobile sensor data

master's thesis

the 2nd cycle master's study programme
computer and information science

Supervisor: doc. dr. Veljko Pejović
Co-supervisor: prof. dr. Dunja Mladenić

Ljubljana, 2018

Acknowledgments

I would like to express my sincere gratitude to my Master's thesis supervisor doc. dr. Veljko Pejović and co-supervisor prof. dr. Dunja Mladenić for their support, guidance, and patience during the process of writing this thesis. I want to thank Luka Bradeško for giving me the opportunity to work on transportation mode detection within the OPTIMUM project. I would like to thank Matej Senožetnik, who jointly with Luka Bradeško developed the library that was crucial to the data collection. Additionally, I would like to thank all my colleagues from the Artificial Intelligence Laboratory (Jožef Stefan Institute) and others, especially Sebastjan Fabijan, Zala Herga, Blaž Novak, Erik Novak, and Tine Šubic. I would also like to thank the members of the committee izr. prof. dr. Zoran Bosnić, izr. prof. dr. Patricio Bulić, and doc. dr. Dejan Lavbič for their comments, critiques, and suggestions that allowed me to improve my work. Finally, I am grateful for all the support I received from my family.

This work was supported by the Slovenian Research Agency under project Integration of mobile devices into survey research in social sciences: Development of a comprehensive methodological approach *(J5-8233), and the ICT program of the EC under project* OPTIMUM *(H2020-MG-636160).*

Jasna Urbančič, 2018

Contents

List of used acronmys

acronym	meaning
CDR	call detail records
GPS	global positioning system
GSM	global system for mobile (communications)
CA	classification accuracy
kNN	k-nearest neighbors
OSM	OpenStreetMap
DHHM	discrete hidden Markov model
API	application program interface
SPD	stay-point detection
GUI	graphical user interface
RF	random forest
SVM	support vector machine
NN	neural network
RBF	radial basis function
PCA	principal components analysis

Povzetek

Naslov: Detekcija prevoznega sredstva z mobilnimi senzorji

V delu obravnavamo detekcijo prevoznega sredstva z mobilnimi senzorji in metodami strojnega učenja. Pri tem uporabljamo kratke vzorce podatkov iz pospeškometra, ki jih zajamemo med uporabnikovim potovanjem v vozilu. Razločujemo med tremi prevoznimi sredstvi — avtom, avtobusom in vlakom. Vzorce predobdelamo tako, da iz pospeškov izločimo gravitacijsko komponento. Iz vzorcev izločimo statistične in frekvenčne značilke ter značilke vrhov. S statistično analizo značilk dobimo vpogled v podatke. Dodatno analiziramo značilke preko različnih množic značilk, ki jih uporabljamo za klasifikacijo. Kot klasifikatorje uporabljamo naključne gozdove, metodo podpornih vektorjev in nevronske mreže. Z uporabo nevronskih mrež smo pravilno razpoznali 65% avtomobilov, 63% avtobusov in 18% vlakov.

Ključne besede

strojno učenje, mobilno zaznavanje, podatkovno rudarjenje, razpoznava vzorcev, inteligentni transportni sistemi

Abstract

Title: Transportation mode detection based on mobile sensor data

This thesis addresses transportation mode detection based primarily on mobile phone data using machine learning methods. Our approach uses short samples of accelerometer readings taken while traveling in a vehicle to distinguish between three modalities — car, bus, and train. We use gravity estimation to pre-process the samples. We extract features from statistical, frequency-based, and peak-based domain. With statistical analysis of the features we gain an introspective into the data. To additionally analyze the features we construct several feature sets for classification. As a classifier we use random forest, support vector machine, and neural network. Our approach correctly classifies 65% cars, 63% buses, and 18% trains using neural network.

Keywords

machine learning, mobile sensing, data mining, pattern recognition, intelligent transportation systems

Razširjeni povzetek

Dandanes ima velik del svetovnega prebivalstva pametni telefon z mnogo aplikacijami. Nekatere izmed njih spremljajo kontekst oziroma okolje, v katerem se uporabnik nahaja, z množico senzorjev, vgrajenih v telefon. Beleženje in analizo informacij o okolici lahko uporabimo kot podporni sistem pri nadzorovanih spremembah vedenja, na primer pri odvajanju od kajenja, ali za zaznavo sprememb v obnašanju, ki lahko nakazujejo težave v duševnem zdravju, na primer depresijo.

Podobno lahko aplikacije z zaznavanjem okolice uporabniku predlagajo uporabo bolj trajnostnih prevoznih sredstev, za kar pa potrebujemo učinkovite in zanesljive algoritme za detekcijo prevoznega sredstva. Večina sodobnih mobilnih telefonov je opremljena s pospeškometrom, giroskopom, magnetometrom in GPS. Z vgrajenimi sistemi lahko razločujemo med mirovanjem, hojo, tekom, kolesarjenjem in vožnjo v vozilu, ne pa med različnimi tipi vozil, na primer avtom, avtobusom in vlakom. Razlikovanje med tipi vozil je ključno za raziskave mobilnosti, usmerjevalne aplikacije v mestnem okolju in ko želimo uporabnike preusmeriti k trajnostni mobilnosti.

Naš cilj je razviti metodo za zaznavanje prevoznega sredstva na podlagi podatkov iz senzorjev v mobilnem telefonu, ki v skoraj realnem času razlikuje med vožnjo v avtomobilu, avtobusu in vlaku, je energijsko učinkovita, ne zahteva uporabe prenosa podatkov in je računsko nezahtevna. Raziskovalna vprašanja vključujejo predobdelavo signalov, luščenje in izbiro značilk, izbiro evalvacijskega scenarija in evalvacijskih metrik, ter izbiro klasifikatorja.

Prvi prispevek našega dela je pristop k zajemanju senzorskih podatkov.

Pri tem si pomagamo z vgrajenimi sistemi in signal zajamemo le, če operacijski sistem zazna vožnjo v vozilu. Poleg tega zajamemo kratek, pet sekundni vzorec, da varčujemo z baterijo in količino podatkov, ki se prenaša med telefonom in strežnikom. Drugi prispevek so uporabljene značilke, saj je bila taka kombinacija značilk do sedaj redko uporabljena. Poleg tega smo dodatno analizirali značilke, da lahko ocenimo doprinos posamezne skupine značilk oziroma signala k rezultatu klasifikacije. Zadnji prispevek je jasno definiran evalvacijski scenarij s tremi strogo ločenimi množicami — učno, validacijsko in testno.

I Kratek pregled sorodnih del

Čeprav so se prvi poskusi zaznave aktivnosti začeli še pred razvojem pametnih telefonov, se je področje zares razcvetelo z razvojem mobilnih telefonov z vgrajenimi senzorji [1]. Za zaznavo prevoznega sredstva lahko uporabimo triangulacijo signala GSM ali lokalne brezžične signale, a so te metode precej nezanesljive v primerjavi z GPS in pospeškometrom [2]. V zadnjem času je največ poudarka na metodah, ki temeljijo na sledeh GPS in/ali podatkih iz pospeškometra.

V splošnem metode za detekcijo prevoznega sredstva sledijo poteku dela, ki je prikazan na Sliki 2.1. Podatke zberemo s pomočjo vgrajenih senzorjev. Oznake za nadzorovano učenje najpogosteje zberemo skupaj s podatki. Običajno zbiranju podatkov sledi predobdelava. V predobdelavi signale filtriramo, da izločimo šum in očitne napake v podatkih. Iz predobdelanih podatkov nato izločimo značilke za klasifikacijo. Najpogosteje značilke določi raziskovalec.

Pospeškometer meri pospešek v treh smereh, relativno glede na orientacijo naprave, zato uporaba telefona med vožnjo vpliva na meritve. Nekateri raziskovalci testnim uporabnikom dajo navodila, naj med meritvami ne uporabljajo telefona. V izogib omejitvam lahko uporabimo filtriranje in oceno gravitacije v predobdelavi [3].

Značilke izhajajo iz petih skupin — statistične, časovne, frekvenčne, značilke vrhov in segmentne [3]. Statistične, časovne in frekvenčne značilke so običajno izračunane iz nekaj sekundnih oken signalov in so zasnovane tako, da zaznajo visokofrekvenčno gibanje uporabnika, motorja in kontakt med kolesi in tlemi [3]. Segmentne značilke so izračunane iz celotnega segmenta vožnje [3], značilke vrhov pa so izračunane glede na lastnosti vrhov, ki se pojavijo v signalu, in naj bi zaznale nizkofrekvenčno gibanje, kot je na primer pospeševanje ali zaviranje. [3].

Najpogosteje uporabljene metode strojnega učenja za zaznavo prevoznega sredstva s podatki iz pospeškometra so odločitvena drevesa, metoda podpornih vektorjev in naključni gozdovi [2, 4], zadnje čase pa so priljubljene tudi nevronske mreže. Pogosto ti klasifikatorji nastopajo v ansamblih [5].

II Predlagana metoda

Za zbiranje podatkov uporabljamo mobilno knjižnico *NextPin* [6], ki jo je razvil Laboratorij za umetno inteligenco na Institutu Jožef Stefan. Knjižnica je vključena v dve brezplačni mobilni aplikaciji. Prva je *OPTIMUM Intelligent Mobility*, ki jo je razvil konzorcij projekta Optimum [7], in je v osnovi aplikacija za načrtovanje multimodalnih poti v treh evropskih mestih — na Dunaju, v Birminghamu in v Ljubljani. V obdobju pilotnega testiranja v maju in juniju 2018 je aplikacijo dnevno uporabljalo približno 120 uporabnikov, ki so bili za sodelovanje v testiranju nagrajeni. Druga aplikacija je *Mobility patterns*, ki so jo tako kot knjižnico *NextPin* razvili na Institutu Jožef Stefan. Aplikacija služi kot dnevnik premikov in detektira mobilnostne vzorce uporabnikov. Ima približno 10 rednih uporabnikov, večino predstavljajo člani laboratorija.

Knjižnica *NextPin* uporablja pasivno zbiranje podatkov, kar pomeni, da ne zahteva uporabnikove komunikacije z aplikacijo za zajemanje podatkov. Za razliko od klasičnega pasivnega pristopa, ki podatke zajema ves čas ne glede na to, ali se uporabnik premika ali ne, v našem primeru knjižnica vsakih

30 sekund pridobi uporabnikovo GPS lokacijo in informacije o aktivnosti. Če sistemski modul za razpoznavanje aktivnosti zazna vožnjo v vozilu, aplikacija zajame pet sekundni vzorec signala iz pospeškometera. Aplikacija pošlje informacije o lokaciji in aktivnosti ter vzorec iz pospeškometra, če je le-tega zajela, na strežnik za nadaljnjo obdelavo.

Prvi korak pri analizi signala iz pospeškometra je predobdelava. V predobdelavi najprej ponovno vzorčimo signal s frekvenco 100Hz, saj podatke na telefonih z operacijskim sistemom Android zajemamo z najvišjo možno frekvenco. Ponovno vzorčenje nam zagotovi, da imajo vsi primeri natančno 500 točk v časovni vrsti. Amplitude ponovno vzorčenih signalov smo prikazali na Sliki 3.3. Po vzorčenju iz signalov odstranimo konstantno komponento pospeška, ki je posledica gravitacije. Sledimo metodi predlagani v [8]. Dinamično komponento pospeška, ki je posledica premikanja, dobimo tako, da od meritev na posamezni osi znotraj določenega časovnega okna odštejemo povprečje teh meritev. S to metodo lahko poleg dinamične komponente pospeška izračunamo tudi vertikalno in horizontalno komponento, ki ocenjujeta pospešek v vertikalni smeri oziroma pospešek v ravnini. Predobdelane signale smo prikazali na Sliki 3.4 in opisali v Tabeli 3.3.

Predobdelavi sledi luščenje značilk. Uporabljamo značilke iz treh skupin — statistične, frekvenčne in značilke vrhov. Statistične značilke vključujejo povprečje, standardni odklon, poševnost ter različne percentile. Frekvenčne značilke temeljijo na predstavitvi signala v frekvenčnem prostoru, najpogosteje pa se uporablja spektralna gostota pri določeni frekvenci oziroma na določenem frekvenčnem intervalu. Značilke vrhov vključujejo število vrhov v signalu ter njihovo višino, širino in površino, ki jo zavzemajo. Skupine značilk s primeri le-teh so podrobno opisane v Tabeli 2.1. Značilke, ki jih uporabljamo v našem pristopu, so navedene v Tabeli 3.4 skupaj s signali, iz katerih jih izluščimo. Potek dela za vsako skupino značilk je orisan na Sliki 3.5.

Izluščene značilke nato analiziramo s statističnimi testi. Statistični testi nam pomagajo razumeti povezave med različnimi značilkami in signali. S ko-

relacijskim testom preverimo korelacijo med značilkami znotraj iste skupine značilk in med različnimi skupinami značilk. Opazili smo, da je poševnost slabo korelirana z ostalimi statističnimi značilkami, prav tako so med sabo slabo korelirane frekvenčne značilke z izjemo frekvenčnega spektra med 25Hz in 40Hz. Statistične in frekvenčne značilke niso korelirane. Značilke vrhov so korelirane s statističnimi značilkami, ne pa s frekvenčnimi. S pomočjo D'Agostino-Pearsonovega testa smo izvedeli, da značilke za posamezen razred niso normalno porazdeljene. Poleg tega se oblike porazdelitev za nekatere značilke razlikujejo po obliki, kar predstavlja oviro pri Kruskal-Wallisovem H-testu. Kruskal-Wallisov H-test preverja hipotezo enakosti median vzorcev za posamezne razrede. Mediane so se statistično pomembno razlikovale pri povprečjih in poševnostih dinamičnega in horizontalnega pospeška vzdolž posameznih osi.

S pomočjo tega znanja smo definirali nekaj množic značilk, ki smo jih nato uporabili pri klasifikaciji primerov z naključnimi gozdovi, metodo podpornih vektorjev in nevronskimi mrežami. Množice smo oblikovali tako, da bi nam omogočile dodaten vpogled v doprinos posamezne skupine značilk oziroma signala k rezultatu klasifikacije. Množice, ki smo jih definirali, so predstavljene v Tabeli 3.7.

III Eksperimentalna evalvacija

Za evalvacijo predlagane metode smo množico podatkov razdelili na tri podmnožice: učno, testno in validacijsko, glede na čas zajema vzorca. S tem smo se izognili metodološko vprašljivi uporabi vzorcev, ki so bili posneti na istem potovanju, v različnih podmnožicah. Porazdelitev razredov v posameznih množicah je prikazana na Sliki 4.1. Evalvacijski scenarij je prikazan na Sliki 4.2. Model najprej naučimo na učni množici in evalviramo na validacijski množici. Postopek ponovimo z različnimi modelskimi parametri, da najdemo optimalno kombinacijo parametrov. Nato združimo učno in validacijsko množico ter na njej naučimo model z optimalnimi parametri. Model

evalviramo na testni množici.

V fazi evalvacije za vsako prevozno sredstvo preštejemo število pravilno pozitivnih, pravilno negativnih, napačno pozitivnih in napačno negativnih primerov. Te vrednosti uporabimo za izračun mer učinkovitosti. Uporabljamo klasifikacijsko točnost, natančnost, priklic in mero F1. Klasifikacijsko točnost izračunamo za celotno množico, ostale mere pa za vsak razred posebej. Ocene združimo v eno vrednost s pomočjo makro povprečenja. Odločili smo se, da bo naša glavna mera F1 mera, saj želimo zmanjšati število lažno pozitivnih in lažno negativnih primerov za vsak razred.

Najprej smo množico podatkov razvrstili s pomočjo trivialnih klasifikatorjev — večinskega in naključnega klasifikatorja. Rezultati so prikazani v Tabeli 4.2, z obema smo dosegli mero F1 0.30. Za naključni klasifikator smo matriko zamenjav prikazali v Tabeli 4.3.

Prvi netrivialni klasifikator, ki smo ga preizkusili, je naključni gozd. Izbrali smo ga, ker dobro deluje na nelinearnih vzorcih v podatkih. Sprva smo naključne gozdove učili na vnaprej definiranih množicah iz Tabele 3.7. Rezultati so zbrani v Tabeli 4.4. Najboljšo vrednost mere F1, ki znaša 0.42, smo dosegli z množico značilk, ki vsebuje statistične značilke horizontalnega pospeška. Opazili smo, da boljše rezultate dosežemo z množicami, ki ne vsebujejo značilk vrhov.

Da bi izboljšali klasifikacijo, smo podatke transformirali z razčlembo načelnih sestavin (PCA). Prve tri načelne sestavine v vseh množicah značilk pojasnijo več kot 80% variance v podatkih, zato smo se odločili za transformacijo v tridimenzionalni prostor. Z uporabo PCA smo dosegli izboljšanje mere F1 za vsaj 0.05 pri večini vnaprej definiranih množic značilk. Tako kot prej smo najboljšo vrednost mere F1, 0.47, dosegli z množico značilk, ki vsebuje statistične značilke horizontalnega pospeška.

Dodatno smo implementirali izbiro značilk z naključnim gozdom. V prvem eksperimentu smo začeli s celotno množico izluščenih značilk, nato pa smo postopoma izločali značilke, pri katerih smo dosegli boljšo F1 mero, če smo jih izpustili iz množice značilk, uporabljene za klasifikacijo. V drugem

eksperimentu smo začeli z množicami značilk moči 1, nato pa smo v množice postopoma dodajali značilke, ki so pripomogle k izboljšanju mere F1. Mera F1 v odvisnosti od števila značilk pri takšni izbiri je prikazana na Sliki 4.3. Z dodajanjem značilk smo dosegli mero F1 0.48, z odvzemanjem pa 0.49.

Nadaljevali smo z metodo podpornih vektorjev. Tako kot pri naključnih gozdovih smo tudi tokrat z vnaprej definiranimi množicami značilk najboljšo vrednost mere F1, 0.41, dosegli z množico značilk, ki vsebuje statistične značilke horizontalnega pospeška. Enako vrednost smo dosegli pri uporabi še dveh množic značilk, vsi rezultati pa so navedeni v Tabeli 4.10. Opazili smo, da metoda podpornih vektorjev v primerjavi z naključnih gozdom več avtobusov razvrsti pravilno, a je manj zanesljiva pri razvrščanju avtomobilov. Tudi tokrat smo podatke transformirali s PCA, kjer smo opazili podobno izboljšanje kot pri naključnih gozdovih. Rezultati so prikazani v Tabeli 4.12.

Poskusili smo še z binarno klasifikacijo, s katero so pravilno razpoznali 55% avtov, 42% avtobusov in 13% vlakov ob uporabi strategije eden proti ostalim. To je slabše kot v večini eksperimentov z večrazredno klasifikacijo. Opazili smo, da je horizontalen pospešek bolj uporaben pri zaznavi vožnje v avtu in avtobusu, dinamičen pa pri zaznavi vlaka. Iz teh rezultatov sklepamo, da uporaba metode podpornih vektorjev ni optimalna za ta problem.

Nazadnje smo testirali še klasifikacijo z nevronskimi mrežami, pri čemer smo se omejili na mreže z največ tremi plastmi. V povprečju smo dosegli višje vrednosti F1 mere kot pri naključnih gozdovih in metodi podpornih vektorjev. Najboljše rezultate smo dosegli z uporabo značilk dinamičnega pospeška, kar je ravno nasprotno kot pri naključnih gozdovih in metodi podpornih vektorjev. Rezultate smo predstavili v Tabeli 4.15. Matrika zamenjav v Tabeli 4.16 razkriva, da smo z uporabo nevronskih mrež pravilno klasificirali večino primerov dveh prevoznih sredstev. Pravilno smo označili 65% avtov, 63% avtobusov in 18% vlakov, kar je bolje kot v prejšnjih poskusih. Opazili smo, da v tem primeru priklic doseže najvišjo vrednost do sedaj, kar pod vprašanje postavlja našo izbiro glavne metrike.

IV Sklep

V delu smo pokazali, da je mogoče zaznati prevozno sredstvo na podlagi vzorcev, zajetih s senzorjev mobilnega telefona, čeprav je dobre rezultate težko doseči. Naši rezultati so bistveno slabši, kot rezultati o katerih poročajo sorodna dela. Razloge za slabše rezultate iščemo v uporabi krajših vzorcev signalov, saj v večini sorodnih del uporabljajo daljše vzorce. Zaradi uporabe kratkih vzorcev lahko zgrešimo pomembne elemente vožnje, kot na primer ustavljanje pri rdeči luči ali na avtobusni postaji, ter zavijanje v ozko ulico.

Poleg tega uporabnikov nismo omejevali pri položaju naprave med vožnjo ali uporabi naprave. Pospešek je zaradi uporabe telefona ponavadi tri- do petkrat močnejši kot pospešek zaradi gibanja vozila [9]. Kratkih gest odstranitev gravitacije ne more izničiti, zato se nam zdi uporaba daljših vzorcev smiselna. Poleg tega raziskava v smeri optimalne frekvence vzorčenja še ni bila narejena, zato v prihodnje predlagamo eksperimentiranje s frekvenco vzorčenja v fazi predobdelave. Možna je tudi vključitev dodatnih senzorjev, na primer giroskopa in magnetometra. Večina aplikacij za detekcijo prevoznega sredstva že tako ali tako uporablja GPS, zato bi kot značilko lahko uporabili tudi hitrost, ki jo sistem izračuna na podlagi lokacije.

Opazili smo, da klasifikatorji, ki so v osnovi zasnovani za večrazredno klasifikacijo, v našem primeru vračajo napovedi, ki dosežejo višje vrednosti mer uspešnosti. Metoda podpornih vektorjev večrazredno klasifikacijo izvede s pomočjo več binarnih klasifikatorjev, ki uporabljajo strategijo eden proti vsem. V naslednjih delih nas zanima uporaba globokega učenja, ki v določeni meri lahko nadomesti luščenje značilk. Trenutno je oblikovanje značilk prepuščeno raziskovalcem, kar pa pomeni, da značilke pogosto niso optimalne. Poleg tega smo se mi pri nevronskih mrežah omejili na tri plasti, kar pomeni, da je ogromen prostor konfiguracij nevronskih mrež ostal nepreiskan.

Chapter 1

Introduction

Nowadays, smart phones are daily companions of a large fraction of people [10]. Modern applications connect billions of people via social media, enable secure online banking, some applications are even aware of their users' context [10]. Context in this regard is any information that can be used to characterize the situation of an entity – a smart phone user [11]. Context logging and analyzing applications can be used as a personal coach, giving prompts to help a person change her behaviors [12]. A few possible examples of that include reminding a person to talk more or less in company meetings, encouraging them to quit smoking [12] or detecting changes in behavior, which can indicate a mental disorder, such as depression, or an early stage of neurological disorder, for example Alzheimer's or Parkinson's disease [12].

In another scenario, context-aware applications are used to suggest using environmentally more sustainable forms of transportation [7, 13]. Identifying meaningful relationships between people, events, and their environment in group data can help policymakers make better decisions [12] regarding transit network, such as identify or build better bike-commuting paths [12] or improve the public transportation network.

To be able to automatically detect the context for advanced traffic and physical activity monitoring applications efficient and reliable activity detection algorithms are necessary. In the recent years we have witnessed a

drastic increase in sensing and computational resources that are built in mobile phones. Most modern cell phones are equipped with sensors containing triaxial accelerometer, magnetometer, gyroscope, and a GPS. Smart phone operating system APIs offer activity detection modules that can distinguish between different human activities, for example: being still, walking, running, cycling or driving in a vehicle [14, 15]. However, APIs cannot distinguish between driving in different kind of vehicles, for example driving a car or traveling by bus or by train. Recognizing different kind of transportation, also known as transportation mode detection, is crucial for mobility studies, for routing purposes in urban areas where public transportation is often available, to push users towards more environmentally sustainable forms of transportation [7], or to inspire them to exercise more.

This thesis will address transportation mode detection based primarily on mobile phone data using machine learning methods. Our goal is to develop an efficient approach to transportation mode detection that:

1. focuses on near real-time classification of motorized transportation modes — travelling by car, bus, and train;

2. preserves battery life and does not require usage of mobile data;

3. is computationally inexpensive.

Additionally, we aim to evaluate proposed approach on real world data collected by Optimum project [7]. Open research questions include signal preprocessing, feature extraction and selection, choice of evaluation scenario and evaluation metrics, and selection of appropriate classifier. Expected contributions are:

1. our unique approach to data collection,

2. the features that we use and comprehensive analysis of their contribution to classification outcome, and

3. our clear and strict evaluation scenario.

The first contribution of this thesis is our approach to data collection. Although we did not develop the *NextPin* [6] library we used to collect the data, we were heavily involved in the design of the data collection pipeline that is used in the library. To collect sensor samples we use passive logging approach. Passive logging assumes there is no user's effort involved in data collection, however as we are using supervised learning methods, user's effort is required to annotate the samples. We rely on activity recognition APIs to know when the user is traveling in a vehicle, which makes our approach to data acquisition for travel mode detection unique. This, combined with the use of short samples of accelerometer and magnetometer measurements, reduces consumption of power and mobile data. We use sensor samples obtained in pilot testing for a multi-modal routing mobile application. The difference between our study and related work is that our test users are not in any way instructed on the desired position of their mobile phone, or told to not interact with their device during the trips.

The second novelty in our approach are the features we use. Not only we use features from three domains — statistical, frequency-based, and peak-based — which was rarely done previously, some of these features were predominantly used on longer accelerometer samples. Additionally, we offer an in-depth analysis of the extracted features. We also analyze the their influence on the classification performance. We do that using several different feature sets. We define feature sets using the results of the feature analysis in combination with features used in similar work.

Finally, we clearly define and state our evaluation scenario. We use three strongly separated sets to evaluate our approach to avoid the contamination of the test set. As a performance measure we use F1 score, which is not as common, though it is very appropriate for this domain.

This thesis has the following structure: Section 2 is an overview of the existing related works. In Section 3 we describe our approach to data acquisition and feature extraction in detail, and offer a brief analysis of the extracted features. Section 4 focuses on evaluation of the proposed approach.

We discuss the results and draw conclusions in Section 5.

Chapter 2

Related work

Ever since smart phones appeared and gained widespread adoption there have been many research efforts for their use in activity recognition and transportation mode detection. While the first attempts to recognize user activity were initiated before smart phones, the real effort in that direction begun with the development of mobile phones having built-in sensors [1], including GPS and accelerometer sensors. There are still some studies that use custom loggers to collect the data [2, 16] or use dedicated devices as well as smart phones [9]. Although GSM triangulation and local area wireless technology (Wi-Fi) can be employed for the purpose of transportation mode detection, their accuracy is relatively low compared to GPS [2], so state of the art research is focused on transportation mode detection based on GPS traces and/or accelerometer data.

Since smart phone sensors were originally developed for human-computer interaction purposes, using them to infer the mode of transport is challenging. For example, accelerometer measures acceleration in a limited range with a limited precision. It turns out that the range and precision are device-dependent and may differ significantly between different phones. Additionally, there are no restrictions on phone placement, users may even play a game that requires tilting and turning the phone while traveling as a passenger in a car, on a bus, or on a train. Acceleration caused by the user's movement

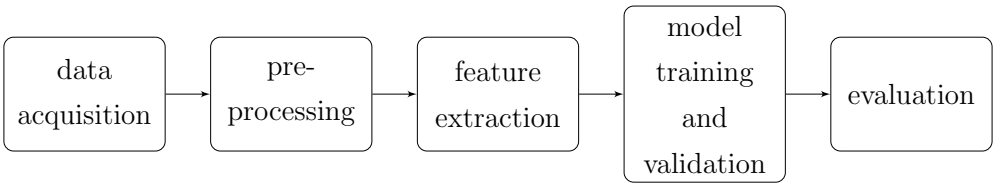

Figure 2.1: General work-flow schema used for transportation mode detection.

is usually three to five times stronger than that of a vehicle [9], which makes the problem of transportation mode inference especially difficult.

Most often the set of possible outcomes of classification includes up to six modalities, where usually up to two of them are non-motorized, for example still, walk or bike [2, 4, 16, 17, 18, 3]. Common motorized modalities considered are car, bus and train [2, 5, 4, 17, 19, 9], however vehicles like subway [5, 20, 3, 19], tram [5, 3], scooter or motocycle [5, 9], light rail [9] and high speed rail [19] appear as well.

In general, approaches follow work flow pictured in Figure 2.1. We briefly addressed means of data acquisition in the first paragraph of Chapter 2. Data is collected by sampling built-in sensors. Labels for supervised training are often collected together with the data. Normally data acquisition is followed by a preprocessing step. Pre-processing includes filtering out the noise. In this case inaccurate readings in the data originate from the sensors themselves, environmental factors such as clouds, the gravitational force, metallic objects or are caused by user's movement and interaction with the device. Mistakes when labeling also occur. Depending on the sensor used and the source of the noise different approach to filtering is applied to the data. Pre-processed data is used for feature extraction. Most common are hand-crafted features. Feature vectors and labels are further used to train the models and evaluate the approach.

Machine learning methods that are most commonly used in accelerometer based modality detection include support vector machines, decision trees and random forests, however naïve Bayes, Bayesian networks and neural networks

have been used as well [2, 4]. Often these classifiers are used in an ensemble [5]. The majority of algorithms additionally use Adaptive Boosting or Hidden Markov Model to improve the performance of the methods mentioned above [5, 3, 2, 1]. Features used for machine learning can be divided into five groups: Statistical, Time, Frequency, Peak and Segment [3], however in most cases statistical features and features based in frequency are used [1, 2, 5]. In the last years, deep learning has also been used [21, 22].

Studies that only use GPS traces often combine that data with available GIS sources, for example OpenStreetMap (OSM) database [4, 17], or with real-time public transport locations, when such information is publicly available [4]. In these cases classification features usually include average bus closeness, average rail closeness, etc. [4], or are based on the distances to various infrastructural objects such as railway, motorway, bus and train stations [17]. These approaches can achieve classification accuracy over 90% [4, 17], however the main drawback for their use is the limited scalability. Since features for classification in this case include distances between user and real-time bus locations, rail line trajectories, and bus stops, doing linear comparison with all these locations is time consuming [4]. Stenneth et al. [4] report that for the Chicago, Illinois area linear comparison took over two minutes, while they were only able to reduce the feature creation time to ten seconds with zip-code based indexing and pruning using reverse geocoding to precompute zip-codes of bus stops, bus routes, and train routes [4]. That also suggests that these approaches are not appropriate for real-time applications, where information of travel mode is expected to be ready for use in less than a second.

Additionally, there have been studies that use GPS trajectories to extract speed and acceleration based features [16, 18]. Such features include heading change rate, stop rate and velocity change rate [16]. Frequency components of speed and acceleration can also be computed using fast Fourier transformation [18]. Since GPS signal might not be available all the time these approaches often use filtering and re-sampling to ensure the quality of

the data [18]. These approaches report recall score of 0.76 [16] and F1 score around 0.88 [18], which suggests that using features based on frequency spectrum increases performance of the system.

The performance of transportation mode detection systems depends on the effectiveness of handcrafted features designed by the researchers, researcher's experience in the field affects the results [23]. Thus, there have been approaches that use deep learning methods, such as autoencoder or convolutional neural network, to learn the features used for classification [23]. Using a combination of handcrafted and deep features for classification with deep neural network resulted in 74.1% classification accuracy [23]. Although the results seem worse than when GPS traces are combined with GIS data, studies used different measurements for evaluation and sometimes do not report on the class representation of the data.

Approaches that rely solely on GPS trajectories require GPS signal of high quality, but the GPS receivers of smart phones are generally severely shielded during daily activities [5]. This occurs during underground travel, inside stations, or even when users are not sufficiently close to a window when travelling in a vehicle [3]. In all of these cases the result is loss of positioning information. There have been successful attempts to combine GPS data with accelerometer measurements to deal with unreliable GPS signal [5]. Additionally to accelerometer data, the proposed use of Kalman filter to smooth out the trajectory of movement results in precision and recall of 0.76 when classifying with an ensemble, followed by a discrete hidden Markov model [5].

Another known issue when using GPS signals from mobile devices is the high power consumption [3], which is especially displeasing in case of longer commutes. Both of these issues suggest that accelerometer sensor data is more appropriate for activity detection, thus accelerometer-only approaches are common [3].

As accelerometer measures acceleration in three directions, orientation of the device and its usage during travel affect the data. Thus, some researchers

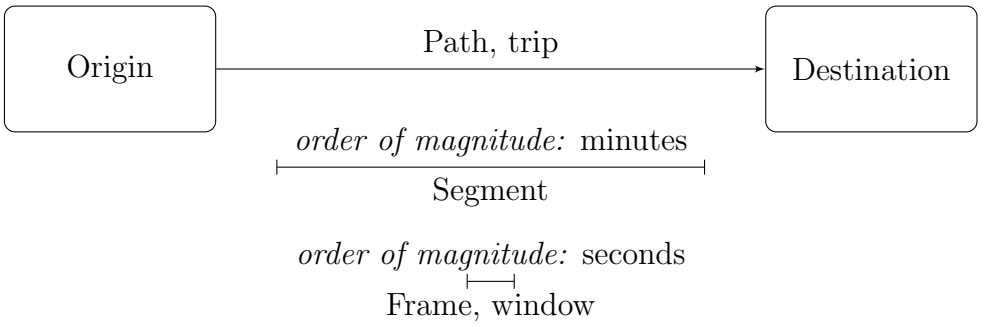

Figure 2.2: The difference in time scale between paths or trips, segments, and time frames or windows.

instruct users to keep the devices in the same position throughout the trip so that acceleration in different directions can be judged of easily [2]. However, this is not feasible when transportation mode detection is used as a part of routing application, since the users are expected to interact with their phones during the trip. To tackle this problem, filtering and gravity estimation methods are often employed [3] in the preprocessing step.

Accelerometer measurements can be recorded throughout the whole path or trip, on shorter segments, or even on a level of time frames and windows. Sometimes the recordings are split into shorter parts during the preprocessing. The difference in time scale between paths, segments, and frames is shown in Figure 2.2. Depending on the length of the trip and data acquisition strategy, segments might sometimes cover the length of the trip.

Features can be divided into five domains based on how they are computed [3]. These domains are statistical, time-based, frequency-based (spectral), peak-based and segment-based and are described in Table 2.1. Statistical, time-based, and spectral attributes are computed on a level of a time frame that usually covers a few seconds, whereas peak-based features are calculated from peaks in acceleration or deceleration. On the other hand, segment-based features are computed on the recordings of the whole trip, which means that they cover much larger scale [3]. Statistical, time-based, and spectral features are able to capture the characteristics of high-frequency motion caused by

Domain	Description
Statistical	These features include mean, standard deviation, variance, median, minimum, maximum, range, interquartile range, skewness, kurtosis, root mean square.
Time	Time-based features include integral and double integral of signal over time, which corresponds to speed gained and distance traveled during that recording. Other examples of time-based features are for example autocorrelation, zero crossings and mean crossings rate.
Frequency	Frequency-based features include spectral energy, spectral entropy, spectrum peak position, wavelet entropy and wavelet coefficients. These can be computed on whole spectrum or only on specific parts, for example spectral energy below 50Hz. Spectrum is usually computed using fast Fourier transform, whereas wavelet is a result of the Wavelet transformation [24]. Entropy measures are based on the information entropy of the spectrum or wavelet [24].
Peak	Peak-based features use horizontal acceleration projection to characterize acceleration and deceleration periods. These features include volume, intensity, length, skewness and kurtosis.
Segment	Segment-based features include peak frequency, stationary duration, variance of peak features, and stationary frequency. The latter two are similar to velocity change rate and stopping rate used by [16]. Segment-based features are computed on a larger scale than statistical, time-based or frequency-based features.

Table 2.1: Feature domains and their descriptions adopted from [3].

user's physical movement, vehicle's engine and contact between wheels and surface [3]. Peak-based features capture movement with lower frequencies, such as acceleration and breaking periods, which are essential for distinguishing different motorized modalities [3]. Additionally, segment-based features describe patterns of such acceleration and deceleration periods [3].

Accelerometer-only approach where only statistical features have been used reported 99.8% classification accuracy, however users were instructed to keep devices in fixed position during a trip [2]. Furthermore, only 0.7% of data was labeled as train [2]. State of the art approach to accelerometer-only transportation mode detection relies on long accelerometer samples [3]. It uses features from all five domains for classification with AdaBoost using decision trees as a weak classifier and achieves 80.1% precision and 82.1% recall [3].

There have also been attempts of leveraging other smart phone sensors, for example magnetometer [9], gyroscope[19] and microphone [20] in combination with accelerometer data to detect the mode of transportation. The main concern when using microphone readings is privacy of users, therefore very short samples, ranging from 0.1s to 1s, are collected to ensure that no word is understood [20]. Sound and magnetometer recordings have different fingerprints in frequency space [9, 20], when driving in different types of vehicles, which indicates that they carry information about the modality of transport. In case of magnetometer the change in magnetic field is due to rotation of metal parts of engine, transmission system and wheels [9]. Dominant frequency correlates with the speed of a vehicle and decreases when the size of wheels increases [9]. Similar holds for microphone recordings, however as the system was only tested on buses with compressed natural gas engines and cars with liquefied natural gas engines [20]. Approach aided by microphone readings yielded classification accuracy of 98.2% with SVM classifier [20], whereas approach with magnetic field measurements achieved F1 score of 94.5% when using random forest and neural network [9]. Additionally, gyroscope was also used in combination with accelerometer and magnetometer

[19]. Although only 14 features have been used in kNN approach, it resulted in 98.2% classification accuracy [19].

In the recent years, the problem of transportation mode detection is experiencing a transition from GPS-based approaches using special devices to sensor-based approaches, that rely on smart phones. This can be observed through the lack of relevant data sets for sensor-based classification and basically non-existence of mobile application that recognize user's transportation mode. To our knowledge there does not exist a mobile application or a service that recognizes motorized modalities at such granular scale.

This thesis is most similar to our previous work presented in [25] and [13], whereas we have presented a part of this thesis in [26]. The main difference between the thesis and our previous work is in the evaluation scenario as this time we opted for much stricter schema. Additionally, in this work we use the data collected since the prototype of our previous work has been deployed.

Chapter 3

Proposed approach

Our proposed approach closely follows general work flow diagram for transportation mode detection shown in Figure 2.1. General tasks in transportation mode detection include data acquisition, data preprocessing, feature extraction, model training and validation, and evaluation, however this chapter does not cover model training and validation, and evaluation, as we will discuss those in the next chapter. We modified the general approach to include feature analysis.

We pictured the proposed approach in Figure 3.1, where we stacked high level task mostly taken from Figure 2.1 vertically, and specific subtasks horizontally. In this chapter we first discuss data acquisition through mobile applications. This covers methodology used to collect the data, incentives given to the participants, and the amount of data we have acquired. Next we address preprocessing of the data, including resampling, filtering, and gravity estimation. The third step is feature extraction with an explanation of the features we use and technical details of the extraction process, followed by feature analysis. In this step, we use correlation analysis and statistical tools to inspect the extracted features. Finally, we use the knowledge about the features to predefine different feature sets and choose appropriate machine learning methods for classification.

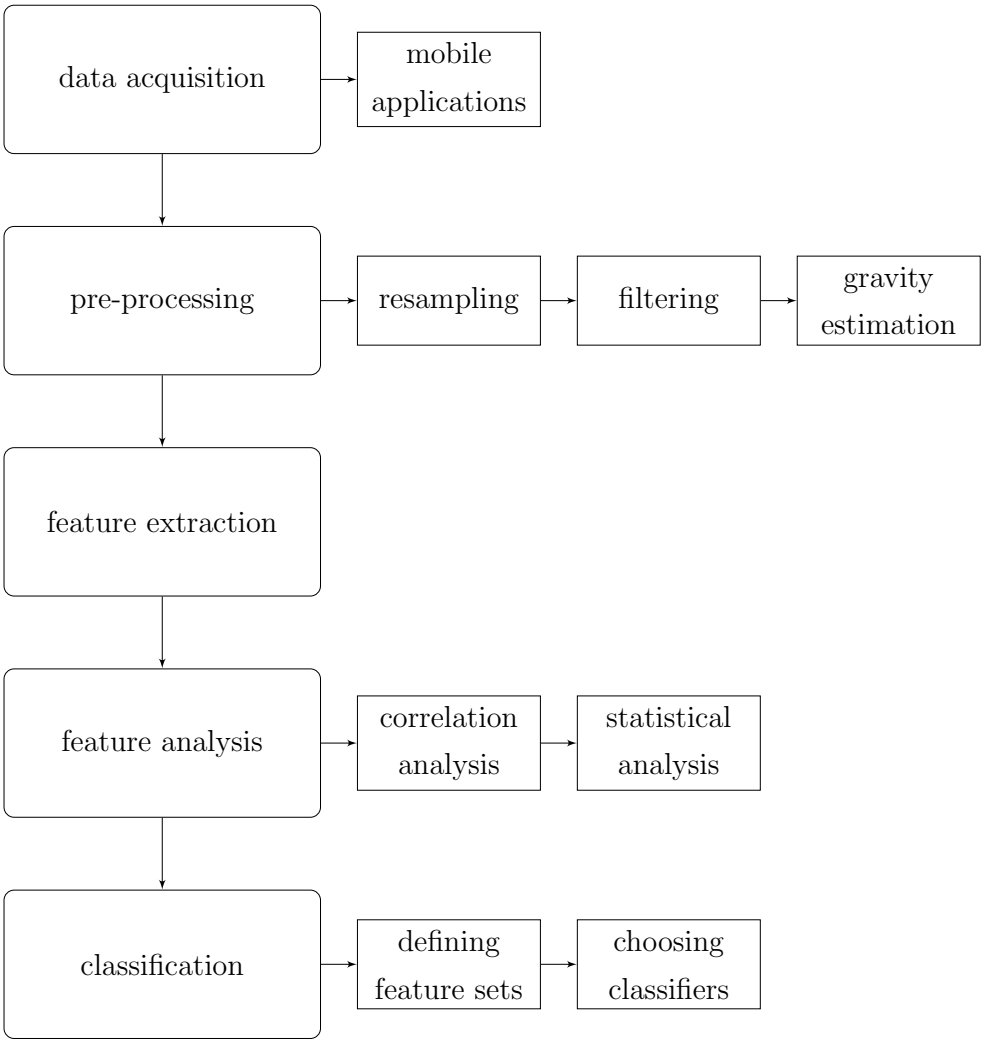

Figure 3.1: Detailed work flow diagram of the proposed approach. We stacked general, high-level tasks from Figure 2.1 vertically, whereas subtasks specific to our approach are pictured horizontally.

3.1 Data acquisition

An important source of much needed data about traffic and commutes in a community are studies and surveys on transportation and/or mobility. Travel surveys enable transportation researchers, engineers, and governors to study human behavior, as well as plan, design, and manage the transportation system [27]. There are two main categories of travel study methods. In the first one, respondents are asked to provide details of their trips based on memory. This also includes traditional pen and paper travel journals. Studies have shown that start and end times of the trips are usually approximate. Furthermore, people's perceptions of in-vehicle time vary according to different modes of travel — travel time in car is typically underestimated, whereas travel time in public transport is overestimated. The second category includes methods where data is recorded automatically by devices placed at fixed locations or carried by respondents themselves [2].

Methods of automatic data collection can be further divided into two classes — passive logging and active logging. Passive logging does not require any effort from the user, however since it is monitoring the sensors and collecting the data all the time whether the user is moving or not, it is known to drain the battery fast. On the contrary, active logging is collecting the data when user signals the data collection application to record the sensor data. Thus, active logging is more battery-friendly as the data collection is controlled by the user. The main drawback of active logging is that it requires user's involvement, which some might find burdensome. Active logging can be used as ground truth for development of supervised machine learning methods [17], whereas passive logging is more convenient when observing the transitions between different activities.

Devices used for automatic collection of the data can be smart phones or specially designed small, possibly wearable, devices with multiple sensors. Use of dedicated devices was common prior to the wide adoption of smart phones. Increased sensing capabilities of smart phones combined with their market penetration, easy programmability, and effective distribution chan-

nels for third party apps, have contributed to smart phones maturing into an effective tool for unobtrusive monitoring of travel behavior [27]. Additionally, carrying a smart phone has become a habit and is therefore less of a burden. Thus, the risk of non-reported trips is reduced compared to special devices [17].

3.1.1 NextPin

Since the subject of this thesis is transportation mode detection based on mobile sensor data we use smart phone applications to acquire data. The Artificial Intelligence Laboratory at the Jožef Stefan Institute developed *NextPin* [28, 6] software in form of a mobile OS programing library for Android and iOS to collect geospatial and sensor data from mobile phone. The behavior of the library when built in the mobile application and its connection to analytics platform on the server are shown in Figure 3.2.

NextPin obtains a user's GPS location and activity every 30 seconds. To determine user's activity the library uses OS's native activity recognition modules, Google's ActivityRecognition API [14] for Android and Apple's CMMotionActivity API [15] for iOS, which can detect walking, running, cycling, staying still and driving in a vehicle. Every time the phone's activity recognition system detects that the user is traveling in a vehicle, we collect a five second sample of accelerometer signal for fine-grained classification of motorized means of transportation. We use short samples in order to reduce the computation time and have faster response time for real-time classification. This also preserves battery life, saves space and reduces the usage of mobile data [3]. Mobile application gathers GPS location, user's activity, and accelerometer sample when taken into a JSON record and sends it to a server at the Jožef Stefan Institute for processing and analysis. That means that every time a record is sent from a phone it includes GPS coordinates and activity information from activity recognition modules, however accelerometer sample is only present when there is a non-zero probability of user traveling in a vehicle according to the native activity recognition API. At this point

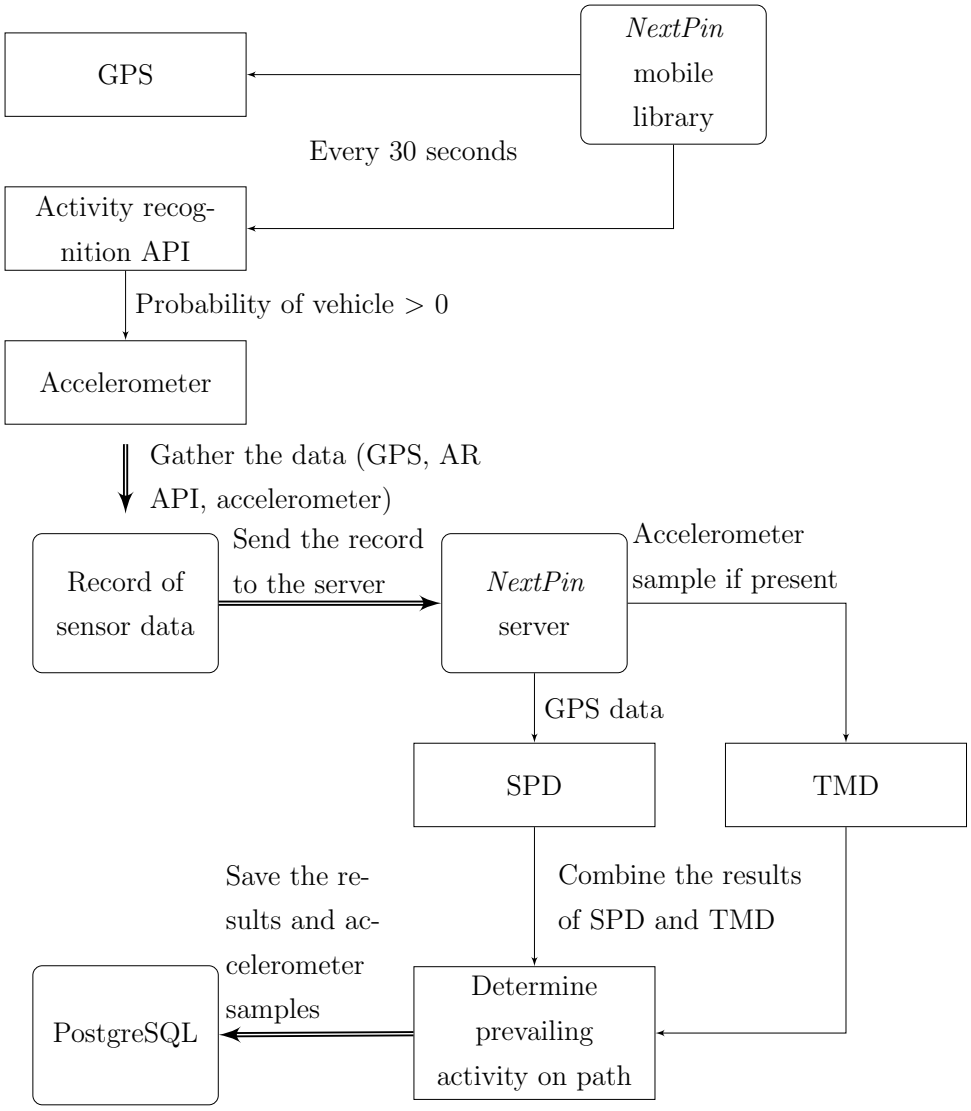

Figure 3.2: Work flow diagram for *NextPin* library and server analytics.

we would like to emphasize that although the library also collects GPS locations, our approach relies exclusively on low-cost and privacy-respectful sensor data.

The main part of server-side processing and analysis is stay-point detection (SPD). SPD performs spatio-temporal clustering on incoming GPS coordinates [29]. If a user spends more than five minutes within 120 meter radius, SPD algorithm detects a cluster and labels it as a stay-point. Coordinates that cannot be clustered connect two stay-points and are labeled as paths or trips. If sensor sample is present in a record, we send the sample to transportation mode detection (TMD) module. In TMD module accelerometer sample is preprocessed according to our preprocessing strategy, we then extract features and feed them to a pretrained classifier for classification into three motorized modalities — car, bus, and train. We replace the general vehicle modality in activity information with the output of classification. As a post-processing part of SPD we compute the prevailing activity on the paths as every GPS coordinate is sent together with activity information and in some cases sensor samples.

Intuitively, we want the prevailing activity to be the modality used to travel the most in that path. There are at least two possibilities of calculating this. The first one is average, which in our case means calculating the activity that was used most of the time as we collect the measurements every 30 seconds. However, the distance traveled in 30 seconds differs a lot depending on the modality and speed. For example, a user has to walk for five minutes to the nearest bus station, then rides a bus for seven minutes, and then walks another five minutes to the final destination. Averaging finds walking as the prevailing activity, although out of five kilometers of the path the user walked less than one kilometer. Therefore, we opted for the second option, which is weighted average. To determine the prevailing activity we construct an ordered vector of activity probabilities $\mathbf{p}$ and calculate a weighted average $\overline{\mathbf{p}}$ of probabilities as

$$\overline{\mathbf{p}} = \frac{\sum_{i \in path} d_i \mathbf{p}_i}{\sum_{i \in path} d_i} \tag{3.1}$$

We use distances d_i between two sequential GPS coordinates as weights.

To detect the change of activity we also store a moving weighted average of the last three records as $\overline{\mathbf{p}}_3$. When cosine similarity of the weighted average for the whole path and the moving average for the last three activities falls below a threshold (set at 0.8) we split the path in two parts. We continue splitting the path until we have seen all records. For each part of the path, the prevailing activity is the one with the largest average probability in activity vector $\mathbf{p}$. A simplified example of path splitting algorithm is shown in Table 3.1.

We save the results to PostgreSQL database. For each path we save the detected activity and all sensor samples that were sent with the GPS locations on this path, among other information. If the detected activity is incorrect, users have an option to edit it in mobile application or via web GUI. By editing trip information we obtain labels for supervised learning and evaluation of transportation mode detection based on sensor samples. It should be noted that usually multiple sensor samples are associated with a trip.

3.1.2 Data collection

NextPin library is integrated in two free mobile applications. *OPTIMUM Intelligent Mobility* [30] Android application is developed in scope of Optimum Project [7]. The main functionality is a proactive multimodal route planner for three European cities (Birmingham, Ljubljana and Vienna), that were chosen as sites for pilot testing. During six-weeks period of pilot testing in May and June 2018, approximately 120 users were asked to use the application daily. Test users were awarded points that corresponded to minutes they have spent walking, cycling or using the public transport, as one of the goals of the project was to promote environmentally sustainable means of transportation. In case of Birmingham and Vienna, the awarded points were then converted into monetary compensation as an incentive, whereas test users in Ljubljana were provided with free bus passes for the duration of the

i	$\mathbf{p}_i$	d_i	$\overline{\mathbf{p}}$	$\overline{\mathbf{p}}_3$	Sim.
0	[100, 0, 0, 0]	100	[100, 0, 0, 0]	[100, 0, 0, 0]	1.00
1	[75, 15, 0, 10]	100	[88, 7, 0, 5]	[88, 7, 0, 5]	1.00
2	[90, 5, 0, 5]	100	[88, 7, 0, 5]	[88, 7, 0, 5]	1.00
3	[50, 10, 5, 35]	100	[79, 7, 1, 13]	[72, 10, 2, 16]	0.99
4	[20, 5, 0, 75]	100	[67, 7, 1, 25]	[53, 7, 2, 38]	0.96
5	[10, 0, 5, 85]	100	[57, 6, 2, 35]	[27, 5, 3, 65]	0.80
6	[5, 0, 75, 20]	100	[50, 5, 12, 33]	[12, 2, 26, 60]	**0.71**
7	[5, 5, 85, 5]	100	[10, 3, 41, 46]	[7, 2, 55, 36]	0.97
8	[30, 0, 70, 0]	100	[14, 2, 47, 37]	[13, 2, 77, 8]	0.85
9	[100, 0, 0, 0]	100	[28, 2, 39, 31]	[45, 2, 51, 2]	0.85
10	[100, 0, 0, 0]	100	[39, 1, 33, 27]	[77, 0, 23, 0]	0.81
11	[100, 0, 0, 0]	100	[46, 1, 30, 23]	[100, 0, 0, 0]	**0.78**

Table 3.1: An illustrative example of path splitting. Due to readability we are using activity vectors of length 4 and only show integers. Threshold for splitting is set to 0.8. Cosine similarity (last column) of averages of vectors 0 to 6 and 4 to 6 is below the threshold so that means that activities 4 to 6 belong to a different path than activities 0 to 3. Activities 0 to 3 form a new, finished path with prevailing activity 1. For activity 7 the average is calculated from activity 4 on. Cosine similarity of averages of activity vectors 4 to 11 and 9 to 11 is also bellow threshold of 0.8 so that path is also split into two – the first one consisting of activities 4 to 8 and the second one of 9 to 11. As there are no more activities left the procedure is finished.

	Car	**Bus**	**Train**
Trip count	272	99	98
Sample count	2864	840	516

Table 3.2: Counts of trips and samples between 1 February 2018 and 15 June 2018.

testing period.

Mobility Patterns [31, 32] application for Android and iOS is developed by the Artificial Intelligence Laboratory at the Jožef Stefan Institute to collect movement of its users and detect underlying mobility patterns. Similarly to OPTIMUM Intelligent Mobility application, it uses *NextPin* library to monitor the sensors and send the data to the same server. The application serves as a travel journal, though it also predicts the next location. Mobility Patterns application has up to ten regular users, the vast majority of them are members of our laboratory. None of the Mobility Patterns users were given any incentive or compensation for their cooperation in application testing and data collection.

During the testing period between 1 February 2018 and 15 June 2018 we have collected 469 labeled trips with 4220 sensor samples from Mobility Patterns and OPTIMUM Intelligent Mobility users. Counts of trips and samples for each mode are listed in Table 3.2. These samples form a *pilot testing* dataset.

The numbers of collected trips and samples, 469 and 4220, respectively, suggest that recorded trips are short. Since we monitor GPS and activity recognition modules every 30 seconds, approximately 9 samples per trip implies that trips lasted around 5 minutes on average. However, since we only take accelerometer samples when the activity recognition system detects that the user is traveling in a vehicle, stopping at the traffic lights, bus stops, and train stations affects the number of collected samples per trip.

Since the number of samples is rather small for a machine learning project, we extended that dataset with the data we used in our previous work [25].

To collect that data a few members of the Artificial Intelligence Laboratory at the Jožef Stefan Institute used a modified Mobility patterns mobile application. Again, no incentives were given to the users. The main difference in these data sets is the methodology of collecting the data. The modified application used active logging approach to continuously record the acceleration, whereas in the regular *Mobility Patterns* application uses passive logging. The majority of the data was collected in August 2016 in Ljubljana and surrounding area. This data includes additional 16824 five second accelerometer samples. Out of these 16824 samples, 8529 are examples of driving in a car, 5438 were collected while traveling by bus, and 2857 were taken during a train ride. These samples form *Mobility Patterns* dataset.

3.2 Preprocessing

The first step in preprocessing is resampling. We collect five second samples of sensor data and resample them to sampling frequency 100 Hz. Android phones collect the sensor data with the highest frequency available for a specific phone. Since Android phones differ a lot in hardware they have installed, using the highest available sampling frequency contributes to the quality of the data when dealing with an older or cheaper model of the phone. Phones running iOS record the data with a sampling frequency of 100 Hz. We found that hardware in iOS phones differs less. Resampling ensures us that our samples all contain 500 measurements. Amplitudes of resampled accelerometer data for each mode are pictured in Figure 3.3. Samples used to produce Figure 3.3 were taken from different users and recorded in different phone orientations. Graphs show that acceleration patterns are different for each mode, thus suggesting that accelerometer data is appropriate for transportation mode classification.

When sampling the sensors we inevitably capture some noise along with the signal. In case of transportation, noise comes from sensors themselves, accelerometer is affected by gravity and user's movement and interaction

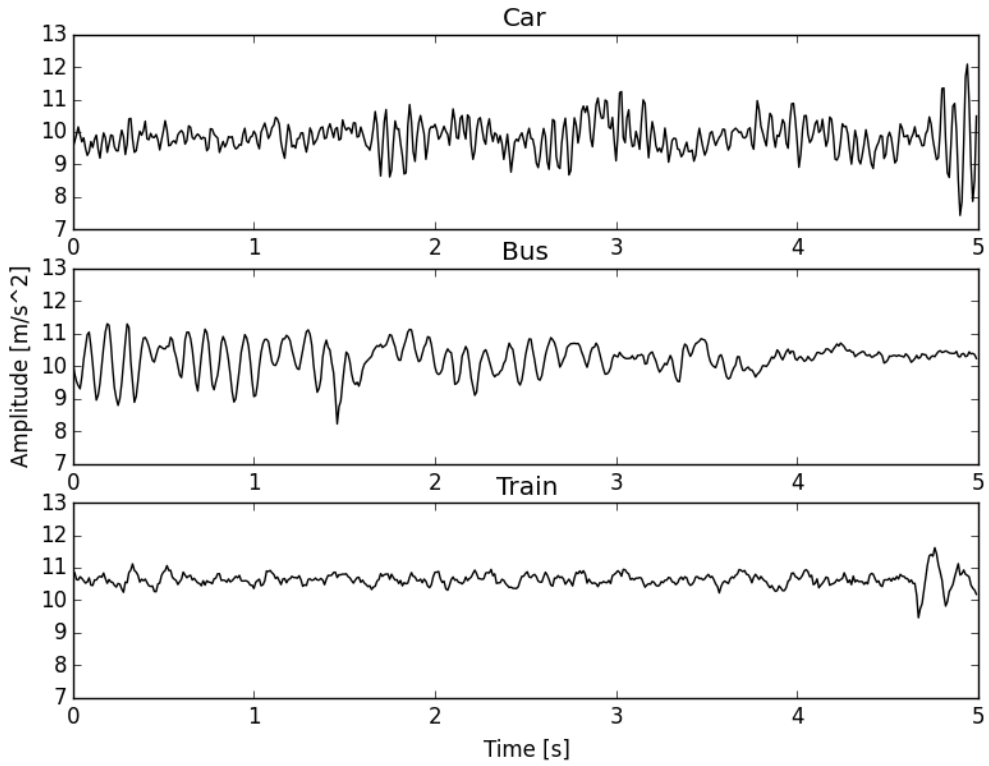

Figure 3.3: Amplitudes of resampled accelerometer data for each mode. Samples were taken from different users and recorded in different phone orientation.

with the device, whereas metallic objects and electromagnetic fields influence magnetometer. To reduce the effects of noise, filtering is often applied. Low-pass filter removes high frequency noise that usually originates in sensor, while high-pass removes low frequency noise that is caused by user. However, it is used more on magnetometer readings than accelerometer data.

There are some strong arguments against the use of filters on accelerometer signal. Firstly, we can learn something about user's travel mode based on user's interaction with the phone. If a person is using the phone they are more likely to travel by bus or by train rather than driving a car. Secondly, car engines operate between 2500 rpm and 3000 rpm at cruising speed, which is 40 - 50 Hz in terms of frequency. As we are sampling the sensors with 100 Hz, this range is right at the top of our spectrum, that is defined by the Nyquist sampling theorem. Therefore, filtering with low-pass filter is not justified because it can remove the signal of the movement, often vibration, of the engine. Thirdly, as the car is traveling at around 50 km/h, the wheels are turning with frequency around 7 Hz. This means that high-pass filters are inappropriate for the problem at hand as they filter out low frequency movement, caused by the turning wheels. All these suggest that filtering might not be justified as we might filter out vibrations that are specific to a certain mode.

A more prominent problem we face concerns the correlation of acceleration measurements with the orientation of the phone in the three dimensional space. In practice this means that gravity is measured together with the dynamic acceleration caused by phone movements. Thus, several wearable computing or ubiquitous computing research projects have detected and distinguished user motion activities by attaching accelerometers in known positions and orientations on the user's body [2, 8]. To bypass that limitation we perform gravity estimation on raw accelerometer signal. We follow an approach proposed by Mizell [8]. Gravity estimation splits the acceleration into static and dynamic components. Static component represents the constant acceleration, caused by the natural force of gravity, whereas dynamic

component is a result of user's motion. Furthermore, using this approach we are able to calculate vertical and horizontal components of acceleration.

3.2.1 Gravity estimation

Acceleration measurements taken at a given point in the interval form acceleration vector $\mathbf{a} = (a_x, a_y, a_z)$. The algorithm works as follows: we obtain the estimation of gravity $\mathbf{g} = (g_x, g_y, g_z)$ by averaging all accelerometer measurements on those respective axes for the time interval [8]. As we collect five second samples of accelerometer readings, we opted for one second time interval. Although this means that we might recognize some of the acceleration as gravity, for example a train might accelerate over a period of ten seconds or even more, we are more interested in low-amplitude acceleration caused by the vibrations of the engine, the contact between the wheels and the ground, and road infrastructure. We chose one second time interval to attempt to eliminate the effects of user's actions. By calculating average for each second we remove acceleration caused by flipping the phone around or taking it out of the bag. Vector $\mathbf{g}$ also corresponds to vertical axis. We calculate the dynamic component of acceleration at a given point in time as $\mathbf{d} = (d_x, d_y, d_z) = (a_x - g_x, a_y - g_y, a_z - g_z)$.

In addition to gravity-free acceleration in direction of x, y, and z, we can also use vertical and horizontal components of acceleration. Vertical acceleration describes the movement in up and down direction. It changes significantly with user's gestures, for example picking up the phone, or when the vehicle is driving over a speed bump or a hole in the road. Horizontal acceleration represents the movement in the plane parallel to the ground.

To obtain vertical component of acceleration $\mathbf{v}$, we compute the projection of the dynamic acceleration vector $\mathbf{d}$ upon vertical axis $\mathbf{g}$ as [8]

$$\mathbf{v} = \left(\frac{\mathbf{d} \cdot \mathbf{g}}{\mathbf{g} \cdot \mathbf{g}} \right) \mathbf{g}. \tag{3.2}$$

Horizontal component of the dynamic acceleration $\mathbf{h}$ is computed as vector subtraction $\mathbf{h} = \mathbf{d} - \mathbf{v}$ [8].

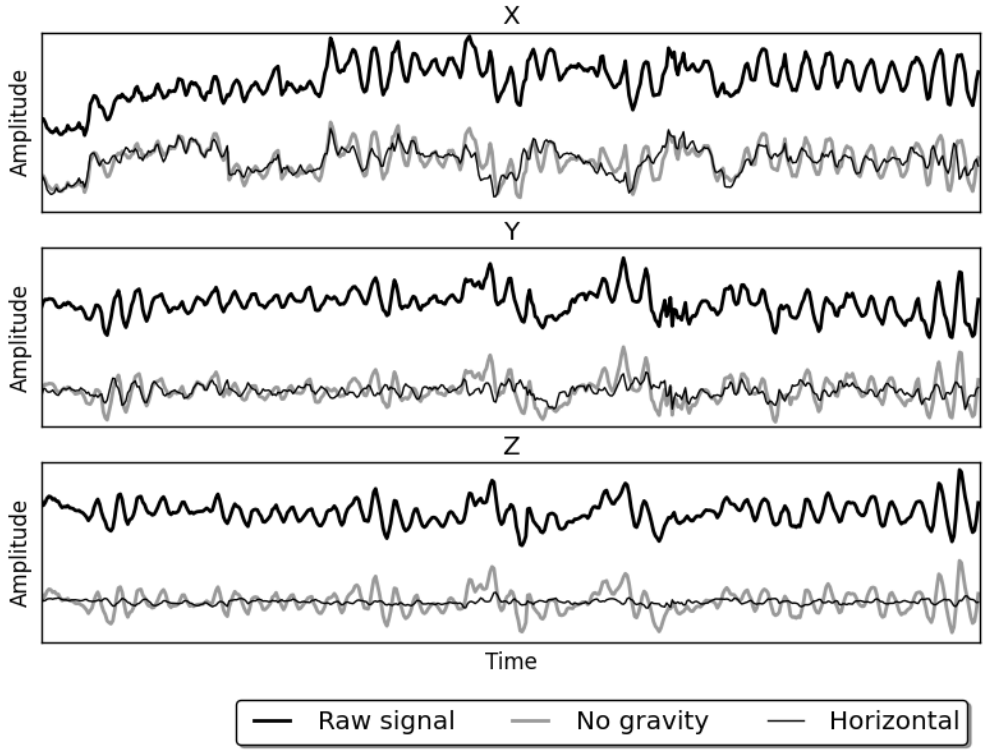

Figure 3.4: The difference between raw signal, signal with eliminated gravity, and horizontal component of the dynamic acceleration on each axis.

The difference between raw signal, signal with eliminated gravity, and horizontal component of dynamic acceleration is pictured in Figure 3.4. We noticed that there were no major differences in acceleration patterns in raw signal and signal with eliminated gravity, whereas patterns in the horizontal component differ from the original signal. This is most notable on z-axis.

Gravity estimation on acceleration signals x, y, and z results in dynamic acceleration signals d_x, d_y, d_z, and amplitude of the dynamic acceleration vector d. We additionally split the acceleration on the amplitude of the vertical acceleration v and amplitude of the horizontal acceleration h with acceleration signals h_x, h_y, and h_z. We also calculate amplitude of raw acceleration a. We noticed that we can divide these signals into two categories — amplitudes and directional signals. Directional signals measure accelera-

Category	Signal	Description
Directional signals	d_x	x-axis of dynamic acceleration vector **d**.
	d_y	y-axis of dynamic acceleration vector **d**.
	d_z	z-axis of dynamic acceleration vector **d**.
	h_x	x-axis of horizontal acceleration vector **h**.
	h_y	y-axis of horizontal acceleration vector **h**.
	h_z	z-axis of horizontal acceleration vector **h**.
Amplitudes	v	Amplitude of vertical acceleration vector **v**.
	h	Amplitude of horizontal acceleration vector **h**.
	a	Amplitude of raw acceleration vector **a**.
	d	Amplitude of dynamic acceleration vector **d**.

Table 3.3: Signals with their descriptions and categories.

tion along one of the three axes and cover the whole range of real numbers, whereas amplitudes are greater or equal to zero. This categorization will be useful in feature extraction. Signals with descriptions and their categories are listed in Table 3.3.

3.3 Feature extraction

We use preprocessed signal to extract features for classification. Features are divided into five domains, based on their meaning and method of computation. The domains are listed in Table 2.1 and described in Chapter 2. We extract features from three domains — statistical, frequency, and peak. Work flow for feature extraction is visualized in Figure 3.5. We opted against the use of segment features as we use only five second samples, which are conceptually closer to frames rather than segments of movement as in [3]. Time-based features are often integrals of signal over time. In discrete settings, like in our case, integrals are very similar to sums and averages, therefore we use statistical features, which cover a larger variety of features. Features are listed in Table 3.4.

Domain	Feature	Computed on
Statistical	Mean Standard deviation 5th percentile 95th percentile Skewness	Amplitudes and directional signals. Directional signals are also split into acceleration and deceleration moments.
	Absolute maximal value	Amplitudes and directional signals.
Frequency	Spectral power $[0.2, 5)$ Hz Spectral power $[5, 10)$ Hz Spectral power $[10, 15)$ Hz Spectral power $[15, 20)$ Hz Spectral power $[20, 25)$ Hz Spectral power $[25, 30)$ Hz Spectral power $[30, 35)$ Hz Spectral power $[35, 40)$ Hz Spectral power $[40, 45)$ Hz Spectral power $[45, 50)$ Hz	Amplitudes and directional signals.
Peak	Number of peaks Peak height (mean) Peak height (st. dev.) Peak height (skewness) Peak width (mean) Peak width (st. dev.) Peak width (skewness) Peak width height (mean) Peak width height (st. dev.) Peak width height (skewness) Peak area (mean) Peak area (st. dev.) Peak area (skewness)	Moments of acceleration and deceleration of directional signals.

Table 3.4: Overview of extracted features by domain. Data that is used to compute the features is also included.

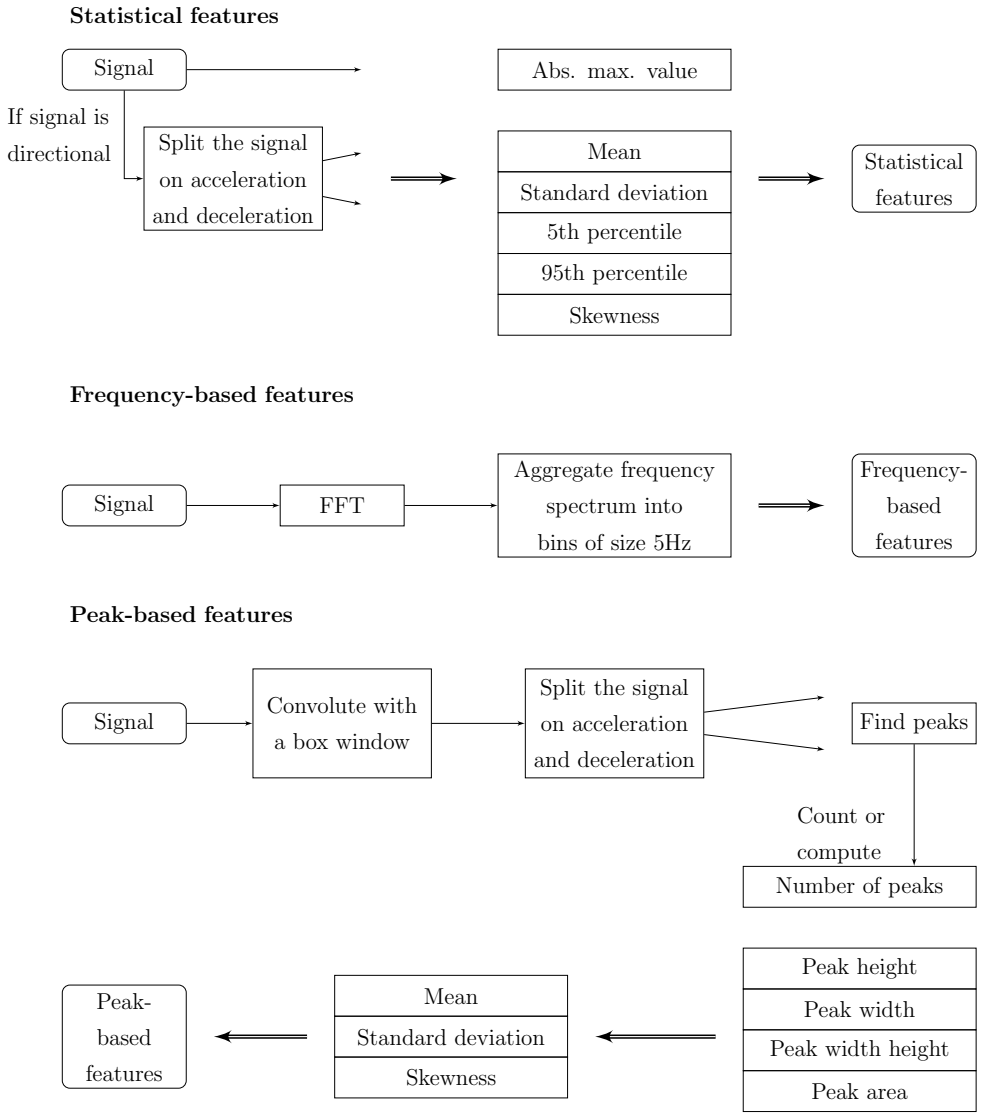

Figure 3.5: Work flows for feature extraction from preprocessed signals. At the top is work flow for extraction of statistical features, in the middle we visualized how we extract frequency-based features, while at the bottom we show diagram picturing extraction of peak-based features.

Statistical features we extract include mean, standard deviation, 5th percentile, 95th percentile, skewness, and absolute maximal value. Work flow for extraction of statistical features is pictured at the top of Figure 3.5. We use `numpy`'s functions `mean`, `std`, and `percentile` to compute mean, standard deviation, and 5th and 95th percentile, respectively. To compute skewness we use function `skew` from `scipy.stats`, whereas we use a combination of python's native `abs` and `max` functions to get absolute maximal value. We further split the signal into moments of acceleration and moments of deceleration along each axis. This means that we take measurements greater than zero, compute the features and repeat the procedure for measurements smaller than zero. Although the directions of the axes are defined by the coordinate system relative to the phone and thus depend on the phone's orientation, this gives us an idea about how strong the acceleration and deceleration along each axis are in that sample. If the phone is turned as the samples are taken, acceleration and deceleration might get flipped, however as the amplitudes remain roughly the same we do not bother with that. For example, we expect that when traveling by train acceleration and deceleration along each axis are much smaller than when driving in a car or a bus since trains usually do not perform sharp turns. If the sample includes only acceleration or only deceleration moments we use default values for the non-existing data. Default values are zero for mean, fifth percentile, 95th percentile, and skewness, and negative one for standard deviation.

Work flow for extraction of frequency-based features in the middle of Figure 3.5 shows that to calculate frequency based features, we first use the fast Fourier transform to compute one-sided power spectrum density of the signal. We use the implementation of Fast Fourier transform from `numpy.fft` library. As we are using sampling frequency of 100 Hz we are left with frequencies below 50 Hz. We normalize the power spectrum so that power densities of frequencies greater than zero sum up to one. Frequency features we compute are cumulative power spectrum densities in the following frequency intervals (measured in terms of Hz): $[0.2, 5)$, $[5, 10)$, $[10, 15)$, $[15, 20)$, $[20, 25)$, $[25, 30)$,

$[30, 35)$, $[35, 40)$, $[40, 45)$, and $[45, 50)$.

As shown in work flow for extracting peak-based features at the bottom of Figure 3.5, we first convolute the signal with a rectangular window to smooth out the signal. To do that we use function `convolute` from `scipy.signal`. We define a peak as a local extrema of width of at least 0.1 second (equivalent to 10 measurements). Peak-based features are computed for both acceleration and deceleration. Features include number of peaks, and statistics (mean, standard deviation, and skewness) computed on peak heights, widths, width heights, and peak areas. To find peaks we use function `find_peaks` from `scipy.signal`, which returns all desired peak properties except for peak areas, which we compute on our own. Instead of computing an integral over time to get the area of a peak, we use sum as a simplification. As there is usually more than one peak per signal, we aggregate the properties of peaks for each signal by calculating mean, standard variation, and skewness. We use peak-based features to characterize acceleration and deceleration periods. Amplitude is in a way invariant to acceleration and deceleration, as it measures how strong the total (axis independent) acceleration is. Therefore, we only compute peak-based features from directional signals.

3.4 Feature analysis

After feature extraction we focused on feature analysis. As the total number of features we have extracted is very large, reducing number of features used for classification might improve the performance of the classifier. Additionally, as we use signals derived from other signals we are interested in how this affects the features. Such derived signals include amplitudes d and h, which are computed as $d = d_x^2 + d_y^2 + d_z^2$ and $h = h_x^2 + h_y^2 + h_z^2$, respectively. For feature analysis we used joint train and validation sets.

First we explored the correlation between different features. Understanding the correlations between different features, feature domains, and signals, from which the features were extracted, is beneficial when constructing fea-

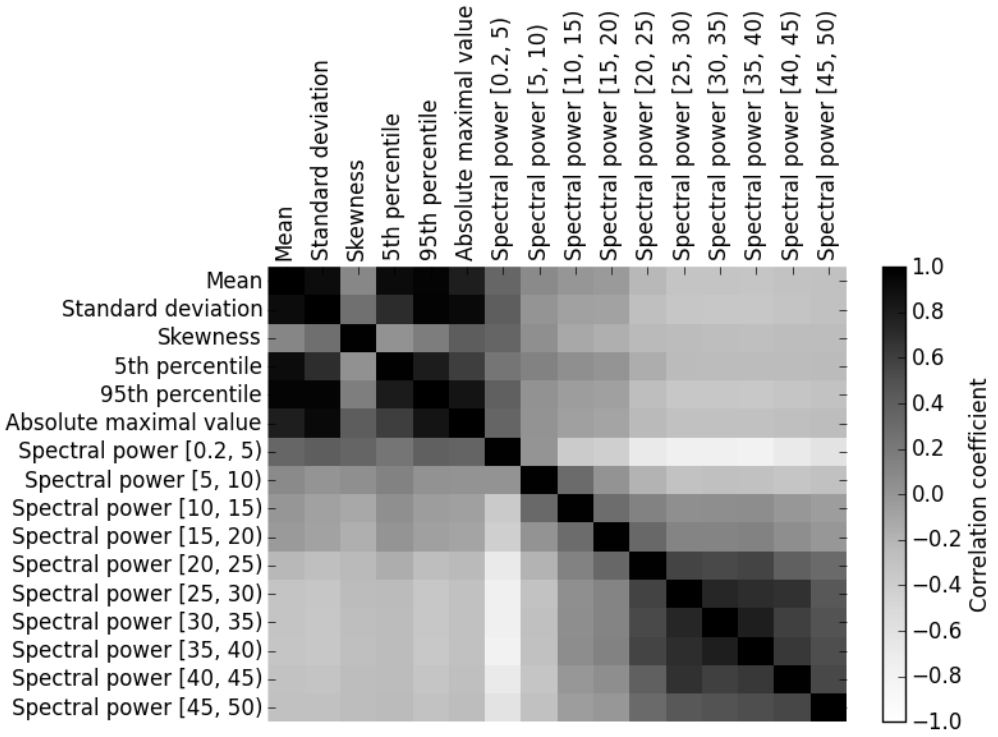

Figure 3.6: Correlation between features obtained from the amplitude of the dynamic acceleration vector d. Correlation coefficients for other accelerometer signals very much resemble those obtained from d.

Category	Signal	Feature domain			Sum
		Stat.	Freq.	Peak	
Directional signals	d_x	16	10	26	52
	d_y	16	10	26	52
	d_z	16	10	26	52
	h_x	16	10	26	52
	h_y	16	10	26	52
	h_z	16	10	26	52
Amplitudes	v	6	10	0	16
	h	6	10	0	16
	a	6	10	0	16
	d	6	10	0	16
Sum		120	100	156	**376**

Table 3.5: Number of features extracted from each signal separated by feature domain.

ture sets for classification. We expect that including less correlated features in a feature set will result in better classification performance. Initially, we were interested in how correlated are features computed from the same signal. Therefore, Figure 3.6 shows correlation coefficients between features obtained from the amplitude of dynamic acceleration. As observed in Figure 3.6 there is an apparent strong correlation between statistical features, however spectral features of amplitudes of accelerometer signals are uncorrelated. It is interesting that skewness is not correlated with any other feature.

Figure 3.7 shows correlation coefficients between features obtained from directional signals, namely dynamic acceleration along the x-axis. Similarly as in Figure 3.6 there is no strong correlation between statistical and spectral features, and statistical features are correlated with each other. Surprising exceptions are skewness and mean, which are not correlated with any other features. However, peak-based features are correlated with statistical features, which we found as unexpected. The only exception are peak width

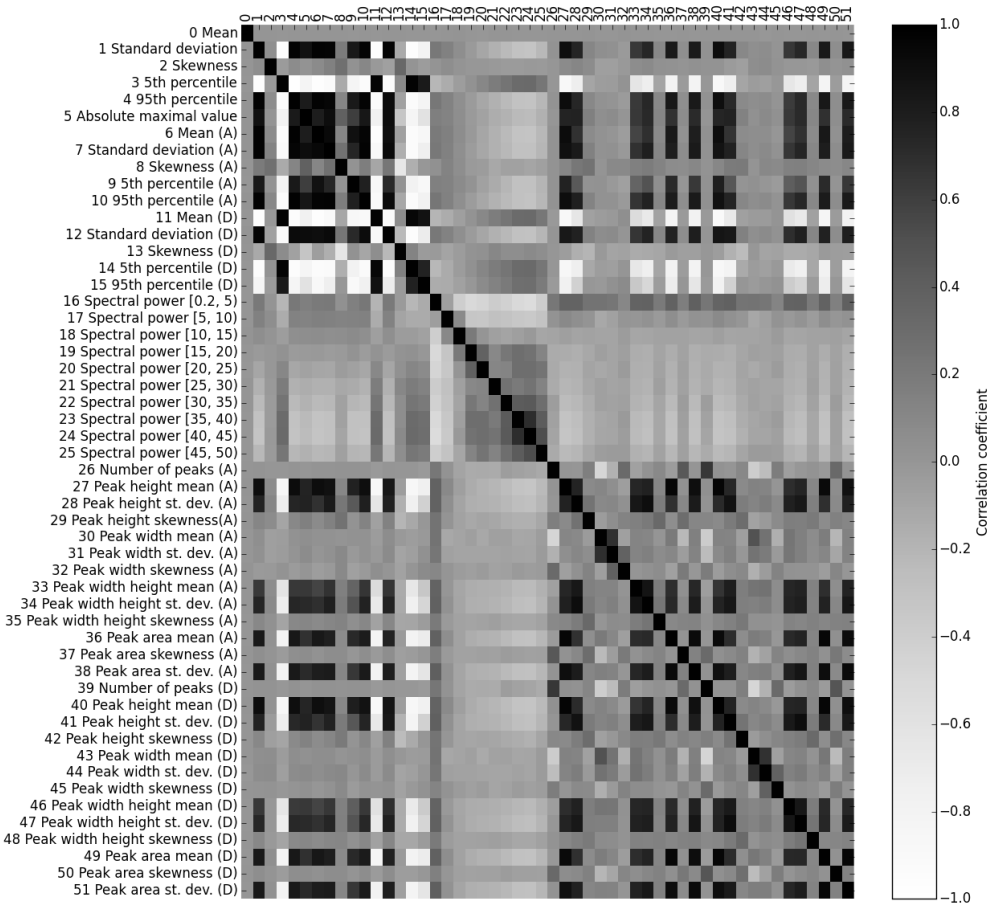

Figure 3.7: Correlation between features obtained from d_x. Correlation coefficients are similar for other non-amplitude signals. Three distinguishable sub-matrices of strong correlation on diagonal represent correlations between statistical features, peak-based features, and spectral features, respectively.

features, which are correlated with each other, but not with other peak-based or any other features. Another interesting observation is that peak-based features involving skewness are on the contrary not correlated with either statistical or peak-based features.

As some of the amplitude signals are non-linear combinations of non-amplitude signals, we tested the correlation between two such signals, namely d and d_x. Results of the test are pictured in Figure 3.8. There is a significant correlation between statistical and peak-based features, however we expected that since we observed it in Figure 3.7. There is also observable correlation between spectral features, which is the strongest for frequencies between 30 and 45 Hz. This is also expected, however we expected a stronger correlation. It is interesting how mean and skewness of directional signal are uncorrelated with any feature of an amplitude. Although we initially used Pearson's correlation coefficient to compute the correlations, which measures linear dependence between the samples, we also computed Kendall's rank correlation coefficients to measure general dependence. The results are very similar to those shown in Figure 3.8.

Furthermore, we tested whether the values of extracted features are distributed according to normal distribution. We performed D'Agostino and Pearson's test using `scipy.stats.normaltest` function on each feature. We set the significance threshold for p-value to 0.05. For most features, p-value is much smaller than the preset threshold, thus features do not follow normal distribution.

Next, we were interested if features with different values of statistical features for different labels exist and which features meet that criteria. To discover the most informative features and construct useful feature sets to train machine learning models on, we used Kruskal-Wallis H-test for independent samples. The null hypothesis of that test is that the medians of given samples are equal. Our goal was to see if the medians of features are different for cars, buses, and trains. We opted for Kruskal-Wallis H-test because it does not assume the samples are distributed according to normal

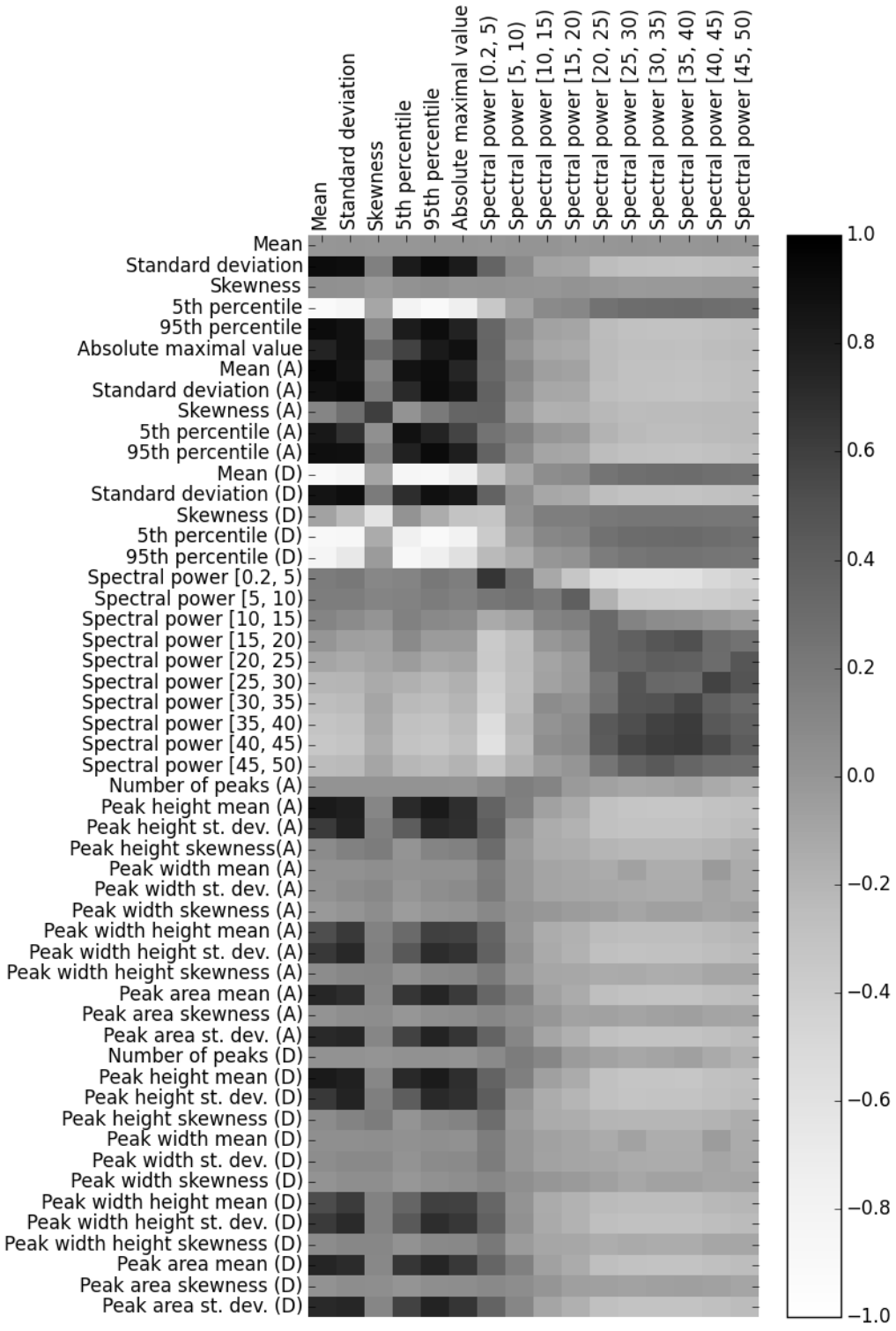

Figure 3.8: Correlation between features obtained from d and d_x.

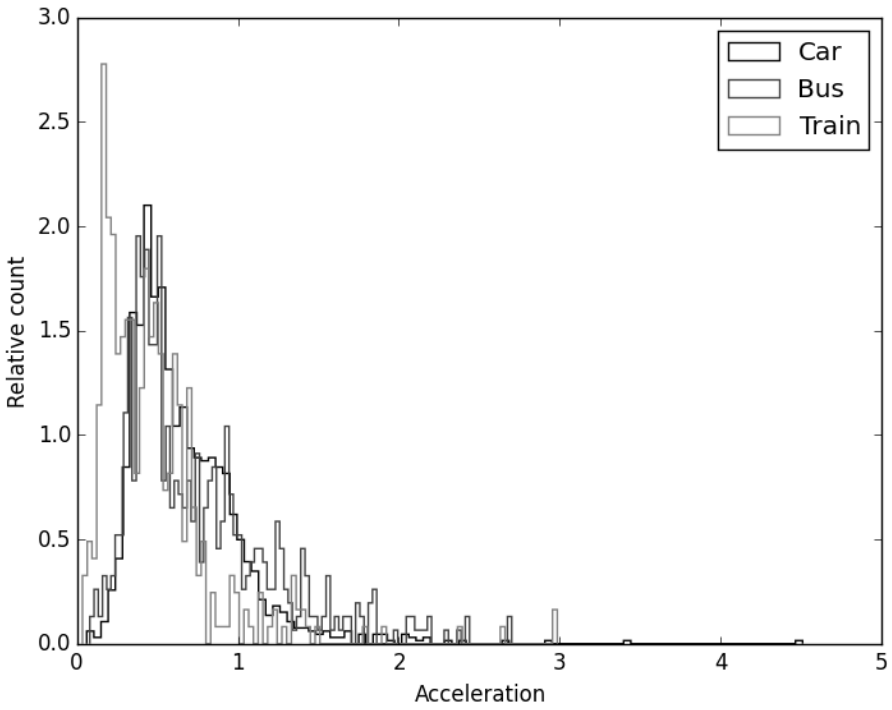

Figure 3.9: Shapes of distributions of mean acceleration for different modalities.

distribution. In order to perform this test we first divide the data in three homogeneous sets based on their labels. That means that first set contains all data points labeled as cars, the second contains all bus examples, and the third all instances labeled as trains. These three sets are our independent samples for the Kruskal-Wallis H-test. The main drawback when using this test is that it operates under the assumption that all tested distributions have the same shape. In our case, that does not hold for amplitudes, which is shown in Figure 3.9. Tested samples of features from amplitude signals have differently shaped distributions for each modality. We did not notice such issue for features from directional signal.

We applied `scipy.stats.kruskal` function to each feature extracted from directional signal to obtain the p-value. Figure 3.10 contains results

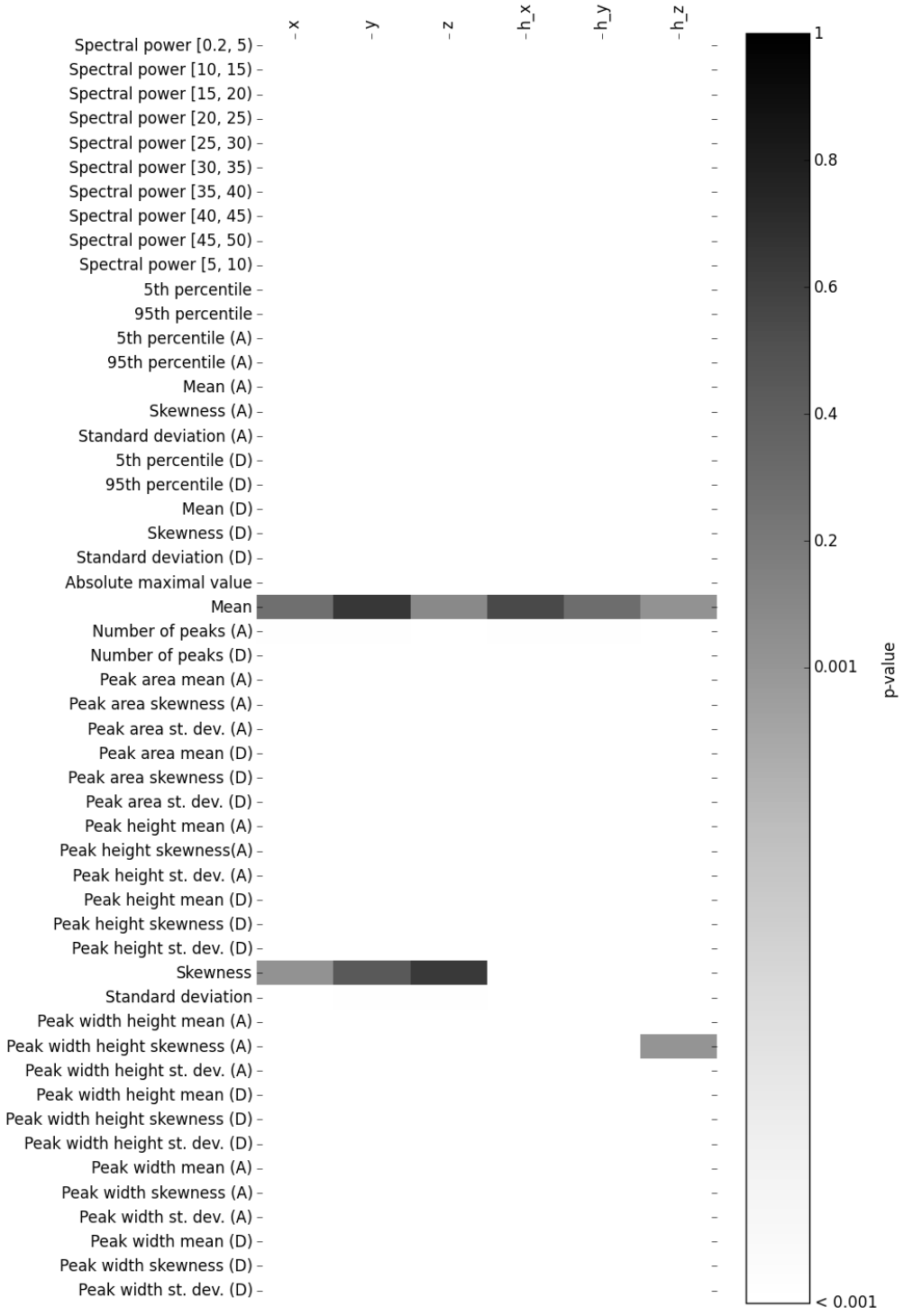

Figure 3.10: p-values of Kruskal-Wallis H-test for non-amplitude features.

Signal	Feature	p-value
h_y	Mean	0.65
h_z	Skewness	0.65
d_x	Mean	0.55
h_y	Skewness	0.45
d_y	Mean	0.30
h_x	Mean	0.28
h_z	Mean	0.09
d_z	Mean	0.03
h_x	Skewness	0.03
d_z	Peak width height (A) skewness	0.01

Table 3.6: Ten features with the highest p-value.

of the Kruskal-Wallis H-test for features from directional signals. Overall, only 8 features have p-value higher than 0.05. We listed top ten features according to p-value in Table 3.6. From the Table 3.6 we can notice that features the highest scores were also the least correlated in correlation tests. It is interesting and surprising that top nine features are statistical features, mean and skewness. Skewness in general is very poorly correlated with other features. A possible explanation is that means are representatives of a group of statistical features, so having one of those features increases the informativeness of the feature set, whereas having many of them just increases the complexity.

3.5 Classification

Using information gained in Chapter 3 Section 3.4, we constructed several feature sets in order to observe how adding a feature domain affects the classification performance. Therefore we added features gradually to be able to judge the influence of a feature group. Most feature sets are named so that the first part of the name reflect the signal used to extract the features,

whereas the second part names the feature domains included in that set. Feature sets are summarized in Table 3.7.

Statistical features are commonly used in related work, hence our first feature set D-S only contains statistical features on dynamic acceleration amplitude d and vector components d_x, d_y, and d_z. We expect this feature set to produce good results as features have high p-values in Kruskal-Wallis H-test. We used a very similar feature set previously in [25], however the methodology used for evaluation is different this time. Previously, we have used cross-validation on samples for evaluation, however we have found that approach to be methodologically questionable as we used samples from the same trip in train and test sets.

Next, we extended the D-S feature set with frequency-based features. We named that set D-SF. Similarly, we extended D-SF with peak-based features, to obtain a new feature set, D-SFP. Both D-SF and D-SFP only contain features extracted from the dynamic acceleration signals. Both of these sets have greater variety of features, however they are also much larger. We expect these sets to perform comparably to D-S. The reasoning for this is that although the variety of features increases, greater number of features can mean more noise in the data. Therefore, we do not expect either drastic improvement or significant decrease in classification performance.

In the same way as feature sets D-S, D-SF, and D-SFP, we define feature sets H-S, H-SF, and H-SFP to contain statistical and frequency-based, and statistical, frequency-based, and peak-based features, respectively. The crucial difference between these sets is that features in sets H-S, H-SF, and H-SFP are extracted from the horizontal acceleration signals h, h_x, h_y, and h_z. We are interested in comparison of dynamic acceleration and horizontal acceleration feature sets, since we can implicitly observe the influence of vertical acceleration on classification.

Feature set DIR-SPF includes all features from directional signals d_x, d_y, d_z, h_x, h_y, and h_z. Features extracted from these signals scored very high p-values in statistical domain. Additionally, we were interested in a feature set

Set	Signals	Features	Size
D-S	d, d_x, d_y, d_z	Statistical	54
D-SF	d, d_x, d_y, d_z	Statistical, Frequency	94
D-SFP	d, d_x, d_y, d_z	Statistical, Frequency, Peak	172
H-S	h, h_x, h_y, h_z	Statistical	54
H-SF	h, h_x, h_y, h_z	Statistical, Frequency	94
H-SFP	h, h_x, h_y, h_z	Statistical, Frequency, Peak	172
DIR-SFP	d_x, d_y, d_z, h_x. h_y, h_z	Statistical, Frequency, Peak	312
ALL			376
TOP			7

Table 3.7: Predefined feature sets used for classification. We constructed these feature sets based on the information gained by feature analysis.

without amplitude signal to see what the contribution of amplitudes might be. Feature set DIR-P contains only peak-based features.

Finally, feature set ALL includes all 376 extracted features, whereas feature set TOP contains 7 features with p-values larger than 0.05 in Kruskal-Wallis H-test.

We use feature sets listed in Table 3.7 to build classification models. We apply three machine learning methods to train the model. Machine learning algorithms that we use are random forest (RF), support vector machine (SVM) with radial basis function kernel, and multilayer perceptron as neural network (NN). These methods are often used in related work. In addition to exploring the feature space with the predefined feature sets from Table 3.7 we will also perform feature selection with random forest classifier.

Chapter 4

Evaluation

4.1 Performance metrics

For each mode, we count the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). For one mode, true positives are samples that belong to that mode and are classified as belonging to that mode by classifier. Similarly, true negatives are cases that do not belong to that mode and are classified as not belonging to that mode. False positives are samples that are not labeled as that mode, however the classifier recognizes them as if they were. On the contrary false negatives are labeled as belonging to that mode, but are not classified as such.

These counts are further used to calculate performance measures. As performance measures we are using F1 score, although we also report on precision, recall, and classification accuracy. Precision measures the ratio between true positives and all samples that were classified as positive for each class,

$$\text{precision} = \frac{TP}{TP + FP}. \tag{4.1}$$

Recall estimates how many samples labeled as positive are actually recognized as positive in classification,

$$\text{recall} = \frac{TP}{TP + FN}. \tag{4.2}$$

Values for precision and recall range from 0 to 1, where 0 means that no examples are true positives. When precision reaches 1, only true positives are classified as positive, whereas recall reaches 1 when no positive sample is classified as negative.

F1 score is defined as geometric ratio of precision and recall, and is calculated as

$$\mathrm{F1} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}. \tag{4.3}$$

F1 score takes values between 0 and 1. F1 score is equal to 0 when precision or recall is 0, and is equal to 1 when precision and recall are both 1.

We calculate F1 score, precision, and recall for each class separately, whereas classification accuracy is calculated for whole set. To combine the scores of all classes in one number we are using macro average, since we are aiming for a classifier that reliably recognizes each mode. We opted for F1 score as the main measure of performance because we want to reduce the number of false positives and false negatives for each class. As our data set is imbalanced, using classification accuracy as a main measure could result in poor recognition of minority modes, whereas classification accuracy would indicate that classification is working well.

4.2 Evaluation methodology

For evaluation it is common use separate testing sets that contain approximately 30% of all their data [2, 17]. Sometimes the data is randomly split into train and test sets [2]. However, cross-validation is also used [5, 3, 9, 23]. Leave-one-user-out [3], leave-one-placement-out [3], and leave-one-trip-out [5] are popular choices for cross-validation. Evaluation methodology is also sometimes unknown or unclear [2, 1, 19].

Performance is usually measured with classification accuracy [2, 4, 17, 20, 19], but as the data sets normally report some imbalance between classes precision [5, 3], recall [5, 16, 3] and F1 scores [18, 9] are often more appropriate.

Set	Dates	Trips	Samples
Train	Aug 2016 — 31 Jan 2018	*	16824
Validation	1 Feb 2018 — 14 May 2018	285	2489
Test	15 May 2018 — 15 June 2018	184	1731

Table 4.1: Basic characteristics of train, validation, and test set. Training set was collected using different methodology, therefore information about the number of trips is not available.

To evaluate the capabilities and performance of the proposed approach, we divide our dataset in 3 subsets — train, validation, and test set — based on the date the samples were recorded on. By doing so, we avoided using in this domain methodologically questionable random assignment of samples collected during the same trip to different subsets. The reason why we did not apply cross-validation is similar. Using samples from the same trip in train and test set would result in significantly higher evaluation scores. Details about train, validation and test sets are listed in Table 4.1. We described the process of data acquisition in Chapter 3. For our train set we used the *Mobility Patterns* dataset from our previous work [25] as there was not enough samples from the pilot study. For validation and test sets we use *pilot testing* dataset. As we used different approach to collect the data, information about the trip count is not available in Table 4.1.

Figure 4.1 shows the distribution of modes in all 3 sets. Train and validation set have very similar distributions, which is desired, whereas car samples are overrepresented in the test set. Although the distributions between train and validation sets, and test set vary, we decided against using over- or undersampling techniques to balance the sets. Our reason to do so is that it is not guaranteed that distribution of modes is the same throughout the year, since weather conditions usually affect the choice of transportation mode.

The scenario for evaluation is shown in Figure 4.2. We first train a model on train set and evaluate the model using validation set. We repeat the process on several sets of parameters. For final training, we select the set of

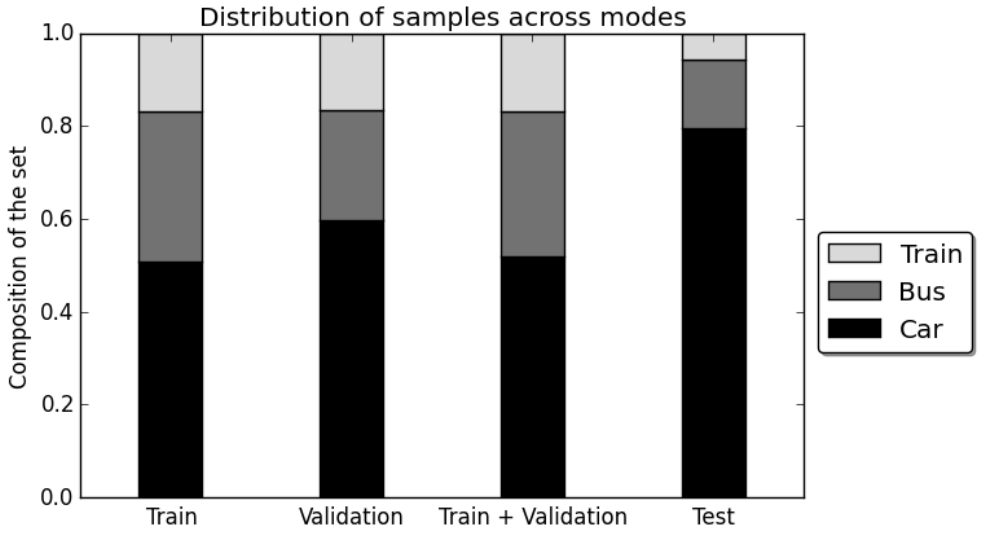

Figure 4.1: Distribution of modes in train, validation, and test set. We also added joint train and validation set, which we use to train the final model.

model parameters that performed the best on validation set and train the model on the joint train and validation set. We evaluate the performance of that model on test set.

4.3 Results

In this section we present the results and briefly discuss the findings. We first test two trivial classifiers and then move on to non-trivial classifiers — random forest, support vector machine, and neural network. As the machine learning methods we use rely on some level of randomness, we run each experiment 100 times to get statistically reliable results. Since we use an implementation of SVM based on LIBSVM [33], which is deterministic, we run all those experiments only once. We report on four performance metrics — classification accuracy, recall, precision, and F1 score — which we specified in previous sections. Additionally, we report on standard deviation (σ) for each of there values. For the most interesting feature sets, we also provide

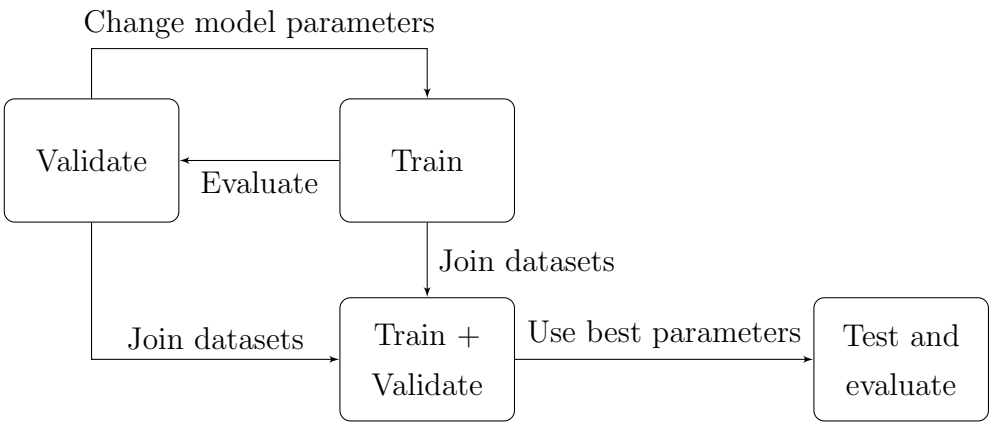

Figure 4.2: Schema of evaluation scenario.

confusion matrices.

4.3.1 Trivial classifiers

To set the lowest bar our approach must reach in terms of performance, we classified the test set using two trivial classifiers — majority classifier and random classifier. In our case, majority classifier classifies all samples as cars, whereas random classifier draws labels from the distribution of labels in combined train and validation set (shown in Figure 4.1). Results for both trivial classifiers are listed in Table 4.2. We will further use Table 4.2 when comparing and assessing performance of non-trivial classifiers. Table 4.2 shows that classifying all samples as a majority class results in F1 score of 0.30, precision and recall reach 0.26 and 0.33, respectively, whereas classification accuracy is 0.79. Using random classifier we achieve F1 score of 0.31 with precision and recall scores at 0.33 and 0.34, respectively. Classification accuracy in that case is 0.47. Confusion matrix for random classification in Table 4.3 shows that the distributions of predicted labels are the same for all labels.

Classifier	**CA**(σ)	**Recall**(σ)	**Precision**(σ)	**F1**(σ)
Majority	0.79	0.26	0.33	0.30
Random	0.47 (.006)	0.33 (.002)	0.34 (.001)	0.31 (.001)

Table 4.2: Classification metrics for classification with trivial classifiers. For random classifier, we report on the standard deviation σ after 100 runs in brackets.

T \P	Car	Bus	Train
Car	0.53 (.008)	0.29 (.009)	0.18 (.001)
Bus	0.51 (.004)	0.31 (.004)	0.18 (.001)
Train	0.46 (.035)	0.38 (.054)	0.16 (.020)

Table 4.3: Confusion matrix for random classifier. In brackets is the standard deviation σ after 100 runs.

4.3.2 Random forest

The first non-trivial classification algorithm that we used is random forest. We chose random forest because it works very well in non-linear cases, it is easy to tune its parameters to avoid over fitting, and is fairly fast to train. These things suggest that the use of random forest to explore the data is appropriate.

Initially, we trained random forest classifier on the predefined feature sets from Table 3.7. We did not use any scaling or normalization of the features, or transformation of the feature space, whereas later we experimented with these options. Results are listed in Table 4.4.

Table 4.4 shows that we achieved the highest F1 score of 0.42 using H-S feature set. This feature set consists of statistical features calculated on the horizontal acceleration vector. Classification accuracy in that case is also high, compared to other feature sets. The highest classification accuracy is the result of classification with the TOP feature set. However, the performance of the TOP feature set is closer to a majority classifier according to

other metrics.

From Table 4.4 we can observe that feature sets ALL and DIR-SFP act more as random classifiers than anything else. Although their precision scores are significantly higher than those of a random classifier from Table 4.2, all other metrics are very close to those of a random classifier. Similar also holds for H-SFP and D-SFP. These observations lead us to believe that by using more features we are introducing noise and thus randomness into the classification procedure. Additionally, since D-S, D-SF, and H-S perform better than other classifiers, peak features seem to be the source of that noise.

Another comparison we can make is regarding the use of dynamic acceleration or horizontal acceleration. Table 4.4 shows that feature set H-S outperforms other feature sets, including its dynamic acceleration equivalent D-S. It is interesting to observe that F1 score and classification accuracy improve when we add frequency-based features to dynamic acceleration, whereas these two measures decrease in case of similar action for horizontal acceleration. This offers two possible interpretations. One is that frequency-based features of dynamic acceleration carry more information compared to frequency-based features of horizontal acceleration. The second one is that statistical features of horizontal acceleration are much better than statistical features from dynamic acceleration. That means that by introducing new features to H-S we are introducing noise, whereas in case of D-S we are introducing new informative features.

Since H-S has been the most successful so far in classifying transportation modes, we are showing a confusion matrix for classification with features from H-S in Table 4.5. For bus mode, this non-trivial classifier acts as a mixture of majority and random classifiers as it classifies the majority of bus samples as cars. For trains, the prevailing predicted mode is still car, however the majority of samples is classified as either a train or a bus. Compared to random classifier there is slightly more train samples classified as trains, but overall the classification of train mode seems random.

Feature set	**CA**(σ)	**Recall**(σ)	**Precision**(σ)	**F1**(σ)
D-S	0.49 (.005)	0.41 (.003)	0.39 (.002)	0.37 (.003)
D-SF	0.57 (.017)	0.40 (.013)	0.42 (.013)	0.39 (.013)
D-SFP	0.46 (.002)	0.38 (.007)	0.39 (.005)	0.35 (.006)
H-S	0.65 (.015)	0.41 (.018)	0.44 (.010)	0.42 (.019)
H-SF	0.48 (.013)	0.37 (.015)	0.41 (.016)	0.34 (.013)
H-SFP	0.49 (.014)	0.37 (.015)	0.40 (.012)	0.34 (.012)
DIR-SFP	0.48 (.012)	0.36 (.014)	0.40 (.014)	0.33 (.011)
ALL	0.47 (.006)	0.35 (.007)	0.40 (.010)	0.33 (.008)
TOP	0.65 (.007)	0.35 (.005)	0.34 (.007)	0.34 (.006)

Table 4.4: Classification metrics for classification with random forest on predefined feature sets. In brackets is the standard deviation σ after 100 runs.

T \P	Car	Bus	Train
Car	0.76 (.015)	0.22 (.022)	0.02 (.005)
Bus	0.70 (.013)	0.23 (.021)	0.07 (.005)
Train	0.41 (.011)	0.36 (.067)	0.24 (.067)

Table 4.5: Confusion matrix for random forest classifier on feature set H-S. In brackets is the standard deviation σ after 100 runs.

To improve classification results we first tried normalization. Normalization is used to center the data near 0 with variance 1. In our case, normalization did not improve classification, furthermore in some cases it even reduced the classification scores.

The next thing we tried is principal components analysis (PCA). For all feature sets defined in Table 3.7, three principal components explain at least 80% of variance in dataset. Therefore, we used PCA to transform our high-dimensional feature spaces defined with feature sets in Table 3.7 into three dimensional space. We trained and evaluated the random forest classifier with these low dimensional features. Results are listed in Table 4.6.

Table 4.6 shows that for most feature sets reducing the dimensionality of feature space positively affects classification scores. Feature sets D-S, D-SF, H-S, H-SFP, and TOP have less than 0.05 increase in F1 score, whereas other feature sets achieve at least 0.05 higher F1 score using PCA than without transformation (Table 4.4). Again, H-S scores the highest F1 score and TOP the lowest, joint with H-SFP.

We notice that F1 score decreases significantly when adding peak-based features in case of horizontal acceleration. This is aligned with our findings in classification without PCA transformation. On the contrary, the score increases when adding peak-based features for dynamic acceleration. This suggest that there are latent features composed of statistical, frequency-based, and peak-based features from dynamic acceleration, which positively contribute to classification performance. It is interesting that there is no improvement when we add frequency-based features for dynamic as well as horizontal acceleration.

Similarly as in case without PCA, we are also interested in confusion matrix of the best performing feature set. We show the confusion matrix for H-S feature set in Table 4.7. Compared to Table 4.5, this time even more cars are correctly classified and less buses are classified as cars. We can notice that compared to 23% buses classified as buses in Table 4.5 there are 37% buses correctly classified in Table 4.7. However, the majority of trains

Feature set	**CA**(σ)	**Recall**(σ)	**Precision**(σ)	**F1**(σ)
D-S	0.60 (.014)	0.40 (.005)	0.39 (.004)	0.39 (.005)
D-SF	0.60 (.001)	0.40 (.005)	0.40 (.004)	0.39 (.004)
D-SFP	0.65 (.045)	0.42 (.014)	0.42 (.012)	0.41(.015)
H-S	0.71 (.030)	0.46 (.014)	0.47 (.016)	0.46 (.015)
H-SF	0.70 (.029)	0.46 (.138)	0.46 (.014)	0.46 (.014)
H-SFP	0.53 (.041)	0.36 (.010)	0.37 (.008)	0.34 (.013)
DIR-SFP	0.52 (.004)	0.41 (.004)	0.42 (.004)	0.39 (.004)
ALL	0.54 (.007)	0.40 (.006)	0.40 (.005)	0.38 (.005)
TOP	0.68 (.006)	0.35 (.005)	0.34 (.017)	0.34 (.006)

Table 4.6: Classification metrics for classification with random forest on predefined feature sets transformed into three dimensional space using PCA. In brackets is the standard deviation σ after 100 runs.

are classified as cars. The amount of correctly classified trains decreased only slightly, whereas the amount of trains misclassified as cars increased significantly, from 41% to 59%, which means that trains are even more often mistaken for cars.

As we have seen that reducing the dimensionality of feature space works, the next thing we focused on is feature selection. Instead of recursive feature elimination, we implemented greedy feature elimination — backward feature selection. We opted for this local optimization method as the complete search

T \P	Car	Bus	Train
Car	0.81 (.045)	0.16 (.045)	0.03 (.006)
Bus	0.59 (.057)	0.37 (.059)	0.04 (.006)
Train	0.59 (.040)	0.22 (.050)	0.20 (.028)

Table 4.7: Confusion matrix for random forest classifier on feature set H-S with PCA transformation into three dimensional space. In brackets is the standard deviation σ after 100 runs.

of feature space is computationally not feasible in our case.

With greedy approach to backward feature selection we initially train the model with all features and evaluate it on validation set. Then we remove each feature one by one, train the model, evaluate it on the validation set and compare all F1 scores. The feature we eliminate is selected from all features that when included in the model that model performed worse. We eliminate the feature, whose absence results in the highest F1 score. We repeat this procedure until there is no features to eliminate - feature set consists of one feature.

We can describe feature elimination as a top-down approach — we start with a large feature set and then reduce it to an appropriate size. We also tried bottom-up approach to feature selection — forward feature selection. We started with 376 feature sets of size one and greedily added one feature at a time until all features were included in the feature set. We called that procedure feature addition.

In both feature elimination and feature addition we should see a peak or an elbow cure when we plot F1 score depending on the number of features. That peak or elbow represents the optimal number of features, whereas we can read the optimal combination of features from the order of elimination or addition.

We used feature elimination and feature addition to select the best feature subset for classification. We show F1 score depending on the number of features in Figure 4.3. We can observe that there are large fluctuations in F1 score. There is a large drop in F1 score at around 150 features for feature addition and a smaller one at approximately 50 features. In case of elimination, drops are not as significant, however there is a near-linear decrease in F1 score after 150 features. The best F1 score we achieve on validation set is a bit over 0.50, which is better than anything so far.

Using feature elimination and addition, we selected two feature sets that performed the best — in case of addition the best feature set has 10 features, whereas feature set produced with feature elimination has 28 features.

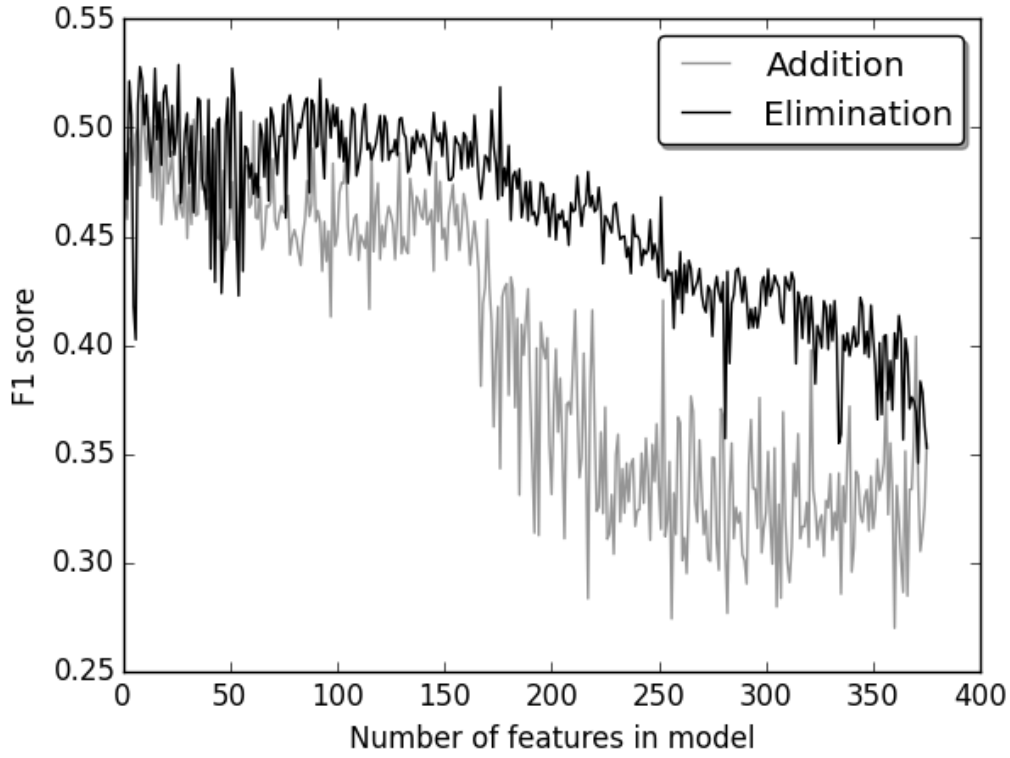

Figure 4.3: F1 score depending on number of features used in the model in feature selection.

While comparing the two sets, we noticed that the smaller set obtained by addition is a subset the feature set constructed with elimination. In fact the features from addition feature set appeared in places 2 to 11 in the reversed order of the elimination, which means that they were among the last eliminated. Feature set obtained by forward selection mostly contains statistical features, followed by peak-based. Only one frequency-based features appears in that set. Additionally, the vast majority of features are extracted from dynamic acceleration. On the other hand, feature set obtained by backward elimination contains more peak-based features than statistical, again only one frequency-based feature appears. Dynamic acceleration and horizontal acceleration appear in similar proportions.

We evaluated the models trained with the feature sets generated by fea-

Feature set	CA	Recall	Precision	F1
Addition (10)	0.69 (.010)	0.50 (.004)	0.47 (.005)	0.48 (.005)
Elimination (28)	0.74 (.007)	0.50 (.005)	0.48 (.006)	0.49 (.005)

Table 4.8: Classification metrics for classification with the selected features in feature selection. In brackets is the standard deviation σ after 100 runs.

ture selection against the test set. Results are listed in Table 4.8. Both feature sets achieve better F1 scores than any previous feature sets, however the improvement is very slight compared to feature set H-S with applied PCA. Feature set obtained by elimination performs with higher classification accuracy and F1 score than feature set obtained by addition.

Confusion matrix in Table 4.9 reveals the differences between these two feature sets. We can see that in case of eliminating features, there are fewer cars misclassified as buses and more buses misclassified as cars. Classification of trains is consistent. For buses and trains, the largest part of samples is still misclassified as cars.

From that we can learn that 18 features that differentiate feature set obtained by feature elimination and feature set obtained by addition, contribute to the increase in classification accuracy by correctly classifying more car samples. However, as more buses are classified as cars, the increase in F1 score and other selected metrics is not significant. This happens because we calculate classification accuracy for the whole data set — meaning that we count all examples that are classified correctly and divide that count by the number of all samples. As the data set is a bit imbalanced and there is more cars than buses and trains combined, a slight improvement in car classification can mean a large change in classification accuracy. We compute all other metrics on a class level, and aggregate them using macro average. The composition of the feature sets in question implies that peak-based features and features from horizontal acceleration, which have stronger representations in the elimination feature set, are not particularly useful when distinguishing between cars and buses.

Addition

T \P	Car	Bus	Train
Car	0.77 (.014)	0.18 (.014)	0.05 (.001)
Bus	0.50 (.014)	0.41 (.014)	0.09 (.002)
Train	0.48 (.013)	0.20 (.014)	0.32 (.002)

Elimination

T \P	Car	Bus	Train
Car	0.84 (.010)	0.11 (.010)	0.05 (.002)
Bus	0.57 (.014)	0.34 (.015)	0.09 (.002)
Train	0.48 (.014)	0.20 (.015)	0.32 (.007)

Table 4.9: Confusion matrix for classification with the selected features in feature selection. In brackets is the standard deviation σ after 100 runs.

We compared feature sets obtained by feature selection to feature importance scores estimated by the random forest classifier trained with all features 100 times. Frequency-based features appear to have the highest importance scores, which means that spectral features are more important. This, however, is contradicting feature selection, where in both experiments only one such feature was selected. As there is a significant difference in performance metrics between these feature sets, we believe that frequency-based features are the least informative for this problem.

4.3.3 Support vector machine

The next non-trivial classifier we used is support vector machine classifier with radial basis function (RBF) kernel. Compared to the random forest classifier, SVM classifier takes longer to train and is much easier to overfit to the training data. We used RBF kernel to cover the non-linear aspect of the data. We set the cost for misclassification for all experiments to 1 and tuned other parameters, such as gamma, which defines how much influence a single training example has.

Feature set	CA	Recall	Precision	F1
D-S	0.48	0.36	0.39	0.34
D-SF	0.49	0.39	0.44	0.36
D-SFP	0.58	0.43	0.46	0.41
H-S	0.69	0.40	0.47	0.41
H-SF	0.64	0.43	0.49	0.41
H-SFP	0.75	0.35	0.34	0.34
DIR-SFP	0.53	0.39	0.41	0.36
ALL	0.53	0.39	0.41	0.36
TOP	0.72	0.36	0.34	0.35
ADDITION	0.57	0.43	0.42	0.40
ELIMINATION	0.61	0.40	0.42	0.40

Table 4.10: Classification metrics for classification with support vector machine classifier on predefined feature sets.

Results for multiclass classification with SVM with RBF kernel are in Table 4.10. Table 4.10 shows that F1 score for classification with SVM is on average slightly higher than when classifying with random forest (results in Table 4.4) for feature sets defined in Table 3.7. Three feature sets, D-SFP, H-S, and H-SF , achieve the same highest F1 score of 0.41. However, classification accuracy for these feature sets differs significantly. We listed confusion matrices for the three best scoring feature sets according to F1 score in Table 4.11.

The first observation from matrices in Table 4.11 is that for all three feature sets classification of train mode is somewhat random. Comparison between the bottom rows of confusion matrices for D-SFP and H-S, and the bottom row of confusion matrix for random classifier in Table 4.3 reveals no major differences between the classifiers regarding classification of train samples. When using H-SF feature set, the largest part of train samples is recognized as buses. The second observation is that SVM seems to classify buses more accurately than any previous classifier on any other feature set.

D-SFP

T \P	Car	Bus	Train
Car	0.63	0.36	0.01
Bus	0.47	0.48	0.05
Train	0.45	0.36	0.19

H-S

T \P	Car	Bus	Train
Car	0.80	0.19	0.01
Bus	0.71	0.27	0.02
Train	0.50	0.37	0.13

H-SF

T \P	Car	Bus	Train
Car	0.71	0.28	0.01
Bus	0.50	0.47	0.03
Train	0.40	0.49	0.11

Table 4.11: Confusion matrices for classification with support vector machine on the selected predefined feature sets.

In case of D-SFP feature set the largest part of buses is classified as buses, and in case of H-SF parts of buses classified as cars and buses are fairly similar. SVM classifier trained with H-S act as a mixture of majority and random classifier for bus examples. Additionally, we noticed that the use of feature sets D-SFP and H-SF results in less reliable classification of cars compared to random forest and other feature sets.

To improve classification scores we again tried scaling and PCA. Just as with random forest, scaling did not improve classification scores, whereas with PCA we observed similar improvement as previously. Results are listed in Table 4.12. Compared to random forest the improvement was generally less significant. We noticed the largest increase in F1 score for feature sets H-S and H-SF, which performed the best in previous experiment as well. Interesting is that there was no significant increase in precision and classification accuracy, while recall gained 0.07 in case of H-S and 0.04 for H-SF. Confusion matrices for these two feature sets are in Table 4.13.

Overall scores from Table 4.12 suggest that models trained with H-S and H-SF are very similar, which we can also observe in confusion matrices in

Feature set	CA	Recall	Precision	F1
D-S	0.54	0.40	0.39	0.37
D-SF	0.55	0.41	0.42	0.39
D-SFP	0.53	0.35	0.37	0.35
H-S	0.69	0.47	0.49	0.46
H-SF	0.68	0.47	0.48	0.46
H-SFP	0.79	0.33	0.26	0.30
DIR-SFP	0.50	0.41	0.41	0.38
ALL	0.50	0.41	0.41	0.38
TOP	0.71	0.36	0.34	0.34
ADDITION	0.54	0.39	0.38	0.37
ELIMINATION	0.66	0.42	0.44	0.43

Table 4.12: Classification metrics for SVM classifier on predefined feature sets with PCA transformation into three dimensional space.

H-S

T \P	Car	Bus	Train
Car	0.77	0.21	0.02
Bus	0.51	0.46	0.03
Train	0.46	0.34	0.20

H-SF

T \P	Car	Bus	Train
Car	0.76	0.22	0.02
Bus	0.50	0.46	0.04
Train	0.44	0.36	0.20

Table 4.13: Confusion matrices for classification with support vector machine on the selected predefined feature sets.

Mode	Feature set	CA	Recall	Precision	F1
Car	H-S	0.57	0.60	0.57	0.53
Bus	H-SF	0.73	0.61	0.57	0.58
Train	D-SF	0.94	0.57	0.73	0.60

Table 4.14: Classification metrics for binary classification with support vector machine classifier.

Table 4.13. These two confusion matrices are also very similar to confusion matrix for classification with random forest on feature set H-S with PCA transformation into three dimensional space in Table 4.7.

In addition to running the SVM classification on the predefined feature sets from Table 3.7 we also trained the models using feature sets produced by feature selection procedure with random forest classifier. However, these feature sets did not produce good results with SVM as seen in bottom two rows of Table 4.10 and Table 4.12.

We noticed that classification for each class might work best with its own model parameters, therefore we turned to binary classification using on versus the rest approach. We used SVM classifier with RBF kernel to train binary models. Results are listed in Table 4.14. We were able to recognize 55% of cars, 42% of buses, and 13% of trains correctly with binary classification using one versus the rest strategy. This is less than with multiclass classification in the majority of cases. For each mode we were able to reach F1 score over 0.5, however as we only have two classes in binary classification, even majority classifier reaches F1 score of 0.5 on average.

We reported on the best performing feature set for each mode in column Feature set in Table 4.14. We were most successful classifying cars and buses using horizontal acceleration features, whereas trains were best recognizable with dynamic acceleration features. We believe the difference stems from the fact that trains are bound to the rails. Therefore, there is very close to zero vertical acceleration in short time intervals. On the contrary cars and buses usually deal with urban furniture, such as speed bumps, which affect vertical

acceleration.

We also used PCA with binary classification. Scores are very similar to those in Table 4.14. When we used PCA we were able to correctly classify 71% of cars, 17% of buses, and 12% of train, which is overall worse than without PCA.

These findings suggest that SVM might not be the most appropriate classifier for this problem. We, however, acknowledge that SVM allows for a number of parameters to be adjusted. This includes kernel, the cost for misclassification, and the influence of a single training example. However, our initial experimentation suggests that it is unlikely that SVM is the best approach for the problem at hand. We were maximizing the F1 score, whereas macro-averaged classification accuracy might work better. The choice of optimization criteria in validation also holds for binary classification, where we also tried to maximize F1 score, but could or even should have opted to maximize the classification accuracy of the minority class.

4.3.4 Neural network

As neural network we used multilayer perceptron with at most three layers. Using deeper networks would result in much more complex tuning of parameters and layer sizes, and would increase training time significantly. We selected an approximate range of layer sizes and then explored possible configurations using validation set to determine the best layer setting.

Results of classification with neural network are listed in Table 4.15. On average, F1 scores for classification with neural network are higher than in case of random forest or SVM. We achieved the highest scores using D-SF, D-SFP, and ALL feature sets, which is slightly unusual as we previously got the best results when using horizontal acceleration. What stands out about these two results are the recall scores, which are as high as those achieved by feature selection with random forest.

Confusion matrix for classification using ALL feature set is shown in Table 4.16. It reveals that though the classification is still not good, neural

Feature set	CA	Recall	Precision	F1
D-S	0.51 (.074)	0.39 (.020)	0.40 (.015)	0.37 (.020)
D-SF	0.64 (.042)	0.48 (.020)	0.44 (.021)	0.44 (.023)
D-SFP	0.58 (.057)	0.49 (.014)	0.48 (.020)	0.44 (.026)
H-S	0.58 (.080)	0.43 (.031)	0.45 (.035)	0.40 (.038)
H-SF	0.59 (.039)	0.46 (.020)	0.42 (.019)	0.41 (.024)
H-SFP	0.57 (.140)	0.39 (.045)	0.41 (.085)	0.35 (.045)
DIR-SFP	0.61 (.078)	0.48 (.037)	0.48 (.054)	0.42 (.044)
ALL	0.62 (.057)	0.49 (.016)	0.49 (.018)	0.44 (.028)
TOP	0.69 (.010)	0.35 (.004)	0.34 (.026)	0.34 (.003)
ADDITION	0.63 (.068)	0.49 (.033)	0.46 (.035)	0.45 (.038)
ELIMINATION	0.60 (.094)	0.43 (.033)	0.42 (.031)	0.41 (.037)

Table 4.15: Classification metrics for classification with neural network classifier on predefined feature sets. In brackets is the standard deviation σ after 100 runs.

networks do a very good job in recognizing buses as the majority of bus samples is recognized correctly. Additionally, there is significantly less trains classified as cars than in any other scenario we have tested. Classification of car samples is slightly worse than with some other models, however the difference is not significant. This means that we are able to differentiate between user traveling in a private vehicle and a user using public transport.

The fact that we are able to differentiate between traveling in cars and using public transportation is often good enough for applications focused on reducing individual's carbon footprint. Additionally, such classifier is also useful in cities, where only one form of public transport is widely available, for example in Ljubljana. In places where different means of public transport are available, city planners might be interested in whether a user is using public transport, not necessarily what the transport mode is.

We again used PCA to improve classification scores, however we were not as successful as in case of random forest and SVM. Results are gathered in

T \P	Car	Bus	Train
Car	0.65 (.086)	0.33 (.085)	0.02 (.010)
Bus	0.33 (.083)	0.63 (.085)	0.04 (.010)
Train	0.35 (.082)	0.47 (.089)	0.18 (.047)

Table 4.16: Confusion matrix for classification with support vector machine using ALL feature set. In brackets is the standard deviation σ after 100 runs.

Table 4.17. For many feature sets classification scores stayed the same or even significantly decreased. Major improvement has only happened when using H-S feature set. We achieved F1 score of 0.43, which is about the same as with feature sets D-SF, D-SFP, and ALL without PCA. Classification accuracy also roughly the same. In this case as 69% of cars are recognized as cars, 50% of buses are classified as buses and about 18% of trains are recognized as trains, which is also similar to experiments without PCA.

We did not explore transportation mode detection with neural networks in such depth as with random forest or SVM, therefore there is still room for further analysis. As we limited ourselves to networks with at most three layers, increasing the depth of the network is the first idea that comes to mind. Increasing the number of layers significantly increases the training time as well as increases the complexity of layer configuration tuning, therefore this makes feature exploration by experimenting with different feature sets very time consuming.

Additionally, we used the simplest example of neural network — multi-layer perceptron. Recently there has been a lot of progress made with different neural units, layers, and configurations in convolutional neural networks and recurrent neural networks. By using such neural networks we might be able to omit hand-crafting the features.

We have tried deeper neural networks and different neural units, however the amount of data we have poses a limitation on the number of parameters our model can have. If the model has more parameters than we have data points to train the model with, there is a new source of randomness we have

Feature set	CA	Recall	Precision	F1
D-S	0.54 (.075)	0.42 (.014)	0.41 (.011)	0.39 (.022)
D-SF	0.53 (.036)	0.42 (.008)	0.42 (.008)	0.39 (.013)
D-SFP	0.47 (.049)	0.37 (.016)	0.38 (.011)	0.34 (.023)
H-S	0.63 (.074)	0.46 (.025)	0.47 (.019)	0.43 (.030)
H-SF	0.58 (.034)	0.43 (.009)	0.47 (.017)	0.41 (.014)
H-SFP	0.64 (.169)	0.33 (.020)	0.32 (.049)	0.30 (.035)
DIR-SFP	0.47 (.146)	0.36 (.033)	0.37 (.034)	0.31 (.042)
ALL	0.42 (.084)	0.36 (.028)	0.36 (.015)	0.31 (.038)
TOP	0.69 (.011)	0.35 (.004)	0.34 (.021)	0.34 (.003)
ADDITION	0.55 (.056)	0.42 (.015)	0.40 (.015)	0.38 (.019)
ELIMINATION	0.61 (.082)	0.38 (.032)	0.39 (.037)	0.38 (.039)

Table 4.17: Classification metrics for classification with neural network classifier on predefined feature sets with PCA transformation into three dimensional space. In brackets is the standard deviation σ after 100 runs.

to deal with. When we took that limitation in account we were left with wide and shallow networks, which we have tested with multilayer perceptron, and deeper and narrow networks that did not reach F1 score of 0.33 after weeks of parameter tuning. Similarly as for other methods, we did 100 runs of each experiment and noticed that the standard deviation in performance metrics was much larger in case of deep neural networks, exceeding 0.1 for all metrics. When inspecting the confusion matrices we observed that the network was likely to recognize one mode (not necessarily the majority class) somewhat well, and randomly classify other two modes, which in turn lead to poor classification metrics.

Chapter 5

Conclusion

In this work we have presented our approach towards transportation mode detection based on mobile sensors. The approach is unique as we used short samples of only accelerometer signal. Additionally, we only collected accelerometer samples if the phone's native activity recognition API senses that there is a non-zero probability of a user traveling in a vehicle. We check the native activity recognition API every 30 seconds, which makes our approach very energy efficient and thus does not drain the battery as much. Furthermore, the users were allowed and even encouraged to use the phone during travel, which sets our approach apart from the related work. As we do not rely on any location information for transportation mode detection, the design allows the use of this approach anywhere in the world. We also did not use any personalized models, so our approach does not suffer from the so called cold start, when a new user joins the study. We carried our a comprehensive and rigid evaluation on separate datasets, which showed that although transportation mode detection using mobile phone sensors is possible, decent classification results in classification of motorized modes are difficult to achieve. Our most important discoveries and contributions include:

1. a comprehensive analysis of multiple machine learning approaches relying on a large number of features extracted from short samples of

accelerometer readings for transport mode detection,

2. identification of dynamic and horizontal acceleration-related features, in particular statistical and peak-based, as the most promising features on which future efforts in the area of transport mode inference should focus, and

3. outline the limits of short accelerometer samples-based transport mode detection by showing that such low-cost low-power practical solution is a promising tool for differentiating between riding in a private or a public vehicle, however, it fails to successfully distinguish among different modes of public transport (e.g. bus or train).

Although we believe that comparing our results to the related work would be unfair due to the significant differences in the approaches and evaluation methodologies, we acknowledge that our results are poor compared to some of the previously published work.

One contributing factor to the difficulty of the problem might be that we use short samples of sensor data, whereas the best results in the field used trip-long data samples to extract features from. We assume that by using only five second samples we miss some of the significant and mode-characteristic drive patterns. These patterns possibly include stopping at a bus station or at traffic lights, reducing the speed before driving over a speed bump or turning into a narrow street, and walking patterns inside a vehicle that would suggest a user is taking a bus or a train.

For the future work we are interested to see how using longer sensor samples influences classification. There is a trade-off between the length of the samples and the efficiency of the approach. Longer samples consume more battery, require more storage space and computation time. It would be interesting to experiment with the sizes of the samples to determine the optimal recording length. We suggest sampling at the highest frequency possible, as there has not been a lot of research done in that direction. If lower sampling frequencies turn out as optimal, down sampling to get the

best prototype is always an option, whereas up sampling does not really make sense for this specific problem. Other sensors, such as gyroscope and magnetometer, could also be used. As many scenarios for applications of transportation mode detection include GPS tracking anyway, GPS speed could be used as a feature. The main drawback of using GPS speed is its unreliability in closed spaces and underground, which means that the measurements may not be present when needed.

Additionally, we did not instruct the users on preferred placements of the device during travel nor did we discourage them from using the device. Moreover, as Optimum application is primarily a multi-modal routing application, we expected the users will interact with the phone while they travel. However, combining users' interaction with the device with the use of short samples may have also contributed to poor performance. Similar studies instruct their participants to keep the device in the same position during the trip or even instruct them on what the position should be. As the acceleration caused by the user's movement is usually three to five times stronger than that of a vehicle [9], a short gesture, such as taking a phone out of a bag, covers a fairly significant portion of the sample. Although gravity estimation is in place, it cannot nullify the event of such magnitude.

As an alternative to gravity estimation, we suggest exploring other motion sensor options provided by native OS's APIs, such as estimated gravity along each axis. Some of these options require calibration and are thus not ideal as they require user's involvement to initialize the application. To tackle the problem of phones orientation in space, approaches from gaming can also be applied. For example, Android's sensors are capable of extracting rotation matrix by joining accelerometer and magnetometer readings [34]. Rotation matrix includes rotation angles — azimuth, pitch, and roll, which define the orientation of the phone in the space. Using this information we can then transform the measurements from the coordinate system relative to the phone to the coordinate system used by the application.

Feature analysis revealed that our features are not normally distributed, which is not an issue per se. However as the shapes of distributions of some features are different for each mode, statistical testing is challenging. We relied on statistical tests to gain introspective into features, but instead we ended up performing feature selection with random forest to learn about the features and the data set. When it comes to feature selection we could argue that the simplest and the most intuitive features work the best in this case. Adding more complicated features did not result in much wanted performance improvement. Using feature selection and estimated feature importance when training with random forest we learned that frequency-based features are likely the least informative domain as only one of these features appeared in each of the selected feature sets. We also noticed that peak-based features and features extracted from horizontal acceleration do not contribute much when distinguishing between cars and buses.

As we are dealing with multiclass classification we learned that using a classifier designed and capable of handling multiple classes resulted in better classification scores. Support vector machine uses several instances of one versus the rest binary classifiers and then combines the results into one prediction. We noticed that SVM classifier performed the poorest compared to random forest and neural network, which are both capable of multiclass classification as is. We believe that it is unlikely that SVM is the best method for the given problem.

Although we reached the highest F1 score using random forest, we consider we achieved the best classification result with neural network as it was able to classify the majority of two modes correctly. This calls for further inspection of alternative evaluation criteria that would still be of practical interest. To some extent evaluation metrics depend on the practical application of the prototype. For instance, city planners might be interested in whether a user is using public transport, not necessarily if the transport mode is bus or train. The best model achieved classification accuracy of 0.62, recall of 0.49, precision of 0.49, and F1 score of 0.44. It correctly classified 65%

cars, 63% buses, and 18% trains. As we limited the number of layers in our neural network to three, there is an opportunity to improve our result by building a deeper network. We did not explore new alternatives in form of convolutional and recurrent neural networks, which gained the popularity in last years. We have tried deep learning approaches, however we realized that the number of parameters in the model quickly exceeded the number of data points we used for training.

Another idea for the future is using deep learning to avoid feature extraction. As already mentioned in the related work, when we rely on handcrafted features, we also rely on the experience of the researchers. Using traditional machine learning approaches allows the researcher to understand and explain which features and patterns are important, whereas deep learning acts as a black box. Understanding of properties and the ability to explain why some transportation mode was detected is important as nowadays the way we travel defines us. The choice of a transportation mode an individual uses daily is a reflection of her habits and lifestyle choices. Socio-economic status, attitude towards the environment, and the desire to be healthy and fit are among common contributing factors to ones lifestyle choices. However, we should also explore other promising methods.

In case of a community driven setting, the biggest challenge of data collection is finding the right incentives for people to participate [12]. Optimum project, which was our source of sensor data, offered free monthly passes to test users for public transportation in Ljubljana, and awarded test users monetary awards for using sustainable ways to commute in Vienna and Birmingham [7]. However, collaborating on such projects is not always an option. Additionally, the amount of data we collected was not adequate for all the methods we attempted to use. Due to the nature of the mobility studies and privacy concerns regarding the use of mobile phone sensor data for transport mode inference, there are very few publicly available data sets. Since there are quite a few different approaches to data collection and feature extraction, none of them were appropriate for our work. Therefore, as one of the final

challenge for the future we propose creating a benchmark data set for transportation mode detection with raw sensor samples and description of data acquisition procedure included. Having an openly accessible benchmark data set does not only help the researchers to compare their approach to others, but also accelerates further development of the field.

Bibliography

[1] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2):13, 2010.

[2] Muhammad Awais Shafique and Eiji Hato. Use of acceleration data for transportation mode prediction. *Transportation*, 42(1):163–188, 2015.

[3] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *11th ACM Conference on Embedded Networked Sensor Systems*, Roma, Italy, November 2013. ACM.

[4] Leon Stenneth, Ouri Wolfson, Philip S Yu, and Bo Xu. Transportation mode detection using mobile phones and gis information. In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, IL, USA, November 2011. ACM.

[5] Peter Widhalm, Philippe Nitsche, and Norbert Brändie. Transport mode detection with realistic smartphone sensor data. In *21st International Conference on Pattern Recognition*, Tsukuba, Japan, November 2012. IEEE.

[6] Luka Bradeško, Zala Herga, Matej Senožetnik, Tine Šubic, and Jasna Urbančič. Optimum project: Geospatial data analysis for sustainable mobility. In *24th ACM SIGKDD International Conference on*

Knowledge Discovery & Data Mining Project Showcase Track, London, UK, August 2018. ACM. http://www.kdd.org/kdd2018/files/project-showcase/KDD18_paper_1797.pdf.

[7] Optimum project - European Union's Horizon 2020 research and innovation programme under grant agreement No 636160-2. http://www.optimumproject.eu/, 2017. [Online; accessed 4-November-2017].

[8] David Mizell. Using gravity to estimate accelerometer orientation. In *7th IEEE International Symposium on Wearable Computers*, White Plains, NY, USA, October 2003. IEEE.

[9] Ke-Yu Chen, Rahul C Shah, Jonathan Huang, and Lama Nachman. Mago: Mode of transport inference using the hall-effect magnetic sensor and accelerometer. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2):8, 2017.

[10] Dennis Kroll and Klaus David. Measuring the capability of smartphones for executing context algorithms. *INFORMATIK 2017*, 2017.

[11] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *1st International symposium on handheld and ubiquitous computing*, Karlsruhe, Germany, September 1999. Springer.

[12] Nathan Eagle and Kate Greene. *Reality mining: Using big data to engineer a better world*. MIT Press, 2014.

[13] Evangelia Anagnostopoulou, Jasna Urbančič, Efthimios Bothos, Babis Magoutas, Luka Bradesko, Johann Schrammel, and Gregoris Mentzas. From mobility patterns to behavioural change: leveraging travel behaviour and personality profiles to nudge for sustainable transportation. *Journal of Intelligent Information Systems*, Oct 2018.

[14] `ActivityRecognition`. `https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognition`, 2018. [Online; accessed 31-August-2018].

[15] `CMMotionActivity`. `https://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CMMotionActivity_class/index.html#//apple_ref/occ/cl/CMMotionActivity`, 2018. [Online; accessed 31-August-2018].

[16] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on gps data. In *10th International Conference on Ubiquitous Computing*, Seoul, Korea, September 2008. ACM.

[17] Ivana Semanjski, Sidharta Gautama, Rein Ahas, and Frank Witlox. Spatial context mining approach for transport mode recognition from mobile sensed big data. *Computers, Environment and Urban Systems*, 66:38–52, 2017.

[18] Heikki Mäenpää, Andrei Lobov, and Jose L Martinez Lastra. Travel mode estimation for multi-modal journey planner. *Transportation Research Part C: Emerging Technologies*, 82:273–289, 2017.

[19] Shih-Hau Fang, Hao-Hsiang Liao, Yu-Xiang Fei, Kai-Hsiang Chen, Jen-Wei Huang, Yu-Ding Lu, and Yu Tsao. Transportation modes classification using sensors on smartphones. *Sensors*, 16(8):1324, 2016.

[20] Sungyong Lee, Jinsung Lee, and Kyunghan Lee. Vehiclesense: A reliable sound-based transportation mode recognition system for smartphones. In *18th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Macao, China, June 2017. IEEE.

[21] Shih-Hau Fang, Yu-Xaing Fei, Zhezhuang Xu, and Yu Tsao. Learning transportation modes from smartphone sensors based on deep neural network. *IEEE Sensors Journal*, 17(18):6111–6118, 2017.

[22] Toan H Vu, Le Dung, and Jia-Ching Wang. Transportation mode detection on mobile devices using recurrent nets. In *24th ACM Multimedia conference*, Amsterdam, Netherlands, October 2016. ACM.

[23] Hao Wang, GaoJun Liu, Jianyong Duan, and Lei Zhang. Detecting transportation modes using deep neural network. *IEICE TRANSACTIONS on Information and Systems*, 100(5):1132–1135, 2017.

[24] Davide Figo, Pedro C Diniz, Diogo R Ferreira, and João M Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662, 2010.

[25] Jasna Urbančič, Luka Bradeško, and Matej Senožetnik. Near real-time transportation mode detection based on accelerometer readings. In *Information Society, Data Mining and Data Warehouses SiKDD*, Ljubljana, Slovenia, October 2016.

[26] Jasna Urbancic, Veljko Pejovic, and Dunja Mladenic. Transportation mode detection using random forest. In *Information Society, Data Mining and Data Warehouses SiKDD*, Ljubljana, Slovenia, October 2018.

[27] Bao Wang, Linjie Gao, and Zhicai Juan. Travel mode detection using gps data and socioeconomic attributes based on a random forest classifier. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[28] Luka Bradeško, Michael Witbrock, Janez Starc, Zala Herga, Marko Grobelnik, and Dunja Mladenić. Curious cat–mobile, context-aware conversational crowdsourcing knowledge acquisition. *ACM Transactions on Information Systems*, 35(4):33, 2017.

[29] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. In *2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, Philadelphia, Pennsylvania, USA, October 2004. ACM.

[30] OPTIMUM Intelligent Mobility for Android. https://play.google.com/store/apps/details?id=com.fluidtime.android.ec.optimum, 2018. [Online; accessed 4-June-2018].

[31] Mobility Patterns for Android. https://play.google.com/store/apps/details?id=net.nextpin.example.collection, 2018. [Online; accessed 4-June-2018].

[32] Mobility Patterns for iOS. https://itunes.apple.com/us/app/mobility-patterns/id1073388328?mt=8, 2018. [Online; accessed 4-June-2018].

[33] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[34] Sensors overview. https://developer.android.com/guide/topics/sensors/sensors_overview.html, 2016. [Online; accessed 31-August-2018].